Mikko Apiola · Sonsoles López-Pernas
Mohammed Saqr *Editors*

# Past, Present and Future of Computing Education Research

## A Global Perspective

Springer

# Past, Present and Future of Computing Education Research

Mikko Apiola • Sonsoles López-Pernas •
Mohammed Saqr
Editors

# Past, Present and Future of Computing Education Research

## A Global Perspective

Springer

*Editors*
Mikko Apiola
University of Eastern Finland
Joensuu, Finland

Sonsoles López-Pernas
University of Eastern Finland
Joensuu, Finland

Mohammed Saqr
University of Eastern Finland
Joensuu, Finland

ISBN 978-3-031-25335-5 ISBN 978-3-031-25336-2 (eBook)
https://doi.org/10.1007/978-3-031-25336-2



This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

*To our families, friends, and loved ones.*

*A special dedication from Mohammed to his daughters, the exquisite roses, Carmen and Layla.*

# Preface

From a farewell to a book project, that is simply the short story of our book. The three editors met in Helsinki to say *adios* to our colleague Sonsoles on her way back to Madrid from a research visit to the University of Eastern Finland. A long chat against a backdrop of Finnish summer, Helsinki islands, and some cold drinks culminated into a serious conversation about a future project. We began to see the need and opportunity for a large-scale analysis of the evolution of computing education research. By using a mixture of modern scientometrics, meta-analyses, and reviews, we could potentially bring forth numerous previously unseen insights of authors, collaboration, articles, themes of research, citation practices, and regional and topical sub-communities of CER. Thus, we set our sights on an expedition to comprehensively map how computing education research started, evolved, the hands that built the field and the communities that spread the word. There, in Helsinki, the book project idea was born and therefore was codenamed the Helsinki project.

Nevertheless, a book on computing education research requires expertise, diverse perspectives, and inclusivity. Building on our networks of connections, we were privileged to have great colleagues join this project, colleagues who were among the most important contributors to the field, or as we call them later, the hands that build computing education research. As such, our early meetings and preparation were joined by Matti Tedre (University of Eastern Finland), Lauri Malmi (Aalto University), Arnold Pears (KTH Royal Institute of Technology), Mats Daniels (Uppsala University), and Simon (Unaffiliated), the latter four agreed to act as associate editors. Several refinements, ideas, and new chapters were conceptualized. Most importantly, the project was strengthened by the insights about what computing education is and where it is heading through the expertise of some of the founders of the field.

To diversify the authorship and gather as many diverse perspectives as we possibly could, invitations were sent to potential authors within the international CER community, and soon a significant number of prominent researchers in the

field showed their interest. To widen the scope of our book and to be inclusive, we launched a public call for chapters for all interested researchers and sent the link through numerous channels. The public call attracted new chapters that further enriched the project beyond our expectations.

To ensure that the book meets the highest academic standards, a rigorous peer review was arranged for each chapter. Each chapter, except the Introduction, was assigned an associate editor (AE) to coordinate the review process. This included inviting three reviewers for each chapter. The AE coordinated the review process, wrote a meta review after receiving the original reviews, and communicated these to the authors. After receiving the revised chapter texts, the AE checked that the review comments had been adequately addressed, and gave any additional recommendations on the second and further revised versions, until the chapter was finally accepted.

Our book contains contributions from several countries, including their experiences and the status of the field. We have ensured – as an editorial team – to open the doors for any group of researchers who wanted to offer such a local experience, and granting the researchers the freedom to tell it as it happened and as they wished the world knew about it. However, it is important to note that our book is not without limitations. For example, some geographical areas are currently not well covered, including Germany, Nordic Countries, South America, India, China, Africa, and parts of North America. In addition, many methodological approaches have been left unexplored. With this said, we consider this project as a beginning rather than an end. Indeed, this book also opens many avenues for future research, to be presented in future editions of this book, and in current and new academic forums.

We believe that we presented a fresh narrative of the field of computing education research, with coverage of authors, their countries of origin, themes of research, citation patterns, and collaboration networks, analysis of regional communities and sub-communities, and of specific capacity-building initiatives. Overall, we have provided new perspectives into the present, past, and future of computing education research as an academic field. We also feel that our approach brings a new methodological extension for research in computing education that complements and extends literature reviews, meta-studies, and historical perspectives. Writing this book has also brought new challenges, many happy moments, nice memories, and an opportunity to meet great people, talk about life and science, and widen our perspectives.

We are grateful for the steadfast support of the associate editors who believed in the mission of the book from the first moment. AEs have worked along the journey from the very inception of the idea to the last moment. We are also thankful to all the reviewers who took the burden of reviewing book chapters during the summer and provided valuable feedback. A special thanks goes to the following people who accepted to work as external reviewers: Mike Joy, Neena Thota, Anthony Robins, Enrique Barra, Aldo Gordillo, Ian Utting, Tony Clear, Kristin Searle, Åsa Cajancer, and Diana Franklin. Of course, we are also thankful for the authors who committed

to writing this book, following the deadlines, and doing their best to address the lengthy reviews by the reviewers and editors on multiple revision rounds. Springer people have been very supportive from the first to the last moment. We can't wait to work with all these amazing people again.

Mikko Apiola
Sonsoles López-Pernas
Mohammed Saqr

# Contents

# Editors and Contributors

## Editors

**Mikko Apiola** University of Eastern Finland, Joensuu, Finland

**Sonsoles López-Pernas** University of Eastern Finland, Joensuu, Finland

**Mohammed Saqr** University of Eastern Finland, Joensuu, Finland

## Associate Editors

**Lauri Malmi** Aalto University, Espoo, Finland

**Arnold Pears** KTH Royal Institute of Technology, Stockholm, Sweden

**Mats Daniels** Uppsala University, Uppsala, Sweden

**Simon** Newcastle, NSW, Australia

## Contributors

**Friday Joseph Agbo** University of Eastern Finland, Joensuu, Finland
Willamette University, Salem, OR, USA

**Michal Armoni** Weizmann Institute of Science, Rehovot, Israel

**Brett A. Becker** University College Dublin, Dublin, Ireland

**Anders Berglund** Uppsala University, Uppsala, Sweden

**Michaela Black** Ulster University, Coleraine, UK

**Juris Borzovs** University of Latvia, Rīga, Latvia

**Steven Bradley** University of Durham, Durham, UK

**Tom Crick** Swansea University, Swansea, UK

**Valentina Dagienė** Vilnius University, Vilnius, Lithuania

**Judith Gal-Ezer** The Open University of Israel, Ra'anana, Israel

**Natasa Grgurina** University of Groningen, Groningen, Netherlands

**Yasemin Gülbahar** Ankara University, Ankara, Turkey

**Arto Hellas** Aalto University, Espoo, Finland

**Petri Ihantola** University of Helsinki, Helsinki, Finland

**Ville Isomöttönen** University of Jyväskylä, Jyväskylä, Finland

**Ilkka Jormanainen** University of Eastern Finland, Joensuu, Finland

**Yasmin B. Kafai** University of Pennsylvania, Philadelphia, PA, USA

**Terhi Kilamo** University of Tampere, Tampere, Finland

**Päivi Kinnunen** University of Helsinki, Helsinki, Finland

**Antti Knutas** Lappeenranta-Lahti University of Technology, Lappeenranta, Finland

**Ari Korhonen** Aalto University, Espoo, Finland

**Mikko-Jussi Laakso** University of Turku, Turku, Finland

**Mart Laanpere** University of Tallinn, Tallinn, Estonia

**Andrew Luxton-Reilly** The University of Auckland, Auckland, New Zealand

**Joseph Maguire** University of Glasgow, Glasgow, UK

**Robert McCartney** University of Connecticut, Mansfield, CT, USA

**Luis Morales-Navarro** University of Pennsylvania, Philadelphia, PA, USA

**Maria Ntinda** University of Namibia, Windhoek, Namibia

**Timo Poranen** University of Tampere, Tampere, Finland

**Keith Quille** TU Dublin, Dublin, Ireland

**Tapio Salakoski** University of Turku, Turku, Finland

**Kate Sanders** Rhode Island College, Providence, RI, USA

**Sue Sentance** Raspberry Pi Foundation, Cambridge, UK

**Judy Sheard** Monash University, Melbourne, VIC, Australia

**Jane Sinclair** University of Warwick, Coventry, UK

**Gabrielė Stupurienė** Vilnius University, Vilnius, Lithuania

**Jarkko Suhonen** University of Eastern Finland, Joensuu, Finland

**Erkki Sutinen** University of Turku, Turku, Finland

**Claudia Szabo** The University of Adelaide, Adelaide, SA, Australia

**Matti Tedre** University of Eastern Finland, Joensuu, Finland

# Exploring the Past, Present and Future of Computing Education Research: An Introduction

**Mikko Apiola, Sonsoles López-Pernas, Mohammed Saqr, Lauri Malmi, and Mats Daniels**

## 1 Introduction

While computing has been practiced since ancient times, the record-breaking speed with which computations can be performed today has brought about whole new concerns, challenges and opportunities. Our society has become increasingly dependent on computational devices, and we have generations of people who are actively using and being influenced by digital technologies. Our reliance on technology has brought forth fundamental new questions around how the new power of fast computations can, can not, should, and should not be used. In addition to many benefits and opportunities, new technologies bring forth previously unseen social and ethical dilemmas. Computing skills are required for many jobs, and needed for equal participation in building the information society. The rapid changes brought about by the megatrend of computerisation highlight the importance of computing education, and computing education research (CER).

More than ever, computing education needs constant rethinking and reshaping. It is important to deeply reflect on which computing topics and skills should be taught, to whom, and by what means. There are a number of open questions that need answers. Reliable knowledge is needed about how different computing topics are learned; what is the impact of specific educational interventions or pedagogical

M. Apiola (✉) · S. López-Pernas · M. Saqr
University of Eastern Finland, Joensuu, Finland
e-mail: mikko.apiola@uef.fi; sonsoles.lopez@uef.fi; mohammed.saqr@uef.fi

L. Malmi
Aalto University, Espoo, Finland
e-mail: lauri.malmi@aalto.fi

M. Daniels
Uppsala University, Uppsala, Sweden
e-mail: mats.daniels@it.uu.se



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_1

innovations on students' learning process covering its various aspects, such as attitudes, motivation, studying practices, and learning results; how to contextualise computing education in different parts of the world, or how to best support learning that is based on creating and inventing. Research in computing education (CER) produces scientific knowledge about teaching and learning, beyond individual opinions, and such knowledge is often directly applicable to teaching practice.

CER started as an activity where computing teachers gathered together to share best practices with each other. Over the course of time, CER became a recognised academic discipline with an increasing number of scholars working on the field, new professorial appointments, launch of new research conferences, and expanded focus on new topics such as informal and life-long learning or AI in education [7]. In this book, our aim is to present a new perspective into the evolution of the scientific discipline of CER, by offering a combination of historical overviews, meta-research and reviews, case studies, and scientometric studies to reveal insights into the emergence, growth and present state of the scientific discipline of CER. Meta-analyses and reviews will delve into the evolution of research methods and theory use in publications of CER, from the early times of publishing mainly experience reports to the present day when publication venues require rigorous use of methods and theory. Case studies present the development of the field within specific and prolific communities of practice. Scientometric methods are used in an attempt to map the evolution of the communities and networks, central research themes, shifts in research focus, birth of publication venues, foundational and awarded work, and citation practices.

We hope to offer readers practical guidelines, highlights of topical areas of research, ideas for whom to connect with, where to publish, and what research methods to use. In this book, we wish to paint a picture of influential research, influential researchers, but also that of diversity. In all, this book offers a new perspective to the past, present, and future of CER, which is hopefully of interest to educational practitioners, researchers, students, the general public, and beyond.

## *1.1 Audience and Related Works*

The primary target audience of the book is computing education researchers, from the junior levels to established researchers, and new faculty members entering the field of CER. The book does not have any prerequisites. However, certain chapters will introduce methods for data analysis that might require basic understanding of statistics. The book may also be of interest to teachers and educational practitioners, learning designers, educational managers, policy makers, and other educators or officials. These may include officials working for the ministries of education of governments, industry practitioners, education administrators, policy makers, or any organizations that may be interested in developing and improving their computing education programs.

There are a number of individual research articles, books and handbooks of CER available, from the seminal book *Computer Science Education Research* [3]

in 2004, to more recent ones focusing on computing education in schools [8], and books that cover integration of computing with other disciplines [1], and books that focus more on educational practice than how research is done [6]. The Cambridge Handbook of Computing Education Research [5] covers methodological and theoretical approaches, how to do research, how to teach, common topics of research, common educational technology tools, providing a compilation of key information on methods, topics and general principles in CER [5]. While it also includes some meta-analyses of CER publications, its main focus is elsewhere. This book offers an alternative and complementing perspective to other published books with its unique focus on analysing publication metadata with scientometrics, on large-scale studies, meta-analyses, combined with narratives of development of CER, and case studies of the evolution of practicing research groups and regional research communities.

Different names have been used to denote the field. Some of the most used ones include "computer science education research" [3], "computing education (CEd)" which has been used to refer to teaching practices without a research component, and "computing education research (CEdR)", with a research component added [4]. In this book, our decision is to use the inclusive phrase "computing education research (CER)", which is widely used in the field [5].

## 2 Organisation of the Book

After the present introduction, the book begins with three chapters that lay out the foundations for understanding the field of CER. First, chapter "What is Computing Education Research (CER)" seeks to describe and define CER: it positions CER as a social science that deals heavily with human participants, and more specifically as an area of discipline-based education research (DBER). The reader is introduced to mainstream discussions and debates of the disciplinary identity of CER, major approaches for classifying CER publications, and mainstream focal areas such as programming education. The chapter higlights central debates, such as frictions caused by CER being a social science, while many CER researchers are trained in computer science.

The foundations of CER are deepened in chapter "Theory and Approaches to Computing Education Research", which introduces the reader to the role of research approach, methodology, and study design in CER. The chapter discusses the role of theory in CER, and points out the pragmatic focus of addressing concrete teaching and learning challenges as paramount in CER, where the questions and nature of useful answers dictate the method and data collection to a greater extent than in general educational research. The chapter also discusses the recent trend of empiricism, and its potential good and bad sides. In all, the chapter lays out a discussion of research quality and rigor, used frameworks and models, and portrays CER as a systematic way of applying scholarly values to understand educational activities in the context of computing.

Chapter “The Evolution of Computing Education Research: A Meta-Analytic Perspective” presents a comprehensive overview of conducted meta-studies of CER. The chapter reflects on CER as a scientific discipline by applying Fensham’s two sets of criteria [2], introducing several mainstream meta-research schemes addressing research topics in CER, research methods and use of theories. Together, chapters “What is Computing Education Research (CER)”, “Theory and Approaches to Computing Education Research” and “The Evolution of Computing Education Research: A Meta-Analytic Perspective” introduce the reader to the foundational characteristics, the grounds of CER.

Chapter “Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research” lays out the methodological approach of this book and introduces the reader to scientometrics as a research methodology. This is followed by a number of chapters that present scientometric findings of various aspects of the field. Chapter “The Hands that Made Computing Education Research: Top Authors, Networks, Collaboration and Newcomers” presents an analysis of author productivity patterns, influential authors, clusters of co-authorship and international collaboration. Chapter “The Venues that Shaped Computing Education Research: Dissemination Under the Lens” presents a scientometric analysis of dissemination practices of CER, revealing top publication outlets, variations in citation rates, and differences in diversity of topics between publication outlets. Chapter “The Evolving Themes of Computing Education Research: Trends, Topic Models, and Emerging Research” focuses on the main topics investigated in CER, showing the major trends of research, such as that on programming education, computational thinking, and K-12 computing education, and analyses how the common research topics are connected with each other. The chapter also brings insights of emerging topics such as machine learning education.

After these scientometric analyses of publication metadata, Chapter “Capturing The Impact and The Chatter around Computing Education Research Beyond Academia in Social Media, Patents, and Blogs” complements the view by turning the focus to social media, news, and blogs. A comprehensive analysis of data shows trendy topics and articles that have sparked public discussion, and attracted attention within the general public. Chapter “A Scientometric Perspective on the Evolution of the SIGCSE Technical Symposium: 1970–2021” continues the scientometric approach by providing an analysis of the publication metadata of the SIGCSE (Special Interest Group in Computer Science Education) Technical Symposium, the oldest and largest venue for presenting CER. The analysis covers research themes, influential authors, and author networks. Chapter “ITiCSE Working Groups as an Engine for Community-Building” continues this trend by focusing on ITiCSE Working Groups, a special form of research collaboration for attendees of ITiCSE (Innovation and Technology in Computer Science Education) conference. This chapter analyses the working group activities and shows how the activity attracts researchers and acts as a pathway for welcoming newcomers into CER.

Chapters “A Case Study: The Uppsala Computing Education Research Group (UpCERG)” and “Future Technology Lab: A Plug-In Campus as an Agent of Change for Computing Education Research in the Global South” provide case-

analyses of two influential CER initiatives: that of the Uppsala Computing Education Research Group (UpCERG), and that of the Future Technology Laboratory (FTLab), a Namibia-Finnish collaborative CER initiative based in Namibia. The UpCERG-group was founded in mid-90's, when two researchers met and agreed to pursue for a better scientific foundation for research of computing education. Chapter "Future Technology Lab: A Plug-In Campus as an Agent of Change for Computing Education Research in the Global South" analyses the trajectory of developing the FTLab-initiative, and gives readers ideas about reforming and contextualising CER in the Global South.

The next series of chapters focus on region-, or country-level analyses. First, chapter "Computing Education Research in Baltic Countries" presents a unique historical retrospective of the development of computing education in the Baltic countries, along with key milestones, achievements, similarities and differences in approaches to CER in Estonia, Latvia and Lithuania. This is followed by analysis of CER in the Global South (chapter "Computing Education Research in the Global South"), in Finland (chapter "Computing Education Research in Finland"), in Australasia (chapter "Computing Education Research in Australasia"), in Israel (chapter "Computer Science Education Research in Israel"), and in the UK and Ireland (chapter "Computing Education Research in the UK & Ireland").

Finally, chapter "Computing Education Research in Schools" provides an analysis of K-12 computing education, which is one of the most researched and rapidly growing domains of CER, including an overview of top categories of research and foundational articles. Chapter "Conceptualizing Approaches to Critical Computing Education: Inquiry, Design, and Reimagination" addresses the critical issues of algorithmic bias, discriminatory practices and techno-solutionism, and discusses potential ways to understand and address them in educational practice.

## 3 Reflections

We acknowledge that building a comprehensive view of an academic field from its formative years to its current state is not an easy endeavor. Moreover, the view should complement existing analyses and descriptions of the field. The most comprehensive one is the Cambridge Handbook of Computing Education Research [5], which focuses on discussing the research methods, theoretical frameworks, and in-depth presentations of the wide variety of subareas in CER. Our current book presents an alternative view of the field. The scientometric data covers a very large pool of literature, much wider than is available, for example, in the ACM digital library. Scientometrics as a method provides tools for building a big picture of the development of the field covering analyses of authors, their countries of origin, collaboration networks, citation patterns and topics addressed in the research. It thus reveals information which is not visible in topical reviews in such depth. Moreover, it allows identifying and analysing the development of regional and topical subcommunities. Several case study chapters in this book

analyze these subcommunities and provide additional perspectives written by people who are experts in those subcommunities. This allows the reader to understand the differences of the communities and factors behind these differences, which helps building a more holistic understanding on the field. Moreover, it allows to identify best practices in the development of subcommunities, information, which can support the whole field. Overall, this book provides a fresh view of the development and current state of the CER as an academic field.

There is much more that could be done. An obvious track of future activity is to update the scientometric data and analyses after a few years to analyze new developments. Another dimension is to complement the case studies with areas which are not covered in this book, such as CER in other countries or regions in Europe, e.g., Germany, Nordic countries, Southern and Eastern Europe, not to speak about CER in other language areas such as CER in South American countries, India or China. CER in North America is now covered only in the chapter addressing the SIGCSE Symposium, but the field is much richer in this region and the analysis would need much further work. This would naturally require many scholars in those areas to join the effort. Another direction would be to augment the data from other publication databases, as Scopus unfortunately does not include comprehensive data from all years of relevant publication venues. The challenge is that the available meta data is less comprehensive and would require manual updates, a very laborious work. A third direction would be collecting data from various subareas in CER, using more targeted search terms to cover the area more comprehensively. In all, this project is a beginning rather than an end, opening up many new tracks for future research, to be presented in future editions of this book and in current and new publication forums.

## References

1. Fee, S.B., Holland-Minkley, A.M., Lombardi, T.E. (eds.): New Directions for Computing Education: Embedding Computing Across Disciplines. Springer (2017)
2. Fensham, P.J.: Defining an identity: The evolution of science education as a field of research, vol. 20. Springer Science & Business Media (2004)
3. Fincher, S., Petre, M.: Computer Science Education Research. Taylor & Francis (2004)
4. Fincher, S.A., Robins, A.V.: An important and timely field. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, pp. 1–8. Cambridge University Press (2019). https://doi.org/10.1017/9781108654555.001
5. Fincher, S.A., Robins, A.V. (eds.): The Cambridge Handbook of Computing Education Research. Cambridge University Press (2019). https://doi.org/10.1017/9781108654555.001
6. Kadijevich, D.M., Angeli, C., Schulte, C. (eds.): Improving Computer Science Education. Routledge (2013)
7. Simon: Emergence of computing education as a research discipline. Ph.D. thesis, Aalto University School of Science (2015)
8. Sue Sentance, E.B., Schulte, C.: Computer science education: perspectives on teaching and learning in school. Bloomsbury (2018)

# What is Computing Education Research (CER)?


**Mats Daniels, Lauri Malmi, Arnold Pears, and Simon**

## 1 Introduction

What is Computing Education Research (CER), who is doing research in the field, and what characterizes different types of CER? We address these questions in this chapter from our perspectives as active researchers in the field for the last almost 30 years. It is not our purpose to act as gatekeepers but rather to provide views on CER and discuss how the research fields and community have evolved. The goal is to provide the reader with a historical perspective and a structure to understand CER, in which we hope the CER community will feel at home. One particular aspect of CER is change, which provides a challenge in framing this chapter. Changes in CER have been considerable, from its early history in computer science to the vast diversity of computing education at all educational levels as well as in non-formal settings of today. This change stems partly from far-reaching changes in the nature and scope of computing education over the past 50 years. Early efforts in computer science education in the middle of the twentieth century were restricted

M. Daniels (✉)
Uppsala University, Uppsala, Sweden
e-mail: mats.daniels@it.uu.se

L. Malmi
Aalto University, Espoo, Finland
e-mail: lauri.malmi@aalto.fi

A. Pears
KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: pears@kth.se

Simon
Wadalba, NSW, Australia
e-mail: pears@kth.se



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_2

to university settings and narrowly defined as related to the teaching of Computer Science. Over the last ten to fifteen years, the field has broadened to include many fields and levels of education. The shift from the term *computer science education research* to *computing education research* also reflects the change of scope.

It seems clear from the development of the computing field, and the consequent evolution in computing education over the past 30 years that research plays an essential role in the development of relevant and innovative delivery of computing education. Researchers must be computing professionals with skills in other relevant disciplines capable of identifying fundamental teaching and learning issues unique to computing education. The researchers should be able to focus on meaningful research on those issues and challenges and interpret and disseminate the results. We posit that the key contribution of computing education researchers is their ability to illuminate and address domain-specific teaching and learning issues in computing education. CER researchers are central to identifying relevant research, either in higher education or in computing itself, and applying this to computing education, developing practical insight and new approaches to teaching and learning tailored to the computing education domain.

Given the rapid growth of Computing as a discipline and the complexity of the research foci aligned with educational transformation, it is clear that a single definition of CER is not possible. However, taking a historical perspective, including the development of a sense of scholarship, allows us to analyze the focus of CER over time. Furthermore, we will provide an environmental structure for CER, that includes the components *computing in general*, *learning and teaching computing*, and *educational research*, to discuss the interaction and overlap between CER and the other aspects of the field of Computing. The concept of scholarship gives a common ground for valuing CER. To that end, we provide a short introduction to scholarship based on a framework developed by Glassick [19] as a basis for the CER community.

Finally, we will reflect on the status of CER as a discipline. In this, we will use some criteria for a discipline, presented originally for Science Education Research by Fensham [15] and provide our assessment of how well CER fulfills these criteria. We argue that CER has matured to be seen as a legitimate research discipline, and conclude by relating CER to other examples of Discipline Based Education Research (DBER).

## 2 The History of CER

As we approach tracing the history of what today is called Computing Education Research (CER), it seems relevant to provide an overview of the organizations and activities that contributed to establishing the field. In particular, the publication venues that developed from the late 1960s to the present and the types of publications and discussions that have been dominant in different phases of development reflect the field's maturation. Other chapters in this book present a more extensive

analysis of the publication trends, topic shifts, and influential research groups and authors.

## 2.1 Dissemination of Results

The Association for Computing Machinery (ACM) Special Interest Group in Computer Science Education (SIGCSE) emerged as an organizational structure within the ACM in the late 1960s, with the first issue of the quarterly newsletter, the SIGCSE Bulletin, appearing in 1969. The first SIGCSE Technical Symposium followed this initiative in November of 1970. There ensued a considerable delay in terms of the internationalization of the SIGCSE conferences and workshops. Scholars outside North America would have to wait until the 1990s and the establishment of the Innovation and Technology in Computer Science Education (ITiCSE) conference series and the Australasian Computing Education Conference (ACE) in 1996, followed by Koli Calling in 2001 and the International Workshop on Computing Education Research (ICER) in 2005. The most recent new SIGCSE-sponsored conference targeting fields outside the US, Europe, and Austral-Asia is CompEd, first organized in 2019. In recent 10 years, also several other new CER-focused conferences have been launched in Europe, such as UKICER and CSERC, as well as WiPCSE which focuses on K-12 computing education.

On the other hand, while the conferences mentioned above have focused solely on CER, they are certainly not the only venues in which CER papers are published. There is a long history of research on professional programming, beginning, for instance, with Weinberg's studies on the psychology of programming [71] and Soloway's empirical studies of programming, e.g. [63], in the 1970s and 1980s. This work has also addressed learning programming, for example, by comparing experts' and novices' conceptions of programming and working patterns. Psychology of programming interest group (PPIG) has organized relevant workshops since 1986, where CER papers are published. Moreover, engineering education conferences, such as Frontiers in Education (FIE), founded in 1971, frequently publish computing education research at the tertiary and school levels. There are several relevant journals that have emerged over time in addition to the conferences. Some of the more influential in the development of the field are the Taylor and Francis publication "Computer Science Education", ACM Transactions on Computing Education (TOCE), and the IEEE Transactions on Education. In addition, numerous educational journals and engineering education journals have accepted CER papers for a long time.

While these venues have mainly focused on adult learning of computing, it is essential to recall early studies on children and programming. Examples are the LOGO language for teaching programming to children developed around 1970 and Papert's seminal work "Mindstorms" in 1980 [46, 62].

In this chapter, we discuss CER from the perspective of its main publishing venues and thematic research foci. For instance, in another chapter in this book, we cover computing in schools.

## 2.2 Views on CER

In the early years, there was little if any discussion regarding CER and the nature of the field. For instance, general discussions of CER's status as a discipline emerged after the turn of the century. Early studies and research relevant to computing education were not considered a separate research field, CER. In many cases, they were reports that described teaching practice and learning approaches broadly assessed as successful. The discussion of theory, structure, and research expectations first emerged in panel discussions on what CER should be in the early 2000s, followed by the release of two seminal handbooks on CER.

### 2.2.1 Early Panel Debates

During the early 2000s, there was considerable debate around the nature of computing education research, its structure, and its rigor. An example from 2002 is the description of Computing Education Research (CER) and the models and methods associated with such research [48]. Early discussions of the nature of what has become CER can also be found in the ACM community proceedings of the 2004 SIGCSE symposium and ITiCSE and ACE conferences somewhat later [4, 11, 20, 47].

Two panel debates at ITiCSE 2004 presented perspectives on research. Ben-Ari et al. [4] argued that theoretically sound research enabled generalization of results beyond a single context and allowed comparison and contrasting results from comparable studies in multiple settings. They exemplified with the swathe of multi-national and multi-institutional studies produced from the Bootstrapping project [18, 52]. They also traced the discourse on research methods and rigor back to their work in 1998 [12].

Goldweber et al. [21], on the other hand, presented four perspectives on CS Education research as a field and what it meant to do good research. Martyn Clarke argued that rigorous research could only be conducted with acceptable quality using research methods and interpretation as a collaborative interdisciplinary exercise between educational science and computing. Sally Fincher advanced the view that the field was concerned with noticing phenomena in the computing education context and, through research, providing insight and explanations for both the phenomenon and its impact on learning. Michael Goldweber described the field as about exchanging ideas and innovations among professionals. Arnold Pears summed up the panel by arguing that all of these approaches had merit and that the vital aspect of working in the field was to be honest about the nature of the

contribution in relation to the three paradigms presented by the other speakers. Each type of contribution has value to a particular audience. However, they are focused on different aspects of the domain and vary in their level of relevance to other educators.

The main argument was that computing domain expertise was vital to success and that it is essential to devote time to establishing a common understanding of the role of the SIGCSE and ITiCSE communities in defining the scope and focus of CER.

### 2.2.2 Handbooks on CER

The first book in the field of CER, by Fincher and Petre [16], emerged at about the same time as the above-mentioned panel debates. While they did not give a comprehensive definition of the field, they described the scope and diversity of the work by dividing the field into ten subfields of research. These included *student understanding* which explores students' mental and conceptual models, their perceptions, and misconceptions; *animation, visualization, and simulation* focuses on building and researching various software tools to support teaching and learning various topics in computer science; *teaching methods*, a broad field that covers aspects such as scaffolding learning, supporting interaction, collaboration, and teamwork among students, and different aspects of project work; *assessment* that covers assessment methods, including validity, as well as automated assessment and feedback, and plagiarism detection; *educational technology* focuses on how new learning and teaching platforms and technologies can be used and integrated into computing education, as well as development and impact of tailored learning environments for computing, such as specialized programming environments; *transferring professional practice into the classroom* aims at understanding professional work practices and how these could be applied in computing education; *incorporating new developments and new technologies* explores how recent new developments and technologies in computing can be incorporated into classroom and courses; *transferring from campus-based teaching to distance education* investigates how educational content and methods can be transferred into online settings; *recruitment and retention* is a subfield that explores ways to attract and retain students into computing, including broadening participation by attracting underrepresented minorities and studying equity and diversity issues in computing education contexts; and finally *construction of the discipline*, a subfield that addresses the curriculum development, building academic qualifications, and explores the nature of discipline itself.

Fifteen years later, the Cambridge Handbook of Computing Education Research [17], currently the most comprehensive source for work in CER, splits work in a new way reflecting the development of the field. The first field is *systemic issues*, topics that persist in the field. These include research on introductory and more advanced programming, various pedagogical approaches, assessment and plagiarism research, and questions addressing equity and diversity. The second broad field, labeled as *new milieux*, addresses more recent issues which have arisen

when computing education has spread beyond the "traditional" university settings in formal classrooms and departments of computing. The *new milleux* covers research on computational thinking and computing in schools (K-12), computing for other disciplines, and new programming paradigms. A third field is *systems software and technology*, where research focuses on how software and hardware tools, tangible computing, and integrated learning environments can support learning. Finally, *teacher and student knowledge* is an field investigating issues concerning the production and acquisition of computing knowledge, for example, teacher knowledge and teacher training, professional development, learning outside classrooms, student knowledge and misconceptions, students' motivation, attitudes, and dispositions, as well as students as teachers and communicators.

While the CER handbook covers the field widely, there are some additional fields that are not minor and thus worth adding. *Broadening participation* is closely related to equity and diversity, but it also covers work that seeks to find ways to attract new people, especially children, to consider computing as their future choice of study. *Games and gamification* have been widely used in computing education to support students' motivation and encourage studying. *Construction of the discipline* addresses curriculum development, building academic qualifications, and exploring the nature of the CER itself. Finally, *learning analytics and educational data mining* are increasingly used to investigate various aspects of students' behavior while studying by exploring data sets collected with software (e.g., [22, 25]).

The above seminal books split the field into subfields based on the topic of research. Other approaches describing the field have considered the nature or type of work that is carried out, as well as the impact of the work.

## 2.3 Classification of CER Papers

In 2004, David Valentine published his paper presenting a meta-analysis of 20 years of CS1 papers in SIGCSE proceedings [69]. Valentine produced what can be considered the first attempt to provide a taxonomy of work in CER. He divided the contributions published to date in the SIGCSE proceedings into the following categories: *Experimental* analysis with a focus on positivist experimentation, *Marco Polo* "I went there and saw this" experience reports, *Philosophy* debating a general issue in the field, *Tools* technology in education, *Nifty* cool tips and tricks, and *John Henry* extreme experimentation.

Pears et al. [49] sought to identify the core literature for the field that all CER researchers should be aware of. While this may have been a feasible goal in 2005, the growth and development of the field challenge whether such a goal is relevant anymore. However, in many subfields, it is possible to identify *influential* and *seminal* papers which have guided future work by opening new avenues for research and/or having a clear impact on future work. During the last 10 years, the field has also seen a significant number of *synthesis papers* that summarize and classify

substantial work. Most of these are reviews, often systematic reviews, while there are also papers defining taxonomies seeking to categorize work in an field.

Simon [58, 60] developed a different categorization system with four dimensions. He separated the research target into two dimensions. *Context* described the curricular context where the research was carried out (if any), such as programming, software engineering, databases, data structures and algorithms, information systems, or networks. *Theme/Topic*, on the other hand, described what was actually investigated, such as teaching/learning techniques or tools, assessment, students' ability/aptitude, distance/online delivery, ethics/professional issues, gender issues and diversity, recruitment, etc. *Scope* described how wide the community addressed was, ranging from a single course to multi-institutional studies. Finally, the *nature* dimension classified the papers on whether they were collecting and analyzing empirical data, or just reporting on experiences or stating a position.

### *2.4 Frameworks for CER*

Pears proposed that a CER framework was a fundamental first step towards the goal of helping the field to interpret and structure its future. Such a framework would be valuable in the computing education community and in the wider interdisciplinary context of Discipline Based Education Research (DBER) [61], of which CER is a part. This work was further developed through efforts to define the core literature [49] and to analyze and try to establish criteria through which to understand the impact of publications on shaping this emerging field. The fundamental argument behind the core literature effort was establishing an understanding of CER work. Such an understanding was widely agreed to be a vital background for younger colleagues and Ph.D. students in their research. It would help to define what the field was about, and what important publications had shaped the discourse and conduct of research. These research frameworks and a discussion of theory's role in CER are covered in detail in chapter "Theory and Approaches to Computing Education Research".

## 3 CER Today

Today, CER is an increasingly diverse field. Using Simon's classification scheme [58, 60] as a framework, CER addresses questions relevant in a wide selection of curricular contexts within ACM curriculum recommendations [2] in addition to informal learning of computing. However, the focal fields of research are very much skewed. Introductory programming education is a persistent subfield in CER, which receives a large share of attention. This skew is visible, for example, in the systematic literature review by Luxton-Reilly et al. [30], which identified 1666 papers focusing on this field from 2003 to 2017. Moreover, Robins' review

of research on novice programmers [56] in the Cambridge Handbook is the most extensive chapter in the book. In general, many popular topical fields in CER are easy to map to typical first and second-year courses in computing programs. However, in recent years, a fast-growing number of research papers have emerged focusing on K-12 computing education, mostly concerning challenges in learning programming from pre-school to high school levels. CER in schools and the general topical fields in CER are discussed in more depth in two other chapters in this book.

The second dimension, using Simon's classification, is the research theme, i.e., what the research is about in the specific context. This theme includes numerous aspects, such as ability, aptitude, assessment techniques or tools, cheating and plagiarism, curriculum, distance/online delivery, educational technology, ethics/professional issues, gender issues, recruitment, teaching/learning techniques or tools etc. (for a more comprehensive list see appendix in [56]). All of these aspects are addressed in CER.

Thirdly, CER covers studies with very different scopes. A large share of studies focuses naturally on course-level research contexts or even task-level contexts. However, on the other end, highly important international studies have collected and analyzed data from multiple institutes internationally, e.g., [29, 40, 68].

In recent years, CER has increasingly paid attention to research rigor, which essentially concerns applying rigorous data collection and analysis methods and using theoretical frameworks to guide research designs and interpretation of results. In addition, improving the quality of reporting has been addressed in many ways, for example, with explicit definitions of research questions in papers, more accurate reporting of applied methods and theories, and stating limitations more clearly. This increase in rigor is also visible in more specific instructions for authors on conference and journal websites, more explicit review criteria, and improved rigor in review processes [53] as well as in methodological reviews of the CER literature, e.g., [27, 31, 38, 41, 55].

Parallel with this, there has emerged a growing interest in the analysis and discussion of the use of theories to inform CER. Multiple reviews focusing on theories in CER have appeared in recent years [27, 32, 34, 64, 65]. Computer Science Education published a special issue, "Advancing Theory about the Novice Programmer", in 2019 with several papers addressing this topic. ACM Transactions on Computing Education publishes two special issues on "Conceptualizing and Using Theory in Computing Education Research" in 2022–2023 [67]. Moreover, there is an increasing interest in developing domain-specific theories and models in CER that emerge from the field to explain the complex phenomena related to teaching and learning computing [33, 35]. Examples are statistical models, such as structural equation models or regression models, qualitative grounded theories, or phenomenographical outcome spaces. Though, more elaborated theories have emerged, such as a theory of instructing programming skills by Xie et al. [75] or models for predicting student success based on their programming behavior and social aspects [7, 8] by Carter et al. There is also discussion on the role of theories in CER, whether they are always needed and if they even restrict designing new learning innovations [44].

All this demonstrates that the field is maturing as a research field. Rigorous research papers have always been published in the field. However, experience reports or practice papers dominated most CER-focused venues in the first decades. During the last 20 years, the share of research papers among publications has been steadily increasing [59], and a growing awareness and emphasis have emerged on research quality. This book discusses these developments in more detail in another chapter focusing on meta-studies in the field.

## 4 An Environment for CER

To complement the above, in this section, we present a more generic structure that places CER in an environment by relating CER to aspects that are informing and/or are being informed by CER. The environment in which CER exists has the following components:

- Computing in general;
- Education research in general;
- Learning and teaching computing.

This structure has *Computing in general* as an essential object that informs CER. *Education research*, which includes any relevant research approach, provides CER with many relevant methods and theories, not least related to learning and collaboration. *Learning and teaching computing* is where CER "results" are received and also a study object for CER. Learning and teaching computing is not restricted to higher education. There is, for instance, a large body of work on non-formal learning computing and computational thinking. Discussing the boundaries and overlap between these elements in the proposed structure will provide a way to understand CER.

### *4.1 CER and Computing in General*

Computing is essential to CER as it is the object to be understood in educational contexts. A confounding factor is that computing is a vast field. ACM curricula 2020 identifies seven subfields: computer engineering, computer science, software engineering, information systems, information technology, cybersecurity, and data science [2]. Research that could be included as CER can be carried out in any of these subdomains. Another aspect of computing is that it is used as a component in other, often quite complex, contexts. There is, thus, a need to understand how computing artifacts interact and influence various settings. This blurs the lines between what is computing and what is not. In CER, a wider definition of computing which includes understanding its role in complex contexts, is appropriate.

In the past, CER focused much on teaching and learning computer science topics, especially programming [6] and other basic undergraduate studies, such as data structures, algorithms, and databases. Research on advanced computer science topics has been rare. A fairly obvious reason is that the main challenges in computing courses focus on the basic courses, where many students struggle and drop out. These courses are also often large in the number of enrolled students (ranging to hundreds or even thousands). In contrast, advanced courses have a small or moderate size, which enables more personal level support for students.

However, as computing has widened as a domain field, also CER has diversified. When computing subdisciplines have established their publication venues, it is natural that CER papers addressing those subdisciplines are published in these conferences and workshops. An example is software engineering education research published in the educational track of ICSE (International Conference on Software Engineering). Similarly, programming education research papers in engineering education contexts are often published in engineering education venues. This diversity forms a challenge in defining the field because the label CER is often used in a narrow perspective which focuses on "CER only" publication venues.

If we contrast CER and computing research, we can identify similarities and major differences worth noting. CER can be characterized as a social science that heavily deals with human participants. As such, it applies many similar methods and theoretical frameworks used in HCI research, usability studies, and empirical software engineering. On the other hand, a sizable subfield of CER is research related to developing advanced software for education, which is close to software engineering research. Some of such work is plain engineering, i.e., designing and implementing new software. At the same time, more specific technical research exists, where novel technological solutions are developed, e.g., for supporting the integration of interactive learning content [5].

Theories have quite different interpretations in CER and computing sciences. In CER, theories focus on describing, explaining and/or predicting human behavior, thus working in the domain of social sciences. On the other hand, theoretical computer science, algorithms research and cryptography strongly focus on theorems and general statements based on the mathematical research tradition. Data science, machine learning, and data mining draw much from Statistics research, thus being also close to mathematics research tradition. In addition, in technical fields of computing, for instance, software technology, computer engineering, and cybersecurity, mathematical and engineering traditions are strong, too.

## *4.2 CER and Education Research in General*

Education together with computing are the "parents" of CER. Education research generally addresses such aspects of teaching and learning which are not tied to some specific discipline. In contrast, Disciplinary Education Research, such as CER, focuses on education-related topics in its specific disciplines. In CER, it can, for

instance, be researching learning obstacles or fruitful ways to motivate learners on a particular topic in computing. Other indirect examples are studying the influence of culture and understanding diversity issues in computing education. The aspects of computing can vary, but computing has to be a component for research to be CER, and CER researchers need to understand the computing aspect.

CER borrows and applies much work from Education, Psychology, and other social sciences like Sociology. Numerous theoretical frameworks have been adopted from these social sciences to support the investigation of various phenomena in computing education [32, 65]. Moreover, CER applies many research methods, e.g., qualitative methods such as grounded theory, phenomenography, phenomenology, and various types of content analysis, that have been used in the social sciences for a long time.

The question can be raised, why is CER not just a subfield of Education? There are several arguments against this claim. First, CER addresses computing-specific concepts, processes, and phenomena, and their investigation requires a deep understanding of these topics. Without a proper understanding of concepts such as objects, classes, recursion, and notional machines, it is impossible to study how students understand the topics. On the other hand, there is certainly work in CER, where this requirement is less important, such as investigating students' team working in capstone projects or computing students' experiences in company internships. However, it is natural that such research focusing on computing education is published in related venues.

CER also increasingly develops its theoretical frameworks in terms of various statistical models of data collected from computing education settings or qualitative analyses of such data [33, 34]. Much work is devoted to building new validated instruments or concept inventories that address learning computing topics or adapting existing more general instruments from Education or Psychology [34] into computing contexts.

Thus, CER is gradually building its own set of theoretical tools and methods strictly tied to knowledge and skills in computing, which general theories and instruments cannot reach.

## *4.3 CER and Learning and Teaching Computing*

Learning and teaching computing is in some way or another the ultimate target of CER as it is its field of practice. This field of practice is even more complex than the computing concept. Part of the complexity stems from the complexity of computing itself, but further complexity arises from aspects such as the influence of educational contexts, formal vs. non-formal education, specialized vs. integrated learning objectives, and higher vs. K-12 education. The boundary, or rather overlap, between CER and its field of practice is unclear. For instance, many CER researchers are expected to provide concrete help to computing teachers.

### 4.3.1 Research vs Development

There is sometimes a distinction between development and research made, especially concerning funding opportunities. This line is problematic in many cases since many CER projects draw on action research where development is part of the research. Furthermore, the difference between computing education research and development of computing education is subtle since computing teachers carry out development in more or less scholarly ways. Scholarship is a concept that addresses the identity of teachers and provides a common ground to understand how development and research are related and complementary with regard to influencing teachers.

Glassick's definition of scholarship provides a basis to calibrate our view of CER and the maturity of the field [19]. He discusses the standards to which responsible scholarship in a discipline might be held and establishes a framework for scholarly quality in six fields. In his view, high-value scholarship should excel in all six fields.

- Clear Goals—Does the scholar clearly state the basic purposes of his or her work? Does the scholar define objectives that are realistic and achievable?
- Adequate Preparation—Does the scholar show an understanding of existing scholarship in the field? Does the scholar bring the necessary skills to their work?
- Appropriate Methods—Does the scholar use methods appropriate to the goals? Does the scholar effectively apply the methods selected?
- Significant Results—Does the scholar's work add consequentially to the field? Does the scholar's work open additional fields for further exploration?
- Effective Presentation—Does the scholar use appropriate forums for communicating work to its intended audiences? Does the scholar present her message with clarity and integrity?
- Reflective Critique—Does the scholar critically evaluate his or her work? Does the scholar use evaluation to improve the quality of future work?

This framework provides a basis to reflect upon CER publications and the nature of the field's strategic research discourse.

### 4.3.2 Research Publication or Not?

There is a long tradition of publishing papers that present novel pedagogical innovations, experiences of applying some existing pedagogical approaches and methods, novel learning resources, and software tools to support education. Many of these include not only a description of these but also some form of evaluation, e.g., student feedback, course or task results, or teacher's reflections. There may be a pre/post-test analysis of learning gains in a group or a comparison with previous years' cohort results. There is no consensus in the field which publications should be considered *research papers* that would be counted as CER contributions, and which

are *practice reports, experience reports, Marco Polo papers* or some other names to identify them as something different.

This split causes tensions within the community, as it suggests that some published papers may be of poor quality. Yet, they may be highly valuable for teachers looking for new ideas and resources for their teaching. However, there is a clear trend in the field that the share of research papers is increasing in the main publishing venues. See, for example, [59] and chapter "The Evolution of Computing Education Research: A Meta-Analytic Perspective" in this book. This change is also reflected in the instructions for authors in journals and conference calls for papers. The expectations for publications have become more specific, typically emphasizing the use of theoretical frameworks to support arguments and analyses, as well as applying rigorous empirical methodologies and styles of reporting results. Many conferences have developed their review processes to better address such requirements while there is also debate on what is the main role and purpose of conferences—presenting research publications, or meeting people, and sharing ideas and innovations [53] as well as what is the role of theory in CER [44]. A part of this debate follows from the practice in computing sciences, where conference papers are considered valuable scientific merit, sometimes more valuable than journal papers. In almost all other disciplines,research contributions are mainly published in journals, and conferences are venues for presenting novel work and meeting people.

There is no sign that this tension would be vanishing, and yet both types of contributions are needed to make progress in improving computing education.

### 4.3.3 CER and Educational Settings

The environment in which learning and/or teaching computing occurs has evolved from being done exclusively at higher education institutions in the early days. Today, aspects of computing are integrated in the school system and are part of most disciplines and degree programs at the higher education level. There is also substantial work on understanding educational settings other than the formal settings, for instance, coding clubs and lifelong learning.

*Learning Outside Formal Settings* While much research has focused on the school environment and formal education, there is also considerable work done in the informal arena. In the last decade, there has been considerable research effort placed into the impact of societal initiatives, e.g. the MicroBit and Arduino systems, learning in informal settings, the Maker movement, Code Dojos, Bebras, Hour of Code, CS for All, and CS Unplugged to name just a few. Informal learning in computing has also been the focus of considerable research, including the EU-funded ComNPlay Science project. Publication of non-formal, informal, and school-level CER has often been in different venues than those utilized by tertiary CER researchers. Venues of interest to these communities include a range of school-related conferences in Europe, Asia, and the Americas, often published in languages

other than English. These activities and related research have increasingly found their way into the last decade's traditional ACM and IEEE publication venues.

*Learning in Integrated Settings* Early work by Papert in 1980 on Computational Thinking (CT) was reawakened by Jeanette Wing in 2006 [74], who emphasized the breadth of contexts where CT is needed. The CT trend has spawned a plethora of works discussing the relevance and definition of CT [13, 51] and what that might entail [37, 45, 50]. CT is also linked to informatics [9] and other European disciplinary fields related to computation and computing machinery, as well as the teaching of these concepts in the various levels of compulsory schooling [10, 36].

CT is also connected to future computing and a broader computing milieu, embracing concepts like virtual and augmented reality, artificial intelligence, and biotechnologies. This increase in technology adoption means that future citizens should be familiar with the above mentioned concepts to stay competitive in the job market.

In addressing these needs, educational institutions have an important role in preparing future entrepreneurs through appropriate curricula and methods that appeal to today's learners. The 2021 World Economic Forum report, a Swedish National Report on Digital Cutting Edge Competence [24, 73], and an influential study by Tedre et al. [66] show that there is a need for a nuanced understanding of CT, calling for extension and contextualization of CT to include explicitly Machine Learning (ML) and AI (CT 2.0). Munasinghe [43] has recently shown that many concepts related to CT should be further elucidated to facilitate their accessibility to teachers. Thus, computational thinking has been actively promoted in schools in an integrative approach, helping to enhance our definition of STEAM (Science, Technology, Engineering, the Arts, and Mathematics) education. STEAM education provides several benefits, such as enhancing the skills of analysis and problem-solving and creativity enhancement [3, 70].

However, there are still some challenges and issues in STEAM-related activities integration into school and within STEAM subjects. Researchers and educators are consequently re-examining the importance of STEAM-related activities and programs, specifically developing computational thinking skills and integrating maker education where learners imagine, design, and create projects that combine learning content with practical hands-on applications. Our approach to STEAM also emphasizes collaboration and integration. These frameworks draw on the work of Yang et al. [76], particularly in terms of how CT could be positioned in the curriculum by designing and implementing an integrated STEM and CT lesson. Yang emphasizes the role of a problem-based process for integrating CT problems and solutions into after-school programs using hands-on inquiry activities which are exciting and engaging for students. Moreover, programming and using physical computing objects—robots enable students to engage in scientific practices and, in such a way, learn some engineering aspects (such as bridge design) and gain satisfying experiences.

The design process takes place through *design thinking* (DT), which realizes learning through experience and complex problem solving for motivation, openness

to new ideas, and creative thinking in the learning process [57]. Thus, in the literature, the need to adapt the design to learning is often emphasized by scholars and practitioners [26]. Design thinking is seen in this context as a learning design that facilitates a constructive way of learning due to practice-based development of a range of target skills [57]. Learners receive several benefits from integrating STEAM learning with computing by modeling various phenomena, such as computational thinking learning [23]. However, researchers and educators still face the challenge of defining computational thinking and getting a theoretical grounding for what form it should take in school. One of the possible solutions proposed by Weintrop and colleagues is to develop CT taxonomy [72]. In such a way, CT can be embedded in the STEAM subjects' context. CT development is then made available through STEAM-related activities integration in school, especially through hands-on projects [39].

## 5 Is CER a Discipline?

Major academic disciplines, such as history, linguistics, mathematics, and computing, comprise further lower-level disciplines. It is easy to think that each lower-level discipline emerged and was accepted. In computing, for example, few would doubt that cybersecurity and data science are legitimate research disciplines, although they have emerged recently. Why is there a need to question whether computing education research is a research discipline?

One reason is that many academics lead what Lister [28] has called a 'double life': they educate students, conduct research, and keep these two activities well apart. It simply does not occur to these academics that work regarding one's teaching, and one's students' learning might be researching.

This 'double life' need not matter, except that it affects such matters as research funding, research-teaching load balance, and promotion prospects. These reasons are why computing education researchers feel the need to persuade other computing researchers that what they are doing is indeed research. One way to do this is to establish that computing education is a legitimate research discipline.

Computing is not the only field whose educational researchers have felt this need. In 2004, Fensham [15] set out the criteria by which a DBER might be assessed to determine whether it is a research discipline. We very briefly review these criteria here.

*Structural criteria* focus on organizational things.

- There are many examples of academic recognition regarding professorships in computing education research or very close titles for the target field.
- Two prestigious research journals, ACM Transactions on Computing Education and Computer Science Education, are focused on the field, and numerous journals in closely related fields regularly publish CER papers.
- There are professional associations in the field, such as SIGCSE.

- Several research conferences, such as ICER and Koli Calling, focus solely on CER.
- There are research centers focusing on CER, for example the Center for Computing Education Research at the IT-University of Copenhagen [1]. There are also numerous longtime research groups globally that focus all or most of their work on CER.
- There are regular research training activities, especially doctoral consortia associated with major conferences in the field.

*Intra-research criteria* focus on how the actual research is carried out.

- There is a substantial amount of scientific knowledge in the field, evidenced by the huge number of papers in the data pool. Applying this information for designing and implementing new research requires substantial expertise in the field.
- CER asks questions that can be answered only in the CER domain, e.g., how students learn programming and what misconceptions they might have.
- CER carries out its own conceptual and theoretical development, as demonstrated, e.g., in [33, 34].
- The field develops its own research methodologies, for example, different types of log data collection and analysis of programming process data, creates its own concept analysis instruments [54], and adapts methods and instruments from other disciplines, e.g. [14, 42].
- The field has much work which builds on previous work, thus demonstrating progression.
- CER has many highly cited model publications guiding future research and seminal publications opening new insights, as mentioned above in [49].

In his doctoral thesis [59], Simon examined Fensham's criteria in detail, and his findings are summarized briefly in Sect. 2 of chapter "The Evolution of Computing Education Research: A Meta-Analytic Perspective" of this book. In short, the conclusion is that according to Fensham's criteria, CER is indeed a legitimate research discipline.

## 6 CER and Other DBER Disciplines

Computing degree programs often include studies of other disciplines that support learning computing, e.g., studies in some fields of mathematics and logic. Moreover, computing students often study other fields as their minor or voluntary studies, e.g., physics, economics, languages, etc. Therefore, it is natural that teaching and learning some topics in these fields can also be addressed in CER contexts when the target group is computing students. Correspondingly, many students from other fields take computing, especially programming courses, and include them in their degrees. Depending on the institutions, such courses can be given by computing

teachers in a department of computer science (or similar), as service courses or minor studies, or the courses can be organized independently by their departments, e.g., in engineering schools or departments.

This overlap possibly occurs most frequently in engineering studies. Therefore, there is a clear overlap between computing education research and Engineering Education Research (EER). CER papers are frequently published in EER conferences and journals, which is natural.

It is difficult to state clear differences between CER, EER, and other DBER disciplines except in the main domain of investigation. Similar research methods and the same theoretical frameworks from social sciences can be used in all of them.

## 7 Conclusions

CER is an inherently interdisciplinary research field combining research findings, theories and research methods from several other fields in addition to developing its own ones. Therefore, the question of what CER is, where it belongs, and even if it is an independent research field is complex. There is no definite answer to this question but rather a set of complementary and partly contradictory answers. The answers are derived from different agendas regarding creating legitimacy for the field, not least economical in relation to funding bodies. This is an interesting question that is interesting to study further in future work.

From a high-level perspective, CER is one research field in a family of Discipline Based Education Research (DBER) fields. The DBERs are based on education research and informed by their respective discipline. They also rely on theories and methods from relevant other disciplines, such as learning sciences, psychology, and sociology. However, the nature and scope of CER are not so well defined when looking more closely at the field. As discussed above in Sect. 2, the panel discussions in ITiCSE 2004 already revealed many different perspectives on defining the field.

In their seminal book, Fincher and Petre [16] discussed the nature of work in the field using a derivation of Pasteur's quadrant as their framework. They split the publication space into four fields with two axes, one addressing how much the paper was based on evidence, especially empirical evidence, and the other one, how much it was based on argumentation or theory. Thus, papers with high argumentation/theory and low evidence were considered *perspective pieces*, while papers with strong evidence but low in theory were considered *practice papers*. CER papers should focus on the quadrant with strong evidence and strong argumentation/theory. Followed by this, they presented the ten subfields of CER, as described in Sect. 2.2.2. Moreover, in the first part of their book, they discussed how to design and carry out research that would match the requirements in the CER quadrant. Their definition thus focuses on describing two relevant aspects of the field: quality of research and what are the research topics in the field.

The Cambridge Handbook of CER [17] provides a similar, though more descriptive than definitive answer to the question of what CER is. In chapter "What is

Computing Education Research (CER)" of the handbook, "Computing Education Research Today", past and present Editors-in Chief of two prominent journals, Computer Science Education and ACM Transactions on Computing Education, discuss the state of the field. Almost the whole discussion focuses on quality aspects of submitted papers, also considering the historical development of the field. The scope of the field—what is included in CER—is not discussed, while the notion that some submissions are "out of scope" is mentioned. Followed by this chapter, the Handbook elaborates on in-depth research designs, methods, and theoretical frameworks, i.e., research quality aspects, followed by presenting a wide coverage of research topics in the field.

One might even argue whether the goal of defining the field is meaningful at all, as it evolves steadily. Furthermore, the domain field Computing is not a static field but evolves and widens rapidly; thus, Computing Education and CER necessarily follow this development. Pragmatic approaches (such as Fincher and Petre's book and the Cambridge Handbook) which focus on the quality of research work and describe the most relevant current research topics, are actually quite suitable for capturing a "snapshot" of CER. That said, we do believe there is a place for future work regarding classification and definitions of the field, not least as it can provide support for future CER researchers.

One example of how the pragmatic approach can be condensed into a few sentences is how Uppsala Computing Education Research Group [UpCERG, see chapter "A Case Study: The Uppsala Computing Education Research Group (UpCERG)" in this book for more on this group] in the field presents its field of work.

*CER addresses learning and teaching in the computing discipline. It is founded on an understanding of the discipline using theories and methods from education research and other relevant disciplines, such as psychology and sociology. Typical fields of study (related to the computing discipline) are learning and teaching core concepts and skills, curricula development, intercultural and interdisciplinary collaboration, identities, and inclusion. The educational context has a focus on higher education, but K-12 and lifelong learning, as well as both formal and informal learning, are also addressed.*

## References

1. Center for Computing Education Research at the IT-University of Copenhagen. https://ccer.itu.dk/. Accessed: 2022-09-14
2. ACM/IEEE Taskforce: ACM curricula recommendations. https://www.acm.org/education/curricula-recommendations. Accessed: 2022-09-14
3. Aithal, P.S., Aithal, S.: Innovation in B.Tech. Curriculum as B.Tech. (Hons) by integrating STEAM, ESEP & IPR features. International Journal of Case Studies in Business, IT, and Education (2019). DOI https://doi.org/10.2139/ssrn.3406824. URL https://papers.ssrn.com/abstract=3406824
4. Ben-Ari, M., Berglund, A., Booth, S., Holmboe, C.: What Do We Mean by Theoretically Sound Research in Computer Science Education? In: Proceedings of the 9th Annual SIGCSE

Conference on Innovation and Technology in Computer Science Education, ITiCSE '04, pp. 230–231. Association for Computing Machinery, New York, NY, USA (2004). URL https://doi.org/10.1145/1007996.1008059. Event-place: Leeds, United Kingdom
5. Brusilovsky, P., Edwards, S., Kumar, A., Malmi, L., Benotti, L., Buck, D., Ihantola, P., Prince, R., Sirkiä, T., Sosnovsky, S., et al.: Increasing adoption of smart learning content for computer science education. In: Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference, pp. 31–57 (2014)
6. Carbone, A., de Raadt, M., Lister, R., Hamilton, M., Sheard, J.: Classifying computing education papers: process and results. In: Proceedings of the Fourth International Workshop on Computing Education Research, ICER, pp. 161–172 (2008)
7. Carter, A.S., Hundhausen, C.D., Adesope, O.: The normalized programming state model: predicting student performance in computing courses based on programming behavior. In: 11th International Computing Education Research Conference, ICER 2015, pp. 141–150 (2015). URL http://doi.acm.org/10.1145/2787622.2787710
8. Carter, A.S., Hundhausen, C.D., Adesope, O.: Blending measures of programming and social behavior into predictive models of student achievement in early computing courses. ACM Transactions on Computing Education (TOCE) **17**(3), 12 (2017)
9. Dagiene, V., Stupuriene, G.: Informatics Concepts and Computational Thinking in K-12 Education: A Lithuanian Perspective. Journal of Information Processing **24**(4), 732–739 (2016)
10. Dagienė Valentina, Jevsikova Tatjana, Stupurienė Gabrielė, Juškevičienė Anita: Teaching computational thinking in primary schools: Worldwide trends and teachers' attitudes. Computer Science and Information Systems **19**(1), 1–24 (2022).
11. Daniels, M., Pears, A.: Models and methods for computing education research. Australian Computer Science Communications **34**(2), 95–102 (2012)
12. Daniels, M., Petre, M., Berglund, A.: Building a Rigorous Research Agenda into Changes to Teaching. In: Proceedings of Third Australasian Computer Science Education Conference ACE (1998)
13. Denning, P.J., Tedre, M.: Computational Thinking: A Disciplinary Perspective. Informatics in Education (2021). DOI https://doi.org/10.15388/infedu.2021.21. URL https://infedu.vu.lt/journal/INFEDU/article/701. Publisher: Vilnius University Institute of Data Science and Digital Technologies
14. Dorn, B., Elliott Tew, A.: Empirical validation and application of the computing attitudes survey. Computer Science Education **25**(1), 1–36 (2015)
15. Fensham, P.J.: Defining an Identity: The Evolution of Science Education as a Field of Research. Springer Science & Business Media (2004)
16. Fincher, S., Petre, M.: Computer Science Education Research. Routledge Falmer (2004). URL http://www.cs.kent.ac.uk/pubs/2004/1819
17. Fincher, S.A., Robins, A.V.: The Cambridge handbook of computing education research. Cambridge University Press (2019)
18. Fincher, S., Tenenberg, J.: Using Theory to Inform Capacity-Building: Bootstrapping Communities of Practice in Computer Science Education Research. Journal of Engineering Education **95**(4), 265–277 (2006). DOI https://doi.org/10.1002/j.2168-9830.2006.tb00902.x
19. Glassick, C.E., Huber, M.T., Maeroff, G.I., Boyer, E.L.: Scholarship assessed: evaluation of the professoriate. Jossey-Bass, San Francisco (1997)
20. Goldweber, M., Clark, M., Fincher, S., Pears, A.: The relationship between CS education research and the SIGCSE community. In: ITiCSE '04: Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, pp. 228–229. ACM Press, Leeds, United Kingdom (2004). DOI http://doi.acm.org/10.1145/1007996.1008057
21. Goldweber, M., Clark, M., Fincher, S., Pears, A.: The Relationship between CS Education Research and the SIGCSE Community. In: Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE '04, pp. 228–229. Association for Computing Machinery, New York, NY, USA (2004). URL https://doi.org/10.1145/1007996.1008057. Event-place: Leeds, United Kingdom

22. Grover, S., Korhonen, A.: Unlocking the potential of learning analytics in computing education. ACM Transactions on Computing Education (TOCE) **17**(3), 1–4 (2017)
23. Grover, S., Fisler, K., Lee, I., Yadav, A.: Integrating Computing and Computational Thinking into K-12 STEM Learning. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, pp. 481–482. Association for Computing Machinery, New York, NY, USA (2020). URL https://doi.org/10.1145/3328778.3366970
24. Gulliksen, J., Cajander, Å., Pears, A., Wiggberg, M.: Digital spetskompetens – den nya renässansmänniskan: Genomlysning, definition, prognosverktyg och rekommendationer för framtida utveckling. ISBN: 978-91-88961-58-7. Tillväxtverket (2020). Publication Title: https://digitalspetskompetens.se/wp-content/uploads/2020/06/DigitalSpetskompetens_Definition_Gulliksenetal.pdf
25. Hellas, A., Ihantola, P., Petersen, A., Ajanovski, V.V., Gutica, M., Hynninen, T., Knutas, A., Leinonen, J., Messom, C., Liao, S.N.: Predicting academic performance: a systematic literature review. In: Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education, pp. 175–199 (2018)
26. Kirschner, P.A.: Do we need teachers as designers of technology enhanced learning? Instructional Science **43**(2), 309–322 (2015). URL https://doi.org/10.1007/s11251-015-9346-9
27. Lishinski, A., Good, J., Sands, P., Yadav, A.: Methodological rigor and theoretical foundations of CS education research. In: Proceedings of the 2016 ACM conference on international computing education research, pp. 161–169 (2016)
28. Lister, R.: After the gold rush: Toward sustainable scholarship in computing (keynote address). In: Tenth Australasian Computing Education Conference, ACE 2008, pp. 3–17 (2008)
29. Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B., Thomas, L.: A multi-national study of reading and tracing skills in novice programmers. In: ITiCSE-WGR '04: Working group reports from ITiCSE on Innovation and technology in computer science education, pp. 119–150. ACM, Leeds, United Kingdom (2004). DOI http://doi.acm.org/10.1145/1044550.1041673
30. Luxton-Reilly, A., Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory programming: a systematic literature review. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, pp. 55–106 (2018)
31. Malmi, L., Sheard, J., Bednarik, R., Helminen, J., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Characterizing research in computing education: a preliminary analysis of the literature. In: Proceedings of the Sixth international workshop on Computing education research, pp. 3–12 (2010)
32. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Kinnunen, P., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Theoretical underpinnings of computing education research: what is the evidence? In: Tenth International Computing Education Research Conference, ICER 2014, pp. 27–34 (2014)
33. Malmi, L., Sheard, J., Kinnunen, P., Simon, Sinclair, J.: Computing education theories: what are they and how are they used? In: 15th International Computing Education Research Conference, ICER 2019, pp. 187–197 (2019)
34. Malmi, L., Sheard, J., Kinnunen, P., Simon, Sinclair, J.: Theories and models of emotions, attitudes, and self-efficacy in the context of programming education. In: 16th International Computing Education Research Conference, ICER 2020, p. 36–47 (2020)
35. Malmi, L., Sheard, J., Kinnunen, P., Sinclair, J.: Development and use of domain-specific learning theories, models and instruments in computing education. ACM Transactions on Computing Education (TOCE) (2022)
36. Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., Settle, A.: Computational Thinking in K-9 Education. In: Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference, ITiCSE-WGR '14, pp. 1–29. ACM, Uppsala, Sweden (2014). URL http://doi.acm.org/10.1145/2713609.2713610

37. Mannila, L., Nordén, L.å., Pears, A.: Digital Competence, Teacher Self-Efficacy and Training Needs. In: Proceedings of the 2018 ACM Conference on International Computing Education Research, pp. 78–85. ACM (2018). DOI https://doi.org/10.1145/3230977.3230993
38. Margulieux, L., Ketenci, T.A., Decker, A.: Review of measurements used in computing education research and suggestions for increasing standardization. Computer Science Education **29**(1), 49–78 (2019)
39. Martín-Ramos, P., Lopes, M.J., Lima da Silva, M.M., Gomes, P.E.B., Pereira da Silva, P.S., Domingues, J.P.P., Ramos Silva, M.: First exposure to Arduino through peer-coaching: Impact on students' attitudes towards programming. Computers in Human Behavior **76**, 51–58 (2017). DOI https://doi.org/10.1016/j.chb.2017.07.007. URL https://www.sciencedirect.com/science/article/pii/S0747563217304193
40. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., Laxer, C., Thomas, L., Utting, I., Wilusz, T.: A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. SIGCSE Bulletin **33**(4), 125–180 (2001). DOI http://doi.acm.org/10.1145/572139.572181
41. McGill, M.M., Decker, A.: A gap analysis of statistical data reporting in K-12 computing education research: recommendations for improvement. In: Proceedings of 51st SIGCSE Technical Symposium on Computer Science Education, pp. 591–597 (2020)
42. Morrison, B.B., Dorn, B., Guzdial, M.: Measuring cognitive load in introductory cs: adaptation of an instrument. In: Proceedings of the tenth annual conference on International computing education research, pp. 131–138 (2014)
43. Munasinghe, B., Bell, T., Robins, A.: Teachers' understanding of technical terms in a Computational Thinking curriculum. In: Australasian Computing Education Conference, ACE '21, pp. 106–114. Association for Computing Machinery, New York, NY, USA (2021). URL https://doi.org/10.1145/3441636.3442311
44. Nelson, G.L., Ko, A.J.: On use of theory in computing education research. In: Proceedings of the 2018 ACM Conference on International Computing Education Research, pp. 31–39 (2018)
45. Niemelä, P., Pears, A., Dagienė, V., Laanpere, M.: Computational Thinking – Forces Shaping Curriculum and Policy in Finland, Sweden and the Baltic Countries. In: D. Passey, D. Leahy, L. Williams, J. Holvikivi, M. Ruohonen (eds.) Digital Transformation of Education and Learning - Past, Present and Future, IFIP Advances in Information and Communication Technology, pp. 131–143. Springer International Publishing, Cham (2022). DOI https://doi.org/10.1007/978-3-030-97986-7_11
46. Papert, S.: Mindstorms: Computers, children, and powerful ideas. NY: Basic Books (1980)
47. Pears, A., Daniels, M.: Developing Global Teamwork Skills: The Runestone Project. In: M. Castro, E. Tovar, M.E. Auer (eds.) IEEE EDUCON 2010 – The Future of Global Learning in Engineering Education (2010)
48. Pears, A., Daniels, M., Berglund: Describing Computer Science Education Research: An Academic Process View. In: Conference on Simulation and Multimedia in Engineering Education, ICSEE'2002, San Antonio, Texas, pp. 99–104. Society for Computer Simulation International (2002). URL PearICSEE2002.pdf
49. Pears, A., Seidman, S., Eney, C., Kinnunen, P., Malmi, L.: Constructing a Core Literature for Computing Education Research. ACM SIGCSE Bulletin **37**(4), 152–161 (2005). DOI https://doi.org/10.1145/1113847.1113893
50. Pears, A., Dagiene, V., Jasute, E.: Baltic and Nordic K-12 Teacher Perspectives on Computational Thinking and Computing. In: V. Dagien\.e, A. Hellas (eds.) Informatics in Schools: Focus on Learning Programming: 10th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2017, Helsinki, Finland, November 13–15, 2017, Proceedings, pp. 141–152. Springer International Publishing, Cham (2017). URL https://doi.org/10.1007/978-3-319-71483-7_12
51. Pears, A., Tedre, M., Valtonen, T., Vartiainen, H.: What Makes Computational Thinking so Troublesome? In: 2021 IEEE Frontiers in Education Conference (FIE), pp. 1–7 (2021). DOI https://doi.org/10.1109/FIE49875.2021.9637416. ISSN: 2377-634X
52. Petre, M., Fincher, S.: Bootstrapping Research in Computer Science Education. In: Tacoma and Port Townsend, Washington, USA (2002). URL http://depts.washington.edu/bootstrp/

53. Petre, M., Sanders, K., McCartney, R., Ahmadzadeh, M., Connolly, C., Hamouda, S., Harrington, B., Lumbroso, J., Maguire, J., Malmi, L., et al.: Mapping the landscape of peer review in computing education research. In: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, pp. 173–209. ACM (2020)
54. Porter, L., Zingaro, D., Liao, S.N., Taylor, C., Webb, K.C., Lee, C., Clancy, M.: BDSI: A validated concept inventory for basic data structures. In: Proceedings of the 2019 ACM Conference on International Computing Education Research, pp. 111–119 (2019)
55. Randolph, J.J., Julnes, G., Sutinen, E., Lehman, S.: A methodological review of computer science education research. Journal of Information Technology Education: Research **7**(1), 135–162 (2008)
56. Robins, A.V.: Novice programmers and introductory programming. In: S. Fincher, A. Robins (eds.) The Cambridge handbook of computing education research, pp. 327–377. Cambridge University Press (2019)
57. Scheer, A., Noweski, C., Meinel, C.: Transforming Constructivist Learning into Action: Design Thinking in education. Design and Technology Education: an International Journal **17**(3) (2012). URL https://ojs.lboro.ac.uk/DATE/article/view/1758
58. Simon: A Classification of Recent Australasian Computing Education Publications. Computer Science Education **17**(3), 155 – 169 (2007). URL http://www.informaworld.com/10.1080/08993400701538021
59. Simon: Emergence of computing education as a research discipline. Ph.D. thesis, Aalto Univerity, Finland (2015)
60. Simon, Carbone, A., Raadt, M.d., Lister, R., Hamilton, M., Sheard, J.: Classifying Computing Education Papers: Process and Results. In: R. Lister, M. Caspersen, M. Clancy (eds.) Fourth International Computing Education Research Workshop (ICER 2008). ACM Press, Sydney, Australia (2008)
61. Singer, S.R., Nielsen, N.R., Schweingruber, H.A.: Discipline-Based Education Research: Understanding and Improving Learning in Undergraduate Science and Engineering. National Academies Press (2012)
62. Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M.L., Minsky, M., Papert, A., Silverman, B.: History of Logo. Proceedings of the ACM on Programming Languages **4**(HOPL), 1–66 (2020)
63. Soloway, E., Ehrlich, K.: Empirical studies of programming knowledge. IEEE Transactions on software engineering **SE-10**(5), 595–609 (1984)
64. Szabo, C., Sheard, J.: Learning theories use and relationships in computing education research. ACM Transactions on Computing Education (TOCE) p. in press (2022)
65. Szabo, C., Falkner, N., Petersen, A., Bort, H., Cunningham, K., Donaldson, P., Hellas, A., Robinson, J., Sheard, J.: Review and use of learning theories within computer science education research: primer for researchers and practitioners. In: ITiCSE 2019 Working Group Reports, ITiCSE WGR 2019, pp. 89–109. Association for Computing Machinery (2019)
66. Tedre, M., Toivonen, T., Kahila, J., Vartiainen, H., Valtonen, T., Jormanainen, I., Pears, A.: Teaching Machine Learning in K–12 Classroom: Pedagogical and Technological Trajectories for Artificial Intelligence Education. IEEE Access **9**, 110558–110572 (2021). DOI https://doi.org/10.1109/ACCESS.2021.3097962. Conference Name: IEEE Access
67. Tenenberg, J., Malmi, L.: Conceptualizing and using theory in computing education research. ACM Transactions on Computing Education (2022)
68. Utting, I., Tew, A.E., McCracken, M., Thomas, L., Bouvier, D., Frye, R., Paterson, J., Caspersen, M., Kolikant, Y.B.D., Sorva, J., et al.: A fresh look at novice programmers' performance and their teachers' expectations. In: Proceedings of the ITiCSE working group reports conference on Innovation and technology in computer science education-working group reports, pp. 15–32 (2013)
69. Valentine, D.W.: CS educational research: a meta-analysis of SIGCSE technical symposium proceedings. In: SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, pp. 255–259. ACM Press, Norfolk, Virginia, USA (2004). DOI http://doi.acm.org/10.1145/971300.971391

70. Wahyuningsih, S., Nurjanah, N.E., Rasmani, U.E.E., Hafidah, R., Pudyaningtyas, A.R., Syamsuddin, M.M.: STEAM Learning in Early Childhood Education: A Literature Review. International Journal of Pedagogy and Teacher Education **4**(1), 33–44 (2020). DOI https://doi.org/10.20961/ijpte.v4i1.39855.
71. Weinberg, G.M.: The psychology of computer programming, vol. 29. Van Nostrand Reinhold New York (1971)
72. Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., Wilensky, U.: Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology (2016). DOI https://doi.org/10.1007/s10956-015-9581-5
73. Wiggberg, M., Gulliksen, J., Cajander, å., Pears, A.: Defining Digital Excellence: Requisite Skills and Policy Implications for Digital Transformation. IEEE Access **10**, 52481–52507 (2022). DOI 10.1109/ACCESS.2022.3171924. Conference Name: IEEE Access
74. Wing, J.M.: Computational thinking. Communications of the ACM **49**(3), 33–35 (2006)
75. Xie, B., Loksa, D., Nelson, G.L., Davidson, M.J., Dong, D., Kwik, H., Tan, A.H., Hwa, L., Li, M., Ko, A.J.: A theory of instruction for introductory programming skills. Computer Science Education **29**(2–3), 205–253 (2019)
76. Yang, K., Liu, X., Chen, G.: The influence of robots on students' computational thinking: A literature review. International Journal of Information and Education Technology **10**(8), 5 (2020)

# Theory and Approaches to Computing Education Research


**Arnold Pears, Mats Daniels, and Anders Berglund**

## 1 Introduction

Research in Computing Education has developed considerably over the last 30 years both in terms of the topics attracting attention, and in terms of the questions addressed, selection of methodologies and use of underlying theories. This chapter provides an overview of some of the key theoretical contributions to the CER literature. We discuss how CER research can be approached, conducted and contextualised. The goal is not to characterise methods common in CER through an analysis of the research literature. Rather, we aim to provide our perspective on the theoretical and methodological issues that face CER, and contribute to the discussion of how rigourous CER studies can, and should, be structured. We hope that this chapter contributes to constructing a framework for meaningful CER. Overviews of the research field emphasise that there is a considerable variation in methodology and theoretical perspectives. In this context structured analysis of published research has demonstrated that, while the field of Computing Education Research (CER) has matured as a discipline, it also exhibits considerable diversity in terms of the approaches used [29] . Consequently, this chapter proposes a framing for CER building on prior work in the design and conduct of CER studies.

The authors of this chapter have been engaged in computing education research for close to three decades. In writing this chapter we provide our perspective on the conduct of CER and how such research can be structured and communicated. Our main objective is to support the endeavours of researchers entering the field

A. Pears (✉)
KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: pears@kth.se

M. Daniels · A. Berglund
Uppsala University, Uppsala, Sweden
e-mail: mats.daniels@it.uu.se; anders.berglund@it.uu.se



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_3

by providing concrete models for the conduct of research. Up until around the year 2000, one characteristic of the field was the predominant focus on descriptive reports of teaching practices and experiences in teaching computing, a major category of these were termed Marco Polo papers by Valentine [42]. Even considering the entire research corpus it is hard to discern a clear trend in terms of models and methods for conducting research. Bibliometric analysis, however, reveals a shift in focus from the dominant paradigm of quantitative and quasi-experimental studies, towards studies focusing on learning impact based on qualitative models [3]. These trends and detailed analysis of the published literature are discussed in detail in later chapters of this book.

Prominent researchers in CER, perhaps most notably Petre and Fincher, worked extensively to establish a tradition of mixed method research [34], which also stimulated international interest in multi-institutional studies [28]. While these initiatives were effective in forming a branch of the discipline they do not, despite two decades of effort, constitute a dominant paradigm. The contribution to methodology was considerable, however, since these studies were among the first in CER to adopt mixed methods approaches, and to provide considerable detail on the research study design in the publications that resulted from the work.

We advance the view that CER forms a bridge between education and learning sciences research, computing sciences research (in the discipline of computing) and the arena of learning and teaching computing sciences (Fig. 1). As Fincher et al. observe [17, p. 2], the initial focus on tertiary learning and university study in the discipline of Computer Science has long since vanished as Computing broadened to include many other areas, such as software engineering, information technology, informatics, data science and cyber-security, not to mention the enormous importance of recent developments in Artificial Intelligence and Machine Learning.

Viewed historically, computing education research (CER), as a branch of Discipline Based Education Research (DBER), displays a considerable eclecticism in terms of method, combining approaches from a range of qualitative and quantitative research traditions. A consequence of this is that we have invested considerable effort in thinking about the research area as a whole. We conclude that a key defining feature of computing education research is the focus on learning in the discipline. The point of departure for nearly all computing education research has been a desire to address educational challenges in the discipline, rather than to take a standpoint in an educational tradition. For those entering the field this is an important observation, since CER typically places the research objective, or question, in focus, making the choice of method a secondary concern for many computing education researchers.

This chapter argues that it is vital to define the nature of the emerging paradigm of CER. To that end we present a framework which scaffolds working in this paradigm, and discuss the tradeoffs that can emerge in designing research studies. During the past two decades there has been considerable discussion aiming to enhance rigour and research foundations in CER [6, 27]. We also argue that a general education research foundation is needed to complement our competence in the computing area [11].

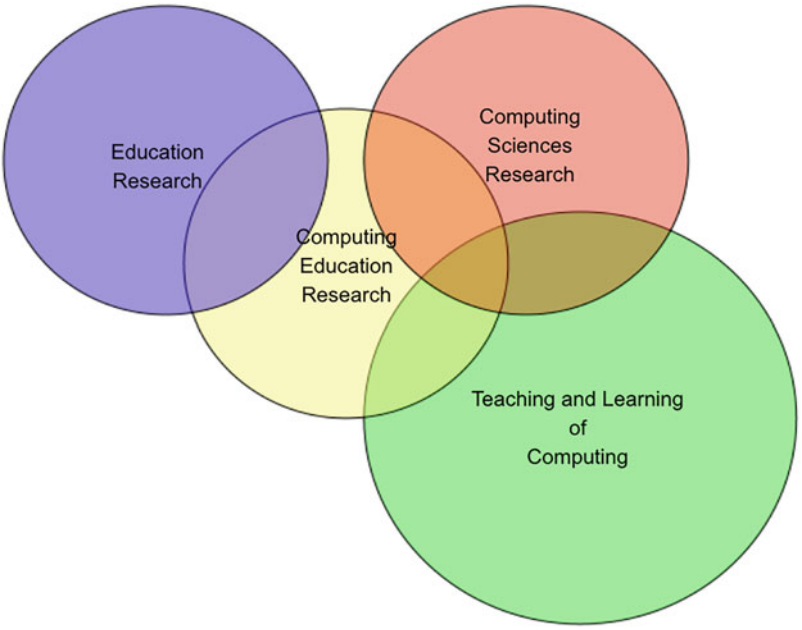


**Fig. 1** Positioning CER

The focus of the chapter is the conduct of CER in terms such as its context, how it could be conducted, and the philosophy behind it. Using discussions of theory from Crotty [9] and Pears et al. [33] and also adopted by Malmi et al. in 2019, we propose a categorisation of the manner in which theory applies to different aspects of CER. The goal of this classification is to help researchers entering the field to position their research, and to understand how theory can be used to support data collection, analysis and synthesis. We also propose models through which CER practitioners can discuss the framing of research in computing education. Finally we explore the area of educational development through action research proposed by Daniels et al. [11].

## 2 CER and Scholarship of Teaching and Learning

The notion of a scholarship of teaching and learning builds on the influential writings of Boyer [7] examining the roles and priorities of academics. The resulting worldwide movement towards a more scholarly approach to teaching and learning at university is characterised by professional development programmes and administrative changes affecting the regulation and conduct of undergraduate and graduate education.

Accreditation standards applied by professional bodies, such as ABET in the USA and EURO-ACE in Europe, create a demand for change. Alternative models of education such as CDIO have also been used to manage large scale change [8, 21]. The Project and Problem Based Learning (PBL) movement is also widely adopted and has strong proponents worldwide, but perhaps especially in Denmark [25].

The shift in the primary mission of tertiary education has been noted by other researchers. Supporting the evolving mission of universities, and also supporting staff in developing a more scholarly teaching approach has high priority. However, there are also some indications that investments in moving to a more scholarly approach are making slow progress. The role of CER therefore seems central to accelerating the rate of change, and providing evidence to drive innovative learning designs.

The complexity of approaching teaching and learning in a discipline requires tools and access to a research discourse beyond that which most discipline researchers experience. While they are familiar with the technical research in their own area of science or engineering, their awareness of research into the teaching and learning of their discipline is often limited. Even when academics are aware of the relevant educational discourses within their disciplines concrete approaches to applying the results to practice are often lacking. This chapter helps to bridge that gap by providing a pragmatic framework describing how computing education research can be directly used as an invaluable resource during the process of formulating an instructional design.

Since much of CER is driven by a pragmatic goal to understand learning phenomena associated with complex disciplinary knowledge/concepts it can be hard to associate the resulting scholarly output with an established research paradigm. In fact we propose that a certain degree of methodological eclecticism may be inherent in the practice of research, and scholarly practice in teaching and learning in our discipline.

This derives from the fact that the framing of research questions is based on a desire to better understand learning in a context, thus the choice of method often depends on the type of insight deemed most useful in that particular context. As a consequence it is not unusual for a single researcher to conduct both qualitative and quantitative studies, and subsequently drawing on elements of action research, phenomenography and statistical analysis to substantiate claims.

We have pursued the following three questions in this chapter.

- What is meant by “theory” in CER?
- What has been written about how research-based computing educational development can, or should, be structured?
- How can CER develop more rigorous research based on theory?

## 3 So What Is CER Theory?

The term "theory" is used in many ways in DBER and the use of theory in CER and Engineering Education Research (EER) has been explored from a range of perspectives. One of the purposes of this chapter is to acquaint the reader with this discourse, and to discuss how research in CER can increase its use to theory to achieve greater generalisability and impact. Scrutinising the literature one can identify three major bodies of work related to the theory of computing education, (1) historical and philosophical, (2) content and structure, and (3) processes and models.

### *3.1 Historical and Philosophical Perspectives*

The CER literature has been concerned with what computing is and how it should be taught since its inception. Many significant technical and theoretical figures have contributed to this debate throughout the years, with perhaps the most influential opinion piece of all time being the CACM paper "Go To Statement Considered Harmful" [16]. Other well known voices in the debate on what computing is and how to teach it include Peter Denning who, in collaboration with colleagues such as Matti Tedre, Raymond Lister, Tony Clear, and many others, has contributed over more than three decades to the debate surrounding what computing is, and what should be taught, and how [10, 12–15, 39–41]. While these publications are relevant to the nature of computing, and the theoretical content to be taught, they do not concern theoretical development in CER itself, rather, these publications, define the content of the upper right hand element of Fig. 2.

### *3.2 Content and Structure*

Some of the earliest attempts to understand what CER was about and what was considered "good scholarship" focused on the research artifacts produced attempting to classify them and attribute value to them [32, 36–38, 42]. These works segment and stratify the research literature attempting to ascribe quality and impact to the outputs of research activity through measures such as citation, nominations from leading researchers, and assessment of promise (see particularly the "seminal" literature category proposed by Pears et al. in 2005 [32]). Simon in his classifications, and ultimately in his PhD thesis provided a much more detailed classification and subsequent analysis arguing that the sophistication of the research area met the requirements to be considered a research discipline, which also concurs with the conclusions reached in chapter "What is Computing Education Research (CER)" of this book.

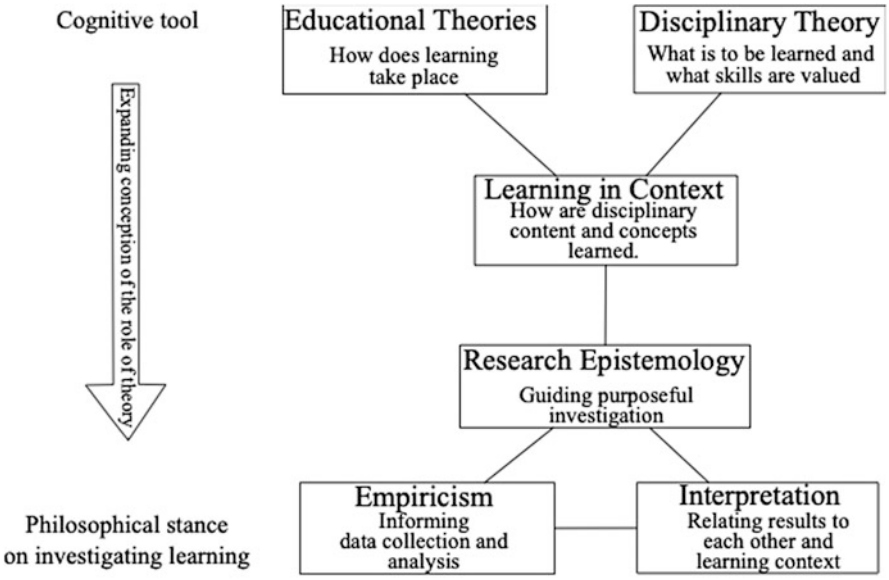


**Fig. 2** Perspectives on theory

Another approach, also focusing on the research output in terms of conference and journal contributions is to analyse the extant literature extracting as much information about the epistemology, methodology and methods as can be gleaned from the published texts. See for example the works of Ben-Ari [4], Randolph [35], Berglund [5], Malmi et al. [29] and Fincher et al. [17]. These works make a significant contribution to our understanding of what research has been conducted, and what methods predominate in the literary corpus. Such works also provide arguments for diversification of method, as well as arguing for diversity in terms of epistemology and methodology as a response to the varied research foci permeating CER.

For instance, Berglund et al. [6] focusing on the use of qualitative methods in CER, argued that a greater emphasis on qualitative methods was needed to offset the positivist tendency towards control group studies and statistical significance in CER inherited from the experimental computer science background of many CER academics. They offered a means to structure CER into subfields based on observed areas of concentrated research. They concluded that a number of distinct subfields could be identified, small scale investigations of individual practice, investigations motivated by the development of educational tools and technologies, and finally studies in the psychological and educational traditions. They argued that the conscious selection of approach and method helped to facilitate communication with other researchers and ultimately enhance the generalisability of results allowing for more robust disciplinary theory to be constructed.

Daniels and Pears [11] continued this line of argument, proposing that studies should be structured to make explicit the levels in the research ecology model of Crotty [9]. This permits a delineation of studies in CER based on an underlying commitment to epistemology, theoretical perspective, and resulting choices regarding methodology and method. However, the situation in CER is multi-faceted and the term "theory" can mean many things depending on the context.

Malmi et al. [29] build on an earlier analysis of 308 papers from three major venues, the ACM International Computing Education Research conference (ICER), ACM Transactions on Computing Education (TOCE), and the Computer Science Education Journal (Taylor and Francis) (CSEd) [30]. They combine their earlier analysis with the work of Lishinski et al. [27], searching for theoretical frameworks and the extent to which theory, models or frameworks are used in CER studies. Their 2019 study attempted to both identify that the authors termed *theoretical constructs* (TCs) and describe how they were used. To accomplish this the TCs were developed by reading the research corpus of the three publication venues ICER (N = 192), CSEd (N = 172) and TOCE (N = 176) in the year range 2005–2015 and relating the focus of the TCs to the didactic triangle [23]. Of the total collection of articles explored ICER had the highest percentage of published works reporting a new TC (N = 38, 20%), followed by CSEd (N = 21, 12%) and TOCE (N = 6, 3%). In summary, they find that published CER work is dominated by research exploring TCs that deal with the student-content relationship edge of the didactic triangle (N = 52), followed by teacher's pedagogical actions (N = 10), and content (N = 6). The vast majority of TCs were developed using qualitative methods, among the most popular being grounded theory and phenomenography, while the dominant quantitative method was regression analysis. They also observe that in 90% of cases, papers that refer to TCs and cite other publications that developed TCs do not subsequently use those TCs in the paper in any meaningful manner. Some research publications, however, did use TCs to inform the work presented in the paper, and the authors provide four examples to illustrate how this can be done, *using theory to discuss and reason about results* (see Fig. 2 bottom right hand block dealing with interpretation), *using theory to predict results, establish an hypothesis* (see Fig. 2 middle section dealing with guiding purposeful investigation), *using theory to inform pedagogy and test the results* (see Fig. 2 middle section dealing with learning in context), and *using a TC as a data analysis framework* (see Fig. 2 bottom left hand block dealing with informing data collection and analysis). They also observe that TCs may not always be made visible in the written article, but, rather are tacit elements of many studies. This conclusion emphasises the need for a more rigorous and explicit treatment of theory in most CER.

### *3.3 Explicit Use of Theory*

The most common approach to rendering tacit elements of the research ecology explicit in research reporting is to provide explicit support for the systematic use

of theory. Several members of the CER community have discussed this issue. We discuss each of them below, as they provide researchers entering the field with considerable practical guidance in terms of reasoning about how to identify research questions, apply theoretical framing to the design and conduct of a study. We also discuss how theory can be used to integrate research results, and connect new work to prior findings.

An attempt to represent the complexity of the situation and discuss the roles of theory in EER appears in Pears et al. [33]. The main classification proposed there is shown in Fig. 2. The upper half of the Figure depicts the types of theory generally considered relevant to curricular and instructional design. The theoretical perspectives associated with combining Educational Theory (Episteme) with Disciplinary Theory to be learned is described in the figure as Learning in Context, but, bears many similarities to the concepts of Pedagogical Content Knowledge [19], and for STEM disciplines the TPACK framework [24].

The Learning Context can also be seen as providing the inspiration for the idea of the teacher context shown in Fig. 4a. In this figure, however, the focus is on the instructional design that results from the integration of Educational and Disciplinary theory in the context of the institution and higher education regulatory system and questions that arise in terms of understanding the impact of the instructional design on student learning.

The transition between Learning in Context and Research Epistemology in Fig. 2 marks a shift in focus, from theory as a basis for informing educational practice to using theory to inform and guide research into the learning situation. That is we make a shift from using theory as a means to reason about what learning is and how it takes place, incorporating understanding of the disciplinary theory that is the subject of instruction and the focus of the learning activity, to taking an epistemological standpoint in terms of how research into teaching and learning in that context can be performed.

The impact of the epistemological framing of CER research is clearly illustrated in Fig. 7, drawn from the work of Kinnunen presented in Pears et al. [33]. The main insight to be gained here is that the manner in which data are collected tend to reflect the underlying epistemological convictions of the research. Additionally, a researcher's (or research community's) epistemological roots tend to affect what kinds of questions are posed, the data collection and analysis methods deployed, and the kinds of research designs that are built. It seems that a research community's 'roots' form schools of thought within the research field that take a stance on what is seen as (particularly) valid, or valued, knowledge. If a researcher starts to collect data, without reflecting on the type of knowledge claims the data can support, the nature of a study can become confused. This is evident in the recent review conducted by Malmi et al. [29] in which it was found that few papers published in CER discuss the epistemology of their data collection.

Finally, Nelson and Ko [31] have argued that theory can actively hinder the development of CER. They argue that the main objective of CER is to evaluate learning design and that when CER subscribes to a narrow view of how to theoretically position research this inhibits the development of innovative teaching

and learning designs. The three problems they identify are that teaching and learning research couched in the educational and social sciences traditions, and often with a focus on exploring the experiences of learners, is not well aligned with innovation and instructional design purposes. This, in combination with reviewing norms in the field, they argue make it difficult to publish some types of design research which they feel is central to successful CER. In common with Malmi [30], they advocate more focus on developing CER specific theory.

## 4 Frameworks and Research Models

In contrast to a focus on the outputs of research, another approach taken to understanding CER activity and arguing for scholarly quality was to take a process modelling approach. This type of approach is discussed in detail in Sect. 4.1 where a number of the published models are summarised and their impact discussed. One aim of these types of models and frameworks is to help academics with emerging interest in CS Education to understand how research in this area is different from research in their traditional areas. A second and more important aim is that such models provide a basis from which to discuss how research is conducted, and how a long term research agenda can be constructed.

In the case of CER much effort has focused on the integration of new technology into computer science (CS) and Information Technology (IT) education programmes. This type of activity is often accompanied by studies which aim to understand and improve the teaching and learning process. How we evaluate the potential of emerging technologies and integrate them into teacher education has clearly become increasingly important. In this context it is important to define the elements of quality research in order to enhance the development, deployment and understanding of educational innovation in scientific disciplines, and specifically in computing.

Several models have been proposed that build on the corpus of research activities in Computing education with the aim of extracting and analysing the underlying principles that contribute to a valuable study of an educational context. Such models present and integrate the multi-disciplinary elements inherent in CER making it easier to explain what is involved and how the elements interact.

### *4.1 Process Models*

Early attempts to model and explain the research process as a contribution to establishing guidelines for research process and rigour were first published in around 2000. Candidate abstractions for designing educational studies include those of Langerth et al. [26] and Holmboe et al. [22]. An early attempt at defining an applied framework gave a very practical applied view of the education research.

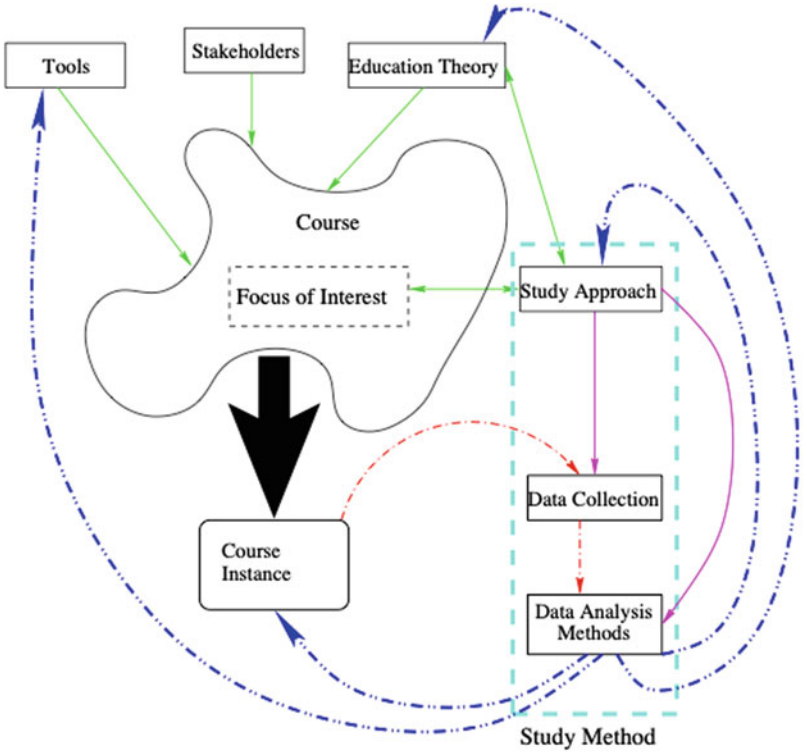


**Fig. 3** Applied educational research

The process was centered around the aspects most vital to the teacher/researcher, and assumed that the goal was to investigate personal educational practice.

The applied educational practice research model proposed by Pears is depicted in Fig. 3. In this model the influences on a course are **tools**, **stakeholders** and **education theory**.

Examples of **tools** are course web sites, laptop computers, computer based teaching products and wireless networking.

**Stake-holders** refers to the community which have an influence on the content, form and approach taken in designing a course.

Implicit or explicit ideas on how teaching and learning take place are represented by the **education theory** box.

There is some aspect of what happens in the course context that we wish to learn more about. This is the **focus of interest**.

Investigating a **focus of interest** requires additional types of activity. Namely, **study approach**, **data collection** and **data analysis**. The heavy dotted arrows show reverse flows representing the influence of feedback on subsequent course instances and research studies.

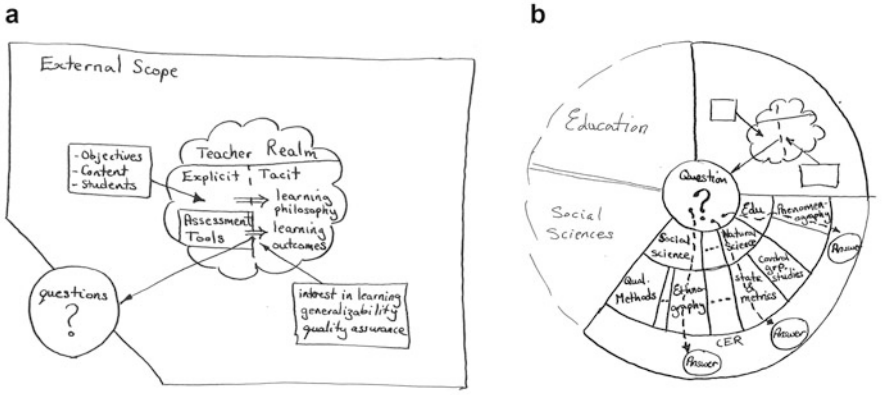


**Fig. 4** Research context. (**a**) Teacher focus. (**b**) Episteme

The choice of **study approach** depends on both the ideas about teaching and learning and the **focus of interest** itself, since the investigative techniques must be suited to the aspects of the learning situation being investigated. **Data collection** techniques are then chosen which provide apposite data for subsequent analysis.

This approach was further refined in Daniels and Pears [11] where they distinguish between the teaching context shown in Fig. 4a and the epistemological foundations lying behind choices of methodology and method shown by the potential study design pathways depicted in Fig. 4b.

While course plans and international curricula [1, 2, 18] provide academic teachers with a guideline as to what should be included in terms of subject matter, and what skills, knowledge and dispositions a student is expected to develop during the course, they typically say very little about instructional design. As Nelson and Ko argue [31], instructional design is a key element of CER and informs the manner in which the course is taught and assessed in order to motivate learners and to achieve the expected outcomes. As a result university lecturers typically adopt a pragmatic approach based on their own prior experience as students, often teaching the material in a similar manner to that in which it was taught to them.

A research informed approach integrates a richer set of resources, and applies them to informing the design of the learning environment and assessment. A visualisation of the overall process and relationships between the activities involved in this model of research driven course design are summarised in Fig. 5.

The upper triangle depicts the research informed or "scholarly" teaching and learning activities that contribute to the instructional design. The focus of this cycle of activities is on relating theoretical and empirical results to the instructional design of a course and their influence on course structure, conduct and assessment.

The central part of the figure shows the instructional setting. Cohorts of students pass into a course and emerge demonstrating skills, knowledge and competencies specified in the learning outcomes in varying degrees. The extent to which

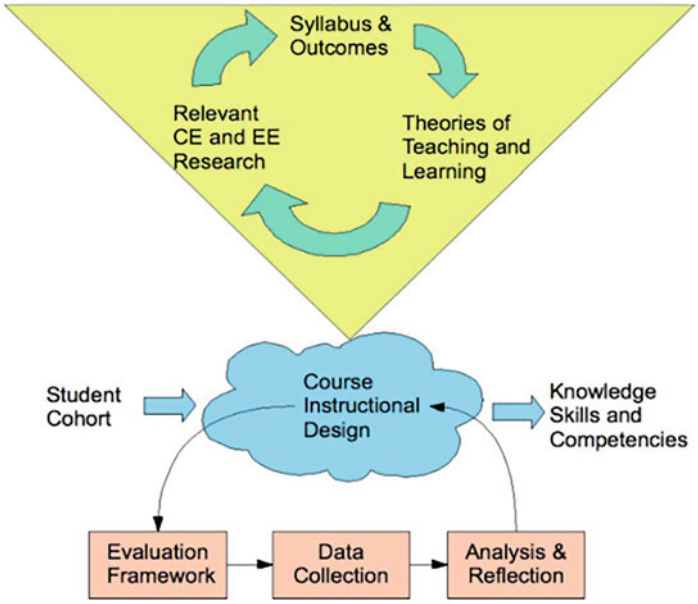


**Fig. 5** A research-based instructional design process

individuals are capable of demonstrating and performing in the relevant learning outcome domains determines their final grade. In this paper we are interested in the relationship between the processes in the triangle and the course design.

The lower section of the figure deals with assessing the merits of the instructional design. The intention of this part of the framework is to make collection of relevant data and reflection on how well the approach has succeeded, an integral part of an informed teaching approach.

Thota in her thesis and subsequent research proposes a related model in which the research process is described as a sort of Venn Diagram of overlapping areas of concern. Each of the areas of the model represent an aspect of research that is crucial to achieving rigour, high quality and in the process enhancing the reliability, and generalisability of the outcomes. The total model is presented in Fig. 6 and emphasises the role of, *research purpose*, *philosophical assumptions*, *design procedures*, and *research outcomes*.

In each of these areas a number of key aspects are discussed in relation to the overall research enterprise. Following this model implies that good research, in addition to aligning with the six attributes of quality research of Glassick [20] discussed in chapter "What is Computing Education Research (CER)", should also make the elements of the four regions explicit to the reader. Though it should be noted that there is significant overlap with Glassick in the areas of Research purpose, Design procedures, and Research outcomes.

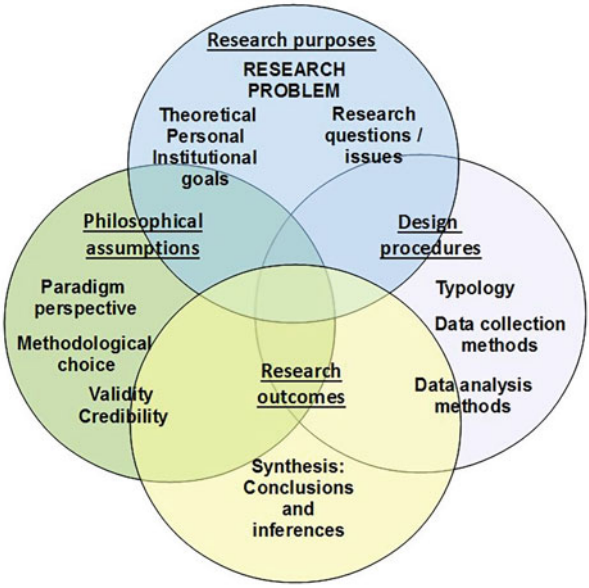


**Fig. 6** Thota's perspective on research in context

To summarise briefly, good research can be characterised in this model as a commitment to explicit consideration of the philosophical assumptions (perspectives on research paradigm, methodological choices, and credibility and reliability), research purpose (encompassing theoretical, personal and institutional goals, as well as research questions and types of desired result or research outcome), design procedures (decisions regarding types and quantity of data to be collected, typologies, and choice of data analysis method), and the research outcomes (including synthesis of results, conclusions and inferences that can be drawn). Thota observes that explicit attention to all of these areas should also be the hallmark of rigorous and high quality research articles and conference publications.

The final model considered in this chapter is that proposed by Kinnunen in Pears et al. [33], which extends the ideas of Crotty in terms of exemplifying three types of distinct research ecology represented as trees. From these three epistemological roots grow great trees, exemplifying the manner in which a researcher's epistemic commitments affect the manner in which data is collected and analysed, as well as the nature of the knowledge claims that are highly valued. This model is an effective reflective tool in the process of research design, helping to focus our attention on how our choices influence the types of answers we can obtain to a given research question. Interestingly the tree metaphor used in this model corresponds more or less exactly to the paths from the research question to a different types of knowledge claim, or result/answer shown in the model in Fig. 4.

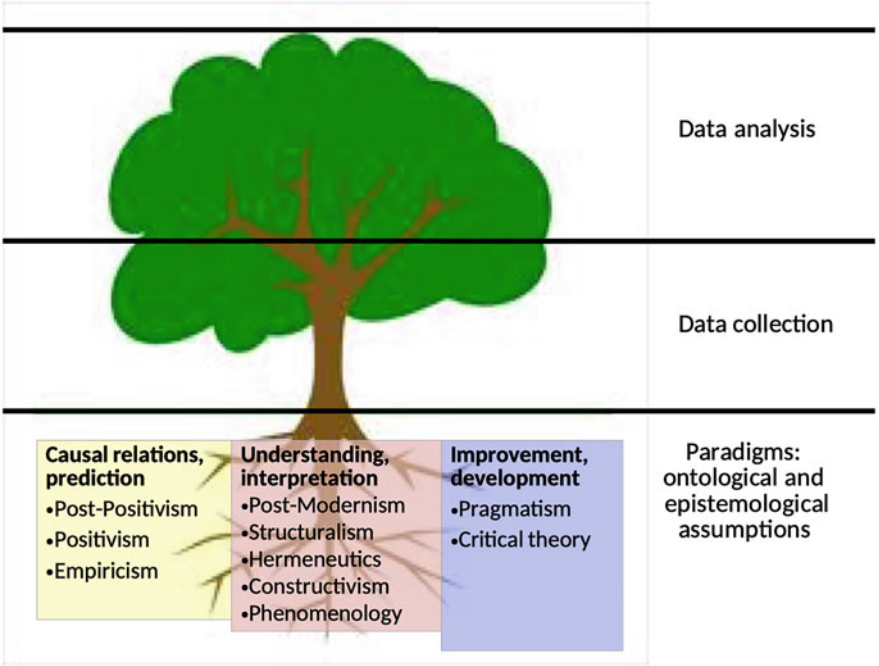


**Fig. 7** Epistemological traditions

## 5 Conclusions

This chapter provides a guide to the nature of theory in the context of computing education research (CER) informed by the perspectives and research of the authors. We develop a model which identifies the different types of theory that can be relevant to CER, and exemplify the approaches used in much of the previously published research literature [29].

One of the key observations is that one of the distinguishing features of CER research studies, in comparison to education research, lies in the point of departure, or focus of the research. Many authors argue that CER addresses concrete teaching and learning challenges in the discipline and thus draws on those methods appropriate to the particular research question being investigated.

We argue that it is this pragmatic focus on the *question as paramount*, that characterizes CER and other discipline based education research. The question, and the nature of useful answers, dictate the choice of methods for data collection and analysis to a much greater extent than is normal in education research.

We also conclude that, as a result of this rich research ecology, there is a need for a framework which assists researchers in contextualizing their study, and describing the context at a level of detail that permits generalization. Three such models are presented and discussed in detail, which we hope will prove a useful reference point for other researchers as they communicate their results.

A number of analyses of the research literature note a trend towards increased focus on empiricism, and a significant increase in the proportion of published works that utilise theoretical approaches from learning sciences, education and psychology. We note also that this trend is not universally viewed with approbation. Some criticism of peer review with a strong focus on empirical studies and a narrow definition of what theoretical framing is appropriate in CER has emerged in recent years [31].

Finally, CER can be seen in many ways as a type of disciplinary Scholarship of Teaching and Learning (SoTL), and consequently involves applying scholarly values to academic educational activities. This chapter demonstrates how an instructional design can draw on theory grounded in the published disciplinary teaching and learning scholarship, as well as educational research.

The goal of this chapter is to provide a series of models for the design and implementation of rigorous CER studies. Our intended audience are researchers entering the field, and PhD students orientiating themselves in this area of research. The chapter provides a much needed discussion of the role and nature of the term "theory" in CER. The methods proposed here have been used extensively in our own research designs over the last two decades, and we take this opportunity to share them with future generations of researchers.

The main contribution is the discussion of what constitutes research rigour and quality, and what frameworks and models have been proposed as systematic ways to think about the conduct of research. We also address how CER papers are written in order to better facilitate the transfer of research results between researchers, and into practice. Further discussion of the key aspects of this process in the context of teaching computing at all levels of the school and university curriculum is needed, and we hope that this chapter forms a cornerstone in that endeavour.

## References

1. ACM/IEEE-CS Joint Task Force: Computing Curricula 2001: Computer Science; Final Report. Tech. rep., ACM/IEEE CS (2001)
2. ACM/IEEE-CS Joint Task Force: Computer Science Curricula 2013 Strawman Draft. Tech. rep., ACM/IEEE CS (2012)
3. Apiola, M., Saqr, M., López-Pernas, S., Tedre, M.: Computing education research compiled: Keyword trends, building blocks, creators, and dissemination. IEEE Access **10**, 27041–27068 (2022). https://doi.org/10.1109/ACCESS.2022.3157609
4. Ben-Ari, M., Berglund, A., Booth, S., Holmboe, C.: What Do We Mean by Theoretically Sound Research in Computer Science Education? In: Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE '04, pp. 230–231. Association for Computing Machinery, New York, NY, USA (2004). URL https://doi.org/10.1145/1007996.1008059. Event-place: Leeds, United Kingdom
5. Berglund, A., Andersson, S., Elmgren, M., Pears, A.: Discipline-based staff development courses to promote a sustainable SOTL environment: An example from science and engineering at Uppsala University. In: Proc. ICED 2014: Educational development in a changing world. International Consortium for Educational Development (2014)

6. Berglund, A., Daniels, M., Pears, A.: Qualitative Research Projects in Computing Education Research: An Overview. Australian Computer Science Communications **28**(5), 25–34 (2006)
7. Boyer, E.: Scholarship Reconsidered: Priorities of the Professoriate. Carnegie Foundation for the Advancement of Teaching and Josey-Bass, Hillsdale, NJ (1990)
8. Brodeur, D.R., Crawley, E.F.: CDIO and Quality Assurance: Using the Standards for Continuous Program Improvement. Engineering Education Quality Assurance pp. 211–222 (2009). URL https://doi.org/10.1007/978-1-4419-0555-0_17
9. Crotty, M.J.: The Foundations of Social Research: Meaning and Perspective in the Research Process. The Foundations of Social Research, pp. 1–256 (1998). http://digital.casalini.it/9781446283134. URL https://www.torrossa.com/en/resources/an/5019222. Publisher: SAGE Publications Ltd
10. Daniels, M.: Developing and Assessing Professional Competencies: a Pipe Dream?: Experiences from an Open-Ended Group Project Learning Environment. PhD Thesis, Acta Universitatis Upsaliensis (2011)
11. Daniels, M., Pears, A.: Models and methods for computing education research. Australian Computer Science Communications **34**(2), 95–102 (2012)
12. Denning, P.J.: A debate on teaching computing science. CACM **32**, 1397–1414 (1989). URL http://doi.acm.org/10.1145/76380.76381
13. Denning, P.J.: Editorial: what is software quality? Commun. ACM **35**(1), 13–15 (1992). DOI http://doi.acm.org/10.1145/129617.384272
14. Denning, P.J., Tedre, M.: Computational Thinking. The MIT Press, Cambridge, Massachusetts (2019)
15. Denning, P.J., Tedre, M.: Computational Thinking: A Disciplinary Perspective. Informatics in Education (2021). https://doi.org/10.15388/infedu.2021.21. URL https://infedu.vu.lt/journal/INFEDU/article/701. Publisher: Vilnius University Institute of Data Science and Digital Technologies
16. Dijkstra, E.: Go To statement considered harmful. Communications of the ACM **11**(3), 147–148 (1968)
17. Fincher, S.A., Robins, A.V.: The Cambridge handbook of computing education research. Cambridge University Press (2019)
18. Frezza, S., Pears, A., Daniels, M., Kann, V., Kapoor, A., McDermott, R., Peters, A.K., Wallace, C., Sabin, M., Cajander, A.: Modeling global competencies for computing education. In: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, pp. 348–349 (2018)
19. Gess-Newsome, J.: Pedagogical Content Knowledge: An Introduction and Orientation. In: J. Gess-Newsome, N.G. Lederman (eds.) Examining Pedagogical Content Knowledge: The Construct and its Implications for Science Education, pp. 3–17. Springer Netherlands, Dordrecht (1999). URL https://doi.org/10.1007/0-306-47217-1_1
20. Glassick, C.E., Huber, M.T., Maeroff, G.I., Boyer, E.L.: Scholarship assessed: evaluation of the professoriate. Jossey-Bass, San Francisco (1997)
21. Gunnarsson, S., Wiklund, I., Svensson, T., Kindgren, A., Granath, S.: Large Scale Use of the CDIO Syllabus in Formulation of Program and Course Goals. In: Proceedings of the 3rd International CDIO Conference. MIT, Cambridge, Massachusetts (2007)
22. Holmboe, McIver, George: Research Agenda for Computer Science Education. In: Proc. Psychology of Programmers Interest Group (PPIG) (2001)
23. KANSANEN, P.: Studying–the Realistic Bridge Between Instruction and Learning. An Attempt to a Conceptual Whole of the Teaching-Studying- Learning Process. Educational Studies **29**(2–3), 221–232 (2003). URL https://doi.org/10.1080/03055690303279. Publisher: Routledge _eprint
24. Koehler, M., Mishra, P.: What is Technological Pedagogical Content Knowledge (TPACK)? Contemporary Issues in Technology and Teacher Education **9**(1), 60–70 (2009). URL https://www.learntechlib.org/p/29544. Place: Waynesville, NC USA Publisher: Society for Information Technology & Teacher Education

25. Kolmos, A.: Problem-Based and Project-Based Learning. University Science and Mathematics Education in Transition pp. 261–280 (2009). URL https://doi.org/10.1007/978-0-387-09829-6_13
26. Langerth, M., Strömdahl, H.: Theory Anchored Evaluation. In: Proceedings of the WGLN Workshop on Teaching Evaluation (2001). URL http://www.learninglab.uu.se/ssi/theory_anchored_evaluation.htm
27. Lishinski, A., Good, J., Sands, P., Yadav, A.: Methodological Rigor and Theoretical Foundations of CS Education Research. In: Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16, pp. 161–169. Association for Computing Machinery, New York, NY, USA (2016). URL https://doi.org/10.1145/2960310.2960328. Event-place: Melbourne, VIC, Australia
28. Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B., Thomas, L.: A multi-national study of reading and tracing skills in novice programmers. In: ITiCSE-WGR '04: Working group reports from ITiCSE on Innovation and technology in computer science education, pp. 119–150. ACM, Leeds, United Kingdom (2004). DOI http://doi.acm.org/10.1145/1044550.1041673
29. Malmi, L., Sheard, J., Kinnunen, P., Simon, Sinclair, J.: Computing Education Theories: What Are They and How Are They Used? In: Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER '19, pp. 187–197. Association for Computing Machinery, New York, NY, USA (2019). URL https://doi.org/10.1145/3291279.3339409. Event-place: Toronto ON, Canada
30. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Kinnunen, P., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Theoretical Underpinnings of Computing Education Research: What is the Evidence? In: Proceedings of the Tenth Annual Conference on International Computing Education Research, ICER '14, pp. 27–34. Association for Computing Machinery, New York, NY, USA (2014). URL https://doi.org/10.1145/2632320.2632358. Event-place: Glasgow, Scotland, United Kingdom
31. Nelson, G.L., Ko, A.J.: On Use of Theory in Computing Education Research. In: Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER '18, pp. 31–39. Association for Computing Machinery, New York, NY, USA (2018). URL https://doi.org/10.1145/3230977.3230992. Event-place: Espoo, Finland
32. Pears, A., Seidman, S., Eney, C., Kinnunen, P., Malmi, L.: Constructing a Core Literature for Computing Education Research. ACM SIGCSE Bulletin **37**(4), 152–161 (2005). https://doi.org/10.1145/1113847.1113893
33. Pears, A., Thota, N., Kinnunen, P., Berglund, A.: Harnessing theory in the service of engineering education research. In: Proc. 42nd ASEE/IEEE Frontiers in Education Conference, pp. 391–395. IEEE (2012)
34. Petre, M., Fincher, S.: Bootstrapping Research in Computer Science Education. In: Tacoma and Port Townsend, Washington, USA (2002). URL http://depts.washington.edu/bootstrp/
35. Randolph J. J., Bednarik, R., Myller, N.: A Methodological Review of the Articles Published in the Proceedings of Koli Calling 2001-2004. In: Proceedings of Koli Calling 2005: 5th Annual Finnish/Baltic Sea Conference on Computer Science Education, pp. 103–109 (2005)
36. Simon: A Classification of Recent Australasian Computing Education Publications. Computer Science Education **17**(3), 155–169 (2007). URL http://www.informaworld.com/10.1080/08993400701538021
37. Simon, Carbone, A., Raadt, M.d., Lister, R., Hamilton, M., Sheard, J.: Classifying Computing Education Papers: Process and Results. In: R. Lister, M. Caspersen, M. Clancy (eds.) Fourth International Computing Education Research Workshop (ICER 2008). ACM Press, Sydney, Australia (2008)
38. Simon, De Raadt, M., Venables, A.: Approaches to Lab Practical Classes among Computing Academics. Informatics in Education **6**(1), 215–230 (2007)
39. Tedre, M.: The Science of Computing: Shaping a Discipline. Taylor & Francis (2014). URL https://books.google.se/books?id=I2tYBQAAQBAJ

40. Tedre, M., Denning, P.J.: The long quest for computational thinking. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli Calling '16, pp. 120–129. Association for Computing Machinery (2016). URL https://doi.org/10.1145/2999541.2999542
41. Tedre, M., Toivonen, T., Kahila, J., Vartiainen, H., Valtonen, T., Jormanainen, I., Pears, A.: Teaching Machine Learning in K-12 Classroom: Pedagogical and Technological Trajectories for Artificial Intelligence Education. IEEE Access **9**, 110558–110572 (2021). https://doi.org/10.1109/ACCESS.2021.3097962. Conference Name: IEEE Access
42. Valentine, D.W.: CS educational research: a meta-analysis of SIGCSE technical symposium proceedings. In: SIGCSE '04: Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, pp. 255–259. ACM Press, Norfolk, Virginia, USA (2004). DOI http://doi.acm.org/10.1145/971300.971391

# The Evolution of Computing Education Research: A Meta-Analytic Perspective


**Lauri Malmi, Jane Sinclair, Judy Sheard, Simon, and Päivi Kinnunen**

## 1 Introduction

In this book, the main perspective for analysing the development of computing education research (CER) is scientometric analysis, which focuses on authors and their collaborations, publication venues, and research topics in terms of keyword analysis. There are sophisticated algorithms and tools that perform this analysis automatically on the basis of publication metadata. However, there are several other perspectives for analysis that can provide deeper and more refined information. Scientometric methods fall short of answering questions such as what theoretical frameworks have been used in research, what research methodologies, data collection tools, or analysis methods have been used, and how the publications match various criteria for research quality or nature. Keyword analysis can give some information in these areas, but a more detailed analysis requires manual inspection, at least of the abstracts and sometimes of the full papers.

L. Malmi (✉)
Aalto University, Espoo, Finland
e-mail: Lauri.Malmi@aalto.fi

J. Sinclair
University of Warwick, Coventry, UK
e-mail: J.E.Sinclair@warwick.ac.uk

J. Sheard
Monash University, Melbourne, VIC, Australia
e-mail: judy.sheard@monash.edu

Simon
Wadalba, NSW, Australia

P. Kinnunen
University of Helsinki, Helsinki, Finland
e-mail: paivi.kinnunen@helsinki.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_4

Over the past 20 years, many researchers have carried out significant work in various meta-studies, mapping studies, and systematic literature reviews in CER, addressing, for example, use of theories [37, 38, 40, 74], research methods [32, 41, 42, 63], learning tools and environments [2, 46, 54, 65], or specific topics, such as introductory programming [36]. In addition, many studies have investigated computing education publications in specific venues, such as the Australasian Computing Education conference [66], Koli Calling [59, 68], and the SIGCSE Technical Symposium [6].

In this chapter, we focus on what these reviews can reveal about the history and development of the CER field. We acknowledge that our perspective is limited to the more recent developments, as there are very few studies that have explored the early years of computing education literature. Valentine [83] explored 20 years of CS1 papers in the SIGCSE Technical Symposium (1984–2003), broadly categorising the type of work that was reported. Becker and Quille [6] explored almost 50 years of research on CS1 in SIGCSE (1970–2018), focusing on the topics of the research. However, we are not aware of any reviews that cover papers before 2000 and discuss the nature and process of the research, or that reflect on the richness and maturity of the field, and the quality and variety of evidence presented.

The absence of such analyses is not surprising considering how the field began. Computer science itself is a new research field, with roots in electrical engineering, mathematics, and science [76], which gained its own identity between 1940s and 1960s. It had to struggle to establish its own position in the academic world, with its own departments and methods [13, 76]. Since then it has grown and greatly expanded its scope, and today its applications are used in practically all fields of science and in society more broadly.

Computing education has naturally followed this development [75]. In 1950s, training of programmers was conducted mainly in computer companies, while mathematics departments within academia focused more on formal aspects of programming and logic. The first ACM Computing Curricula, published in 1968 and 1977 [3, 4], helped universities to set up their computer science degree programs in a more uniform way. In this phase, it became natural to organise conferences where pedagogical practices were discussed and novel educational innovations were presented. The SIGCSE Technical Symposium, launched in 1970, was the first leading venue to focus solely on computing education, followed later by other venues where computer science teachers could meet one another, present their work, and exchange ideas and experiences. This exchange was the prevalent idea in the beginning, as the whole concept of ‘computing education research’ or ‘computer science education research’ was still vague, and teaching practitioners placed a high value on acquiring new ideas and sharing experiences. In the meantime, educational scientists were conducting research, addressing challenges emerging in more established disciplines and investigating teaching, learning, and studying in generic terms.

While most computing educators in 1970s and 1980s focused on developing pedagogical approaches and adequate learning resources, significant research was already being carried out in CER, especially investigating the work of professional

programmers and the learning of programming. Weinberg's book, Psychology of Programming, was published in 1971 [85]. Papert's book addressing children's learning of programming, turtle graphics, and LOGO was published in 1980 [51]. Soloway and Ehrlich carried out important research on experts' programming plans in 1980s [72] as well as comparing the ways that novice and expert programmers work. Important early venues for presenting research were the Empirical Studies of Programmers (ESP) conferences and Psychology of Programming Interest Group (PPIG) workshops, which both started in 1986. The ESP conferences ceased in 1990s, but PPIG is still active. Guzdial and du Boulay [22] present a more comprehensive history of the early years of research in computing education.

In this chapter, we focus on later developments, mainly during the past 20 years, when there emerged a growing awareness and interest in the quality of evidence of the impact of educational innovations, as well as building deeper theoretical understanding of the factors involved in teaching and learning computing. We discuss these developments in the light of reviews and meta-analyses of CER that have been carried out during this period. We begin by looking at the development of CER using Fensham's framework [17], which was originally developed to analyse the growth of a neighbour discipline, science education research. In Sect. 3, we discuss a number of categorisation schemes that have been developed as tools to analyse various aspects of computing education literature. In Sects. 4 and 5 we look at collected evidence of the use of theoretical frameworks and methodological approaches. We continue in Sect. 6 by looking at findings of analyses which have focused on specific publication venues. Finally, we discuss what these findings holistically report on the development of CER, concluding with some recommendations.

While each of the review papers in our data pool covers a limited span of years, perhaps focusing on certain aspects of papers in a small number of publication venues, together these reviews cover a very significant share of papers published in central CER publication venues during the past 20 years, as CER has emerged as an independent field. They thus provide an interesting additional perspective to complement the scientometric analyses presented elsewhere in this book.

## 2 Emergence of CER as an Independent Field

Discussion of the need for more rigorous research in computing education gained wider attention in the early 2000s. Fincher and Petre published their seminal book, Computer Science Education Research [18], in 2004. The book discussed the nature and scope of work in the field and presented a comprehensive tutorial for conducting empirical research in computing education, with a number of case studies illuminating different types of research in the area. The Koli Calling conference (Koli), which had been launched in 2001, took steps towards becoming an international research conference in 2004. The International Computing Education Research workshop (ICER) was launched in 2005, clearly laying out in its call for papers what would be expected from submissions as research papers: a clear theoretical

basis building on literature, a strong empirical basis, drawing on relevant research methods, and explicitly explaining how the paper contributes to existing knowledge in computing education. At the end of the decade, the Journal of Educational Resources in Computing (JERIC), which had published many experience reports, curricular and course descriptions, and learning resources in various subareas of computing, transformed into ACM Transactions on Computing Education (TOCE), which sought to publish high quality research, and has joined the longer-standing Computer Science Education (CSE) as a premier journal in the field.

It is likely that several factors contributed to this increasing interest in research quality. While experiences of various pedagogical approaches and observations of their impact are highly valuable for teaching practitioners seeking to develop their teaching, their value as evidence of the impact is unclear. The quality of data collection and analysis matters, as does the chain of inference behind the findings. If computing education researchers wish colleagues in other disciplines to acknowledge that they are undertaking serious research, publication quality matters. Computing education research is by its very nature an interdisciplinary field which draws on several disciplines, especially education, psychology, and the computing sciences. Thus, comparison with work in these areas is also to be expected.

Moreover, PhD theses, which are evaluated based on academic standards, have to pass the evaluation process in faculties, schools, or departments where academics from other disciplines have an opportunity to assess their quality. High-quality publications are also needed for appointment and promotion in academic positions. Finally, many academic institutions explicitly specify how teaching quality is evaluated and how relevant it is when making decisions on academic promotions, and publication of research papers in education-related conferences and journals has thus become stronger evidence for candidates.

There were consequently a growing number of people who enjoyed carrying out computing education research and for whom the question of quality was very obviously relevant, and these people had to start considering structures and resources that would support quality improvement not only in their own work but also more broadly in the computing education community. As most computing educators teach their classes and carry out their main research in specific topic areas of computing itself, specific venues were needed where the focus of work would be the pedagogical methods and tools in teaching and learning computing across the board, as well as studies on various aspects of students, recruitment, retention, etc.

This development fits well with Fensham's analysis on the development of science education as an independent field of research [17]. Fensham identified two sets of criteria for an independent research field. *Structural criteria* include conferences and journals dedicated to publishing work in the area, established professorships in the area, professional associations, and centres for research and organised research training. All of these elements have existed in CER for a considerable time, but they are not relevant for this chapter, so we shall not discuss them further.

*Intra-research criteria* focus on the content of the work. Fensham defined seven criteria for evaluating how a field is conducting research. The first criterion,

*scientific knowledge*, refers to basic knowledge and skills in the field, without which studies cannot be carried out. CER certainly has a massive literature covering several decades of work, which cannot be ignored when building new knowledge. A closely related criterion, *asking questions*, means that the discipline asks questions that other disciplines do not. For example, CER has directed much focus on how students learn programming, and how this learning can be supported by the application of various pedagogical approaches and tools.

The third criterion, *conceptual and theoretical development*, is of particular interest to us. This development is embodied, for example, in the extensive empirical work that researchers have carried out to investigate the impact of various pedagogical innovations on students' learning outcomes and study practices. These works often build on theoretical constructs from educational sciences and psychology, such as self-efficacy [5], fixed/growth mindset [16], and goal orientation [25], seeking to identify relevant factors and their roles in students' learning outcomes, motivation, and study practices. Numerous statistical models have been built to describe these complex relationships, as reviewed and described in the work of Malmi et al. [37, 38]. Moreover, computing education researchers have invested a great deal of work in qualitative analyses to investigate students' conceptions of different computing concepts, resulting in numerous phenomenographical outcome spaces [9, 77, 78] and grounded theories [28, 88]. Computing education researchers have also developed various taxonomies and classification schemas to analyse the relationships and structures among computing concepts, tools, or pedagogical methods [26, 31, 44, 52].

A closely related criterion, *research methodologies*, describes how the field has adopted methods used in other fields and adapted them to its specific needs, as well as developing its own methods within the field itself. For example, CER has adapted several general psychological instruments to measure such matters as students' attitudes and self-efficacy in computing or programming [14, 30, 80]. Moreover, the field has developed several concept inventories to analyse students' conceptions and misconceptions, for example in programming and in data structures and algorithms [12, 42].

The fifth criterion, *progression*, describes how the field builds on previous work to further accumulate scientific knowledge. While obviously all scientific papers (should) incorporate literature reviews, genuine progression is more difficult to measure. Many research publications cite related work without actually building on it in any way [37]. On the other hand, there are streams of research that have systematically explored a single area over a long period, such as Guzdial's studies on media computation [21], the long-term research around the Jeliot program visualisation tool [7], and research in developing BlueJ and analysing data collected with it [10, 29].

As part of this progression, some publications become *seminal* or *model publications*, opening new avenues for research, and therefore become widely cited (criteria 6 and 7). As examples, Hundhausen in his meta-study on algorithm visualisation [24] identified students' interaction with the visualisation as an important factor for learning; an ITiCSE working group report from 2002 [48]

built on this observation and proposed the engagement taxonomy as a hypothesis of the impact of various engagement methods, and this taxonomy led to substantial research on evaluating the impact of visualisations [82]. Another example is the idea of programming puzzles, where students construct a program by ordering a given set of program statements into a working program [53]. These—often called Parsons puzzles—have been further developed and extended in many ways and have led to substantial empirical work [15].

## 3 Classifying Papers

Research publications can be classified in many ways beyond that involving scientometric publication metadata. We can categorise papers based on the research topics addressed, such as topical areas in a curriculum, on the pedagogical techniques covered in the paper, on the perspectives or properties of stakeholders (students, teachers, the organisation...) [27, 36, 66], or on the educational level of the target group (primary, secondary, or tertiary) [60]. We can classify papers based on their type, such as empirical research papers, experience reports, learning resources, tool descriptions, or opinion pieces [83], or based on whether their main purpose is to describe, evaluate, or formulate something such as concepts, activities, resources, novel methods etc. [84]. We can also look at various aspects of the documented research process, such as what theoretical frameworks have been used, if any, or what research methodologies, data collection, and analysis methods have been used [41, 60]. In this section, we discuss some categorisation schemes that are most relevant to our goal of addressing research processes in CER publications, as well as the nature of their contributions.

Simon [66, 71] presented a systematic categorisation scheme comprising four dimensions. The *Context* dimension describes the curricular context in which the research is carried out, such as programming, data structures and algorithms, security, operating systems, etc. If there is no such context, the paper is classified as broad-based. *Theme* (originally Topic) categorises what the research is about: for example, teaching and learning techniques, teaching and learning tools, assessment techniques, ability and aptitude, etc. *Scope* describes the breadth of community involvement in the research, which might be a single subject (course), a program of study, a whole institution, or more than one institution. Finally, *Nature* seeks to differentiate types of paper. A *study* paper reports a research question, the data collection and analysis used to address it, and the findings. An *experiment* paper does the same, but in a manner that would typically be recognised as at least a quasi-experiment. An *analysis* paper investigates existing data such as course or program results or literature. A *report* describes something, such as a new pedagogical approach, learning tool, or learning resource, which has been implemented and possibly used in practice, perhaps with some initial experiences of using it. Finally, a *position/proposal* paper presents the authors' beliefs on a particular matter or a proposal for work yet to be done. Simon considers the first three categories to be

*research papers*, contrasting them with the other types of papers. Simon's system has been used to analyse papers in multiple publication venues [66–68, 71].

Malmi et al. [41] developed a classification scheme focusing on theoretical constructs, research goals, and various aspects of research process. Their scheme has several dimensions, building on and extending the work by Vessey et al. [84], who had classified computing research literature in several dimensions. From their work, Malmi et al adopted the dimensions of reference discipline, research approach, and research method and extended the classification scheme into seven dimensions: (1) the theories, models, theoretical frameworks or instruments that had been used; (2) the disciplines from which these had been adopted; (3) the technologies or tools that were used and reported in the work; (4) the general purpose of the work: descriptive, evaluative, or formulative, each with several subcategories; (5) the overall research framework that had been used in the empirical work; (6) the data sources that had been used; and (7) the data analysis methods that had been used. The Malmi et al. classification scheme has been used in a couple of extensive analyses of the computing education literature [40, 41].

Randolph [60] analysed a wide selection of CER literature using a broad scheme, which addressed authors, reporting elements, research topic or content, and a comprehensive analysis of research designs, collected data, and generated evidence.

In the following sections, we analyse what previous studies have revealed about the overall state and trends of CER publications in different venues, beginning with reviews of the theoretical background to work and of the methods used in research, and proceeding to survey some work analysing publications from specific venues. We appreciate that there are many reviews that focus on specific topic areas in computing, such as recursion [43] or event-driven programming [35], on specific educational tools, such as program visualisation [73] or algorithm visualisation [64], or on pedagogical approaches, such as pair programming [11, 81]. However, we do not discuss these reviews, as their perspective is too narrow for a consideration of the overall development of CER as a research field.

On the other hand, reviews that focus on analysing the research presented in specific venues are relevant for this chapter, as they reflect on the development of the CER community's preferences and approaches in research and illuminate the holistic development of the field. Moreover, they complement the scientometric analysis results presented in other chapters of this book.

## 4 Theoretical Development in CER

In educational research there is a symbiosis between theory and practice. Purely academic theory without connection to practice is unhelpful to practitioners, whereas pure description of the practice without a framing theory—or an aim to develop a more abstract model or theory of the practice—limits the researchers' and practitioners' potential to see beyond immediate empirical observations. This symbiosis between theory and practice is crystallised by Kant: "theory without

practice is empty; practice without theory is blind" (cited by Morrison and Werf [47]). In addition to their potential to help us improve our understanding of the practice, theories have a more immediate practical affordance for researchers. Theories may guide the research process in many ways. Theories can point to relevant phenomena to study and suitable methods with which to study them, and can provide a distinctive viewpoint from which the data can be interpreted [61, 62]. Finally, conceptual and theoretical development of a research field is one of the criteria used by Fensham [17] to describe and evaluate development of science education. The use of existing theories and the development of field-specific theories are signs of a mature research field. In the past decade, there has been an increasing interest in understanding how and to what degree CER is maturing as a research field. This has resulted in several publications that aim to describe and map out what, how, and based on which theories computing education has developed as a research field.

These recent CER papers have approached the definition of theory from three viewpoints. Szabo et al. [74, p. 92] define theory by stating what a theory is: "we use an inclusive definition of theory as a generalisation, abstraction, explanation or prediction of a phenomenon". Malmi et al. [40, p. 29] approach the definition of theory from another point of view by emphasising what theories provide or enable us to do: "we define 'theory' to mean a broad class of concepts that aim to provide a structure for conceptual explanations or established practice". Finally, a later definition used by Malmi et al. [37, p. 188] focuses on the quality of the process through which theory is formed: "we defined the concept *theoretical construct* as a theory, model, framework, or instrument developed through application of some rigorous empirical or theoretical approach". Combining these three approaches, we could define theory—or theoretical construct—as

- a structure for conceptual explanations or established practice, a generalisation, abstraction, explanation, or prediction of a phenomenon
- that is developed through application of some rigorous empirical or theoretical approach
- and can be expressed in the form of a model, framework, or instrument.[1]

One sign of CER as a maturing research field is that increasing numbers of papers report the use of theories to guide their research. Malmi et al. [40] found on the basis of data from 2005 to 2011 that just over half of the papers published in CSE, JERIC/TOCE, and ICER explicitly used theories. Often the theories used are borrowed from other fields. The studies by Malmi et al. [40] and Szabo et al. [74] both investigated which theories CER is building on. The findings of these two studies suggest that CER is heavily borrowing theories from other fields, especially from education and psychology. The findings of Szabo et al. [74] suggest that, for

[1] While instruments are generally considered methodological tools to measure something, we take here the perspective that they are theory-informed constructs which support the implementation of some specific theory or theories in research.

example, flow theory, chunking theory, and learning styles were much referenced theories of learning in CER publications. The motivation for borrowing theories from outside the field of CER is readily understandable as many phenomena in the teaching and learning of computing can be explained through non-subject-specific theories such as expectancy-value theory [86].

However, there has been a rising interest in understanding what field-specific theories or field-specifically tuned versions of more generic learning-related theories have been developed within the CER community. Malmi et al. [37, 38] studied which field-specific theories, models, and instruments had been developed within the preceding 10–15 years. The results suggest that the CER community has developed theories of its own, but that new CER-specific theories are still rare. Only 12% of the papers published in ICER, CSE, and TOCE in 2005–2015 proposed a new CER-specific theory. An example of one of the most cited new CER-specific theories in learning is Lopez et al.'s hierarchical model of programming skills [34].

A recent study by Malmi et al. [39] shows that the great majority (65%) of the new CER-specific theories are developed as a result of quantitative research. Qualitative research designs and a combination of literature analysis and argumentation are each used to develop new CER-theories in almost one third of cases. The same study classifies the papers according to the main purpose of the developed theories [20]. The results reveal that only a fraction (5%) of the newly developed CER-specific theories aim at prediction or design and action. The purpose of the great majority of the new CER-specific theories is analysis (rich description of the phenomena), explanation (rich description with explanations but no predictions), and explanation and prediction (predictions and causal explanations).

In light of the recent surveys focusing on theories, there are signs of CER maturing as a research field. Many research publications are based on some existing theory or are presenting a new field-specific theory that the authors have developed. The purpose of many new CE-specific theories is rich description and explanation of what, why, and when something happens; but few publications are able to propose theories that are capable of offering a basis for predictions. This distribution of different kinds of newly developed theory makes sense for a relatively young research field, as we need to begin by gaining a wide understanding of the nature of the phenomena, interactions, and processes in different settings in computing education. As Gregor [20] suggests, analysis theories are the basis of all other types of theory. It takes time for the knowledge to accumulate enough to inform further theoretical development.

Finally, there is one aspect that may hold back the further theoretical development of CER. The recent studies suggest that the community is not building widely on the published theoretical contributions. For instance, over 90% of papers just briefly describe the theoretical construct from the paper they cite, not using the construct or developing it further in their own paper [37, 38]. This same trend, of new theoretical constructs not being cited, used to inform further research, or further developed by others, was also noted in a recent study that analysed 85 papers on six broad topics from ICER, CSEd, and TOCE published between 2005 and 2020 [39].

## 5 Methodological Development in CER

One specific class of papers identifiable within the body of CER meta-analyses is that of methodological review. An often-cited measure of maturity for an emergent research area is the strength of the methodological underpinning for its research activities and the corresponding judgement of what is considered acceptable for publication. Metareviews that take a structured approach to analysing publications in the field and the methods they employ can contribute to our understanding of areas of strength and weakness in research practice and of trends and developments over time. Literature reviews generally focus on particular topics or themes within a subject area. A methodological review is concerned not with the specific subject-related outcomes that are reported but with how the work has been conducted, what research practices are reported, and on what basis the conclusions have been drawn.

A 2004 review by Valentine [83] that covered 444 SIGCSE papers focusing on CS1 published between 1984 and 2003 is often cited as an early exploration of the research approaches used in computing education. This is not specifically a methodological review (and its own methodological basis has been questioned [66]) but it nevertheless provides an interesting snapshot of CER publications in one particular venue. Valentine distinguished between experimental papers that employ some type of "scientific analysis" (either qualitative or quantitative, but somewhat loosely defined) and those that do not (which are purely descriptive or discursive and which he assigned to one of five other classifications). Valentine found that 94 papers (21%) could be classed as experimental. His conclusions challenged authors to go a step further in their work, not just reporting but also evidencing outcomes.

At about the same time as this challenge, seminal work in CER methodological review was being carried out by Randolph et al., later published in Randolph's 2007 PhD thesis [57] and in several related studies conducted using a similar approach [56, 58–60]. Randolph's thesis is the first detailed review of CER methodology that has a broad coverage and that itself employs a carefully articulated methodological approach. The novelty of the work was demonstrated by the findings of the thorough literature review in Randolph's thesis, which identified just three previous CER methodological reviews: two involving his own work [58, 59], and the third being Valentine's paper [83]. In the thesis, a stratified random sample of 352 papers was drawn from 1306 articles published between 2000 and 2008 in the CER publications of five conferences (SIGCSE Technical Symposium, Innovation and Technology in Computer Science Education (ITiCSE), ICER, Australasian Computing Education Conference (ACE), and Koli Calling) and three journals (Computer Science Education, SIGCSE Bulletin, and Journal of Computer Science Education Online). The overarching purpose of the research was to determine the methodological characteristics of the articles surveyed. This objective was broken down into specific sub-questions to determine the proportion of articles with human participants and the types of method they involved, the measures and instruments they used, the independent/dependent/mediating variables used, and the characteristics of the paper's structure; the types and proportions of articles not

involving human participants; the types of research design used in articles taking an experimental approach; and the statistical practices used in articles taking a quantitative approach.

Randolph's analysis was approached from a behavioural sciences perspective using both quantitative and qualitative methods to investigate how (and why) individuals and groups act. This influences the structure of the research questions and explains the differentiation between articles involving human participants and those addressing other types of data. Randolph takes the reasonable stance that when conducting research on human participants "the conventions, standards and practices of behavioural approach should apply". The work looks most closely at papers of this type, which constitute roughly a third of the papers sampled.

The findings of Randolph's work are presented with detailed breakdowns across many variables and comparisons. However, the data overall present an interesting picture regarding the structure, methodology, and reporting in CER articles published between 2000 and 2005 through the lens of a number of accepted measures of robust research. Of the articles without human participants, over 60% were purely descriptive reports of interventions, innovations, etc. Nearly 40% of papers involving human participants provided only anecdotal evidence for the claims made. Of those that were evidenced, nearly two thirds used (quasi-)experimental (quantitative) approaches and over a quarter employed an explanatory descriptive (qualitative) methodology. Within the (quasi-)experimental group, roughly half used a single group post-test method only. That is, there was no pre-test and no control group or comparison. Without such measures, an analysis is much less likely to accurately discover a causal relationship. Dependent variables related most frequently to attitude (60%), with attainment the second most prevalent (52%).

Randolph also noted other issues with research design, such as inadequate amounts of data and results not supporting the conclusions drawn. Questionnaires were the most common measurement instrument, but measures of validity or reliability were given in only one article. Issues were also observed with presentation across all papers in the sample. For example, over 28% did not review existing literature; 78% did not state research questions or hypotheses; and over 63% did not state the purpose of the work.

Randolph's work also evidenced some development of methodology during 5 years spanned by the sampled articles. The proportion of papers with claims backed by purely anecdotal evidence decreased consistently from 58% in 2000 to 27% in 2005. While this may be seen as a welcome indication that published results were becoming more evidence-based, other apparent trends, such as a seeming decline in the use of qualitative research methods, are harder to interpret.

Randolph concluded from his findings that CER was at a crossroads. The high proportion of papers failing to adopt robust research practices could be viewed as helpful in generating hypotheses but not in confirming them. Either the current approach could continue or (as might be considered a mark of developing maturity in a research area) the balance could shift towards more application of rigorous methods. The challenge for CER researchers was clear. Randolph's data and analysis provide a good baseline against which to compare future findings.

However, the behavioural science perspective of the work means that quantitative methods are given greater attention, and baseline information on the robustness of qualitative methods used in CER is lacking [33]. Further, this division does not allow for consideration of mixed methods approaches, whose combination of different perspectives has the potential to deliver rigour through triangulation and insight. This point was taken up by Thota et al. [79], who document a structure for "paradigm pluralism" and present a case study of its application in a computing education research project.

Aspects of methodology have also formed part of the classification criteria used in other meta-studies. In 2010, Malmi et al. [41] sought to establish a broader picture of the kind of work being carried out in CER against a categorisation scheme with seven aspects—theory/model/framework/instrument (TMFI), technology/tool, reference discipline, research purpose, research framework, data source, and analysis method. Several of these dimensions clearly relate to the methodology used. The sample consisted of all 79 ICER papers published over the first 5 years of the conference. This work set out not to critique the application of the methods (for example, to consider control variables used or validation of instruments) but to map the field more generally. It was found that the approach was descriptive in 11% of papers, evaluative in 71%, and formulative (of a novel concept, model etc.) in 18%. Sixty percent of the sample used one or more TMFI, 86% provided some kind of empirical evaluation, and 79% used an identifiable research framework (most commonly a survey). Although the sample is from just one conference, these findings indicate a significant reduction in purely anecdotal evidence. The authors note that the ICER requirements for papers make stipulations on the theoretical and empirical basis of the work and research design. Findings such as these and those of Simon [67] suggest that conference and journal policy plays a major role in promoting a shift towards more rigorous approaches.

Survey instruments and quantitative analysis remain the two main ways in which data is collected and analysed in CER, and several metareviews have considered the ways in which these have been developed and applied. Margulieux et al. conducted a review of 197 CER papers to determine the variables and instruments used and the analysis performed [42]. In this case, the sample consisted of all papers with empirical evaluation involving human participants published in CSE, TOCE, or ICER from 2014 to 2017. The authors note the benefit of standardisation where this is possible, to allow reuse of validated instruments and greater comparability between different studies. It was found that 37 standard instruments had indeed been used, but noted the lack of standardised CER survey instruments for further commonly assessed variables, such as perceptions of the computing field. Overall the authors find that the proportion of papers assessing indicators of learning in CER is in line with that observed in general educational research. However, as with Randolph's earlier findings, they note that reporting of data collection and analysis often falls short of commonly accepted good practice. For example, 51% of the papers surveyed did not provide information on learner characteristics and nearly 15% did not state the number of participants. Recommendations of aspects of reporting to improve are set out by McGill and Decker [45].

Where survey instruments are used, the level of confidence in results obtained is related to the reliability (consistency of measurement) and validity (assurance that survey measures the intended constructs) of the instrument itself. In a study of 297 CER papers published from 2012 to 2016 across a number of venues, Decker and McGill found that of the 47 different instruments used, 94% were obtainable (either published or available on request), 60% provided some measure of reliability, and 51% provided some rationale for validity (although this was often by expert opinion of face validity) [12]. However, the findings of Heckman et al. [23] paint a less positive picture. They observed that most of a sample of 427 papers from five major CER venues in 2014 and 2015 did not publish the survey questions used. In some cases, there may be good reason for this, such as space restrictions in publications or the desire to prevent future survey participants becoming familiar with the questions in advance. However, this has led to the same concept being measured by multiple different instruments and a tendency to 'reinvent the wheel' rather than the community making use of standard instruments that allow comparison of results and for which reliability and validity could be established [12]. One useful resource for CER researchers is the CER database (https://csedresearch.org), which includes (in September 2022) a collection of 140 computing-focused instruments (as well as many others) that are classified by topic and can be reused, However, while information on the reliability and validity of these instruments would be a useful guide for researchers, it is not provided.

Where quantitative analysis is performed, the robustness of the work reported also depends not just on using (particular types of) statistical tests but on the appropriateness and correct application of those tests. Further, for both quantitative and qualitative analysis, full information needs to be reported to the reader in order for the method to be transparent and the work to be replicable. Randolph's work showed a baseline position that allowed considerable scope for improvement in these areas. More recently, research by Sanders et al. [63] looked at the use and reporting of inferential statistics in all 270 ICER papers published between 2005 and 2018. They found that 51% of the sample provided statistical analysis beyond purely descriptive, while 28% either had no data or did not describe it numerically. The authors describe the reporting as "not encouraging", with very few papers giving sufficient information to adequately describe the analysis. Amongst many deficiencies noted, hypotheses were rarely stated, the precise variant of the statistical test was not given, and there was no indication that the assumptions for the test were met. Sadly, a cautious comparison (given the different venues surveyed) with Randolph's work from over 10 years earlier showed no improvement in this regard. It seems that when it comes to the robust application of statistical methods, CER has not advanced. This picture is confirmed by the findings of Heckman et al. [23]. While over 82% of the papers published in five major CER venues in 2014 and 2015 were found to provide some empirical evidence, norms for reporting were often not being met, and only a quarter of the papers reported survey results in a manner that was strongly replicable.

Other comparisons with Randolph's work come to similar conclusions. Lishinski et al. [32] reviewed papers published in CSE and ICER from 2012 to 2015. Of 136

papers found, 110 were judged to be empirical in nature and only eight of these relied on purely anecdotal evidence. However, no significant move was observed towards more rigorous qualitative and quantitative approaches. Results did not suggest any difference in methodological rigour between the journal and conference publications.

These methodological metareviews conducted on publications now spanning more than 20 years show some developments, but also point to other areas where little progress is observed. It is certainly difficult to make direct comparisons, given the different venues from which the samples of papers are drawn and the different interpretations of classifications such as 'experimental'. However, the broad picture shows that CER has moved away from publications with purely anecdotal evidence, largely meeting Valentine's challenge to make the extra effort in providing evidence. Journal and conference policies appear to have had a considerable influence in this. On the other hand, repeated studies show that the design methods used are at the weaker end of the spectrum (such as post-test only) and the analysis reported still falls short of expectations in many respects.

## 6 Analyses of CER Publication Venues

With maturity comes the potential for introspection. Literature from key publication venues for a discipline provides lenses through which to assess the evolution of the discipline and possible future directions.

A number of metareviews have focused on analysing the publications from one or two computing education venues. We found reviews of five prominent computing education conferences (SIGCSE TS, ITiCSE, ACE, NACCQ (a New Zealand conference), and Koli Calling) and the journal Informatics in Education (InfEdu). These reviews typically classify papers published at the venues based on their general characteristics and addressed research topics, and investigate any trends over time. Sometimes the metareviews were timed to mark milestones of the particular venue [6, 70]. A summary of the metareviews we discuss is presented in Table 1.

We note that a number of premier publication venues, ICER, CSE, and TOCE, are not among these venue-specific reviews; we have not found any such reviews for them. It is of course conceivable that such reviews have been written, submitted to the venues, and not accepted for publication. As discussed in the previous sections, papers published in these three venues have been analysed in several reviews that have focused on the use and development of theoretical frameworks as well applied research methods. The five venue-specific reviews considered in this section focus on other aspects. A sixth review, on InfEdu, is included because the review considered it as a journal with strong parallels to the Koli Calling conference despite its broader coverage of topics, thus permitting some interesting comparison between the two venues.

**Table 1** Summary of venue-focused metareviews

| Venue | Acronym | Year of first conference/issue | Metareview time frame | Papers analysed |
|---|---|---|---|---|
| **Conferences** | | | | |
| Special Interest Group in Computer Science Education Technical Symposium | SIGCSE TS | 1970 | 1970–2018 | 481 |
| Innovation and Technology in Computer Science Education | ITiCSE | 1996 | 1996–2019 | 1295 |
| Australasian Computing Education Conference | ACE | 1996 | 2004–2006<br>1996–2008 | 129<br>328 |
| Koli Calling International Conference on Computing Education Research | Koli | 2001 | 2001–2006<br>2001–2007 | 102<br>130 |
| National Advisory Committee on Computing Qualifications (NZ) | NACCQ | 1988 | 2004–2006<br>2000–2007 | 46<br>157 |
| **Journal** | | | | |
| Informatics in Education | InfEdu | 2002 | 2002–2007 | 121 |

A series of metareviews of literature published in ACE, NACCQ, InfEdu, and Koli were reported from 2007 to 2009. The analyses of the literature at these venues were conducted using a classification scheme for computing education literature developed by Simon [66]. As mentioned in a preceding section, Simon's scheme comprises four dimensions (*context*, *theme*, *scope* and *nature*) and has subsequently been applied in a number of reviews of the computing education literature, particularly studies of venues.

The first review in this set is an early review of computing education papers published at the ACE and NACCQ conferences in 3 years from 2004 to 2006 [66]. The 175 papers (129 from ACE and 46 from NACCQ) gave insights into the focus of interest of researchers and practitioners in the Australasian computing education

community. A major outcome of this work was the development and trialling of Simon's scheme for classification of computing education literature.

A review by Simon et al. [71] extended this work but focused on the NACCQ conference. In this review, the 157 computing education papers published at NACCQ from 2000 to 2007 included 3 years from the Simon study [66]. As the analysis involved multiple classifiers, the inter-rater reliability of classifications was tested. A system where classifiers worked individually and then in pairs produced fair to good agreement for three dimensions (context, theme, and scope) and excellent agreement for the fourth (nature). Analysis of the NACCQ papers found that the most frequent themes of work reported were teaching, learning, and assessment, with 20% of papers having a theme of teaching/learning techniques, 10% with a theme of teaching/learning tools, and 10% concerning assessment techniques and tools. Curriculum was a strong feature of work at NACCQ, with 15% of papers having this theme. The most common context was programming (13%); however, a high number of papers (30%) had no specific context. In regard to the nature of the research, the most frequent type of paper was report (40%) and almost the same number of papers were classified as research (37%). Almost half the work (45%) was conducted in single courses. The longer time frame of this review allowed for analysis of trends. The main trend reported was for the nature of the work, with a decreasing proportion of report papers and a corresponding increase in research papers.

A similar review by Simon [69] again extended his previous work [66], this time focusing on the ACE conference. The review covered all 10 years of papers published at ACE from 1996 to 2008 (the conference was not held each year during this time), including 3 years from the previous study [66]. Overall, 328 papers were analysed using Simon's scheme. As with NACCQ, the most frequent themes concerned teaching, learning, and assessment; however, the percentages for ACE were higher in each case. At ACE, 34% of papers had a theme of teaching/learning techniques, 15% teaching/learning tools, and 12% assessment. The 11% of papers with a theme of curriculum was lower than was found at NACCQ. At ACE almost a third of the papers (32%) were in the context of programming, considerably higher than at NACCQ (13%). The nature and scope of papers at ACE showed different profiles from NACCQ, with ACE having a higher percentage of reports (70%) and a lower percentage of papers classified as research (23%). Almost two thirds of the work at ACE (64%) was conducted in single courses. The analysis of trends found an increasing proportion of research papers, from 10% in 1996 to nearly 50% in 2008, a similar but stronger trend to that at NACCQ [71].

A further review by Simon [67] applied his scheme to papers published at the Koli conference from 2001 to 2006. The 102 papers comprised the complete set of full papers published since the conference began. As with NACCQ and ACE, the most frequent themes at Koli concerned teaching, learning, and assessment. The profile was similar to that of ACE, with 28% of papers having a theme of teaching/learning techniques, 20% teaching/learning tools, and 13% assessment. The theme of curriculum accounted for 9% of the papers. At Koli almost a quarter of the papers were in the context of programming (25%). The nature and scope of

papers at Koli Calling showed a similar profile to NACCQ, with 47% classified as reports and 35% as research. Around half of the work at Koli (53%) was conducted in single courses. An analysis of trends in the nature of the papers showed increasing proportions of research papers, as was found in the NACCQ [71] and ACE [69] studies. For the first 3 years, research papers were 14%, 10%, and 7% respectively of the full papers published; this then jumped to 47%, 44% and 59% for the next 3 years, reflecting the change of the conference goals as reported in chapter "Computing Education Research in Finland" of this book.

Simon's scheme was also applied to a comparative analysis of all the papers published in the Informatics in Education journal (InfEdu), based in Lithuania, and the Koli conference, held in Finland [68]. The analysis comprised 121 papers in six volumes of InfEdu from 2002 to 2007 and 130 papers at Koli from 2001 to 2007. Although the goals of Informatics in Education are broader than those of Koli, encompassing the use of computers in all education rather than just in computing education, the report found many similarities between the themes and contexts of the papers. The most frequent themes in both were teaching/learning techniques (Koli 30%, InfEdu 26%), teaching/learning tools (Koli 19%, InfEdu 12%) and assessment (Koli 12%, InfEdu 13%). The main difference was that InfEdu had higher proportions of papers in the theme of educational technology. The prominent context for Koli was programming (37%), whereas broad-based contexts were most common at InfEdu (26)%, with programming the next most frequent (20%). The venues showed some difference in the nature of the papers, with Koli having 52% reports compared with 44% for InfEdu. The main differences with the scope were that Koli had more than half its papers reporting work in a single course (53%) whereas more than half the papers at InfEdu had no applicable scope (57%).

This series of metareviews between 2007 and 2009 provides a unique snapshot of these computing education venues over the preceding decade. There were many similarities between the venues. Each venue had a strong focus on work in the context of programming and themes relating to teaching and learning techniques or tools and assessment. Each venue had an increasing proportion of research papers. However, there were also key differences that indicated the particular focus of each venue; for example, NACCQ's focus on curriculum and InfEdu's focus on educational technology.

Recently there have been two larger reviews of leading computing education conferences. A review by Simon and Sheard [70] analysed 24 years of literature from ITiCSE. Motivated by the 25th anniversary of the conference, they applied Simon's scheme to analyse all full papers and working group reports published at ITiCSE from its inception in 1996 through to 2019. During this time there were 1295 full papers and 129 working group reports published at the conference. The analysis considered working group reports separately from full papers. Analysis of the papers shows that the most common context was programming (38%), with increasing focus on school computing. Teaching and learning techniques (28%) and tools (22%) were the focus of more than half the papers, but there has been an increasing focus on ability, aptitude, and understanding. More than half the papers (53%) report work done in a single course of study. The most common nature of

papers is report (46%), with study (29%) as the next most common. There has been steady growth in work with natures of analysis, study, and experiment.

The 50th year of the SIGCSE Technical Symposium inspired a review by Becker and Quille [6]. The review focused on full papers about introductory programming courses at the university level published at the symposium in the 49 years from 1970 to 2018. Using a systematic search and selection the authors built a list of 481 papers. The analysis examined the focus of each paper, using a framework of 54 subcategories in eight top-level categories: first languages and paradigms; CS1 design, structure and approach; CS1 content; tools; collaborative approaches; teaching; learning and assessment; and students. Trend analysis showed an increasing focus on students and learning and assessment and a decreasing focus on CS1 design, structure, and approach, and on first language and paradigms.

While these reviews use different approaches to examine the literature of different venues over different time frames, their findings do tend to show some common features:

- within computing education, programming education garners far more attention than education in any other topic area;
- there is broad evidence of a move away from experience reports toward generally empirical research;
- there is evidence of diminishing interest in some topics (such as what programming languages should be used) and increasing interest in others (such as how students learn).

The set of reviews of these key publication venues for computing education research has deepened our knowledge of the literature profiles of the different venues and our understanding of how research in this field has evolved.

## 7 Discussion

Overall, our analysis of various meta-analysis papers in CER builds a picture of a discipline that is growing in maturity and independence. Our work focuses strongly on the development during the past 20 years, because we found very few relevant meta-analysis papers addressing the field prior to 2000. Moreover, from our perspective, the last two decades are the most relevant years for the maturing field, as the numbers of published papers and submissions have increased substantially during this time. This is well reflected in the ICER conference. In its early years the acceptance rate of papers was around 40% and sometimes more, and during the past few years it has dropped to about 20%. At the same time, the number of participants has increased from around 50 to more than 200. The field is growing rapidly.

We have seen the proportion of research papers increasing in different venues, as discussed in Sect. 6. The growing interest in improving research quality has been clearly visible in the past few years, with the publication of more specific methodological reviews [12, 32, 42, 45, 63], as well as surveys of the use and devel-

opment of theoretical frameworks [37, 38, 40, 74]. These works are complemented by several chapters of the recent Cambridge Handbook of Computing Education Research [19].

The evidence collected in these reviews identifies increased and broadening use of theoretical frameworks from other disciplines, especially from education and psychology, as well as growing interest in developing domain-specific theoretical constructs. Moreover, there has been a clear transition from anecdotal data towards more versatile data collection and analysis. However, there is much room for improvement in methodological rigour. Considering these findings from the point of view of Fensham's criteria (those that are relevant to this chapter), we find that they support the claim that CER fulfills the criteria of "conceptual and theoretical development" and "research methodologies". Moreover, the findings also support partially fulfilling the criterion "progression", as methodological rigour forms a base for any replication studies and theoretical developments support the building of more complex theories and models. However, replication studies are still rare in the field and difficult to carry out [1]. Moreover, the examples of research where some previous theoretical construct is being extended or used to inform new research are still scarce, except for research that applies or further develops theory-informed instruments [39]. Validated instruments are increasingly used in the field, and new instruments are being developed and validated [37–39, 42].

Despite the general interest in research quality, there is still much scope for improving the rigour of research and how it is reported, as pointed out in Sect. 5. One reason for some of the commonly-noted shortcomings in reporting may be the limited space in papers. It is widely acknowledged that reporting qualitative research generally requires more space than reporting quantitative research. While this may be a factor in some cases, rigour is not related solely to the length of the publication. The challenges include how well the research design, data collection, and analysis methods have been documented in the paper, and how well the arguments for these choices have been reported. As discussed above, there are shortcomings in reporting why specific tests were used and whether their applicability for the specific cases was checked. There are shortcomings in describing the demography of human participants, which may be significant for understanding the specifics of the research context. In qualitative research, the process of how categories were formed from the data is sometimes described only vaguely. In content analyses including quantitative results, checking inter-rater reliability or other ways of confirming the classification of data items might not be reported and might not even have been conducted.

Overall, these shortcomings may undermine future research in several ways. While some results might not hold up under closer scrutiny, other researchers might nevertheless build on them in their own work. Moreover, while replication is a powerful tool for confirming and generalising findings, shortcomings in reporting research settings and methods can make replication difficult or even impossible. Furthermore, the results of meta-analyses that summarise findings from original research in are weakened, as it is more difficult to judge the reliability of the individual studies on which they are building.

Another source for these challenges in reporting research is shortcomings in researchers' own competences. Statistics is a broad area with numerous methods, and many computing education researchers are not well versed in it. Statistics is not widely needed in many areas of computing, so experts in those areas who undertake computing education research may have a lot to learn when using statistical analysis methods. Learning analytics is also increasingly used in the field, using data mining and machine learning methods and broadening the scope of quantitative methodological approaches even more.

A similar situation concerns qualitative methods, which can present even more challenges. They are used only in some subareas of computing, such as human-computer interaction and empirical software engineering. However, computing degree programs may have no compulsory courses on these methods. Most researchers naturally extend their methodological competences during their doctoral studies, but they are likely to focus only on the skills that are most relevant for their particular research.

There is also a wider context to the development of CER. It is still the case that, in some departments, computing education research is not accorded the same status as "real" computing research. In such an environment, academic staff may not receive recognition for work in CER and indeed may have to undertake such work outside their full academic workload. Similarly, in many countries, funding for CER projects and PhD studentships is very difficult to come by. These factors do not provide an ideal climate for a research area to flourish and it is to the credit of many academic staff that their commitment to developing the teaching of their subjects leads them to devote their own time to researching and publishing in the area.

Finally, theoretical frameworks in educational sciences and psychology are very rarely addressed in computing research proper, and thus people entering CER with a background in computing have much to learn. This is not limited to learning some specific theory; rather it concerns learning the whole research paradigm of the social sciences, where theoretical frameworks have a much stronger role than in computing. The whole concept of theory is different in the social sciences, when compared with theoretical computer science which builds on a mathematical research tradition with its emphasis on the proof of theorems.

There are several ways in which these challenges might be addressed to improve research quality. First, the organising bodies of conferences and journals have an important role to play in motivating progress. Calls for papers in conferences and instructions for authors in journals give researchers important guidance on how they can improve their work and their prospects of having their papers accepted. However, this is only one perspective. Another important perspective is the organisation of the review process and the competence of reviewers in giving adequate feedback and judgment of submitted papers.

In the past few years there has been a significant increase in both the number of submissions to CER venues and the number of participants in conferences. One factor behind this is the growth in publications addressing computing in schools. This has caused pressure to add more reviewers to the field, and the need to monitor

the quality of reviews. Leading conferences have revised their review processes by introducing senior program committee members or associate program chairs, whose main task is to encourage reviewers, after submitting their reviews, to discuss them with one another to resolve possible conflicting views or to clarify their arguments. The senior members will subsequently write metareviews, which not only summarise the main issues and arguments raised in the reviews to support the proposed acceptance or rejection, but also judge the significance of the issues based on the quality of reviewers' arguments and the metareviewer's own judgement as a senior researcher in the field. This helps program chairs to make their final decisions on acceptance, as well as improving the overall quality of reviews. These roles thus mirror the role of editorial board members in journals, who coordinate reviews of papers assigned to them by the editor-in-chief.

Another development is the clarification of review criteria based on feedback from reviewers and observations made by program chairs. It is vital for review quality that papers are assigned reviewers who know the topic area well and are familiar with the methods applied. Some conferences help in this regard by using a bidding phase, where reviewers can give their preferences for papers they are willing and competent to review. Moreover, conferences frequently tune their review criteria to clarify their interpretation both for reviewers and authors: this also helps to improve the quality of future submissions. An in-depth discussion of review practices in CER is presented by Petre et al. [55].

There are many other ways in which the research community can support its members to build their competence. One traditional practice is organising doctoral consortia for PhD students, which has been a regular practice in ICER, is somewhat less frequent in Koli Calling, and has recently been adopted by ITiCSE. In Lithuania, Vilnius University conducts an extended doctoral consortium annually in Druskininkai, which is targeted to STEM education and educational technology PhD students and has frequently had CER PhD students and senior CER researchers participating. In Europe, the annual SEFI engineering education conference organises doctoral consortia which many CER PhD students have attended.

Another community activity is organising narrow workshops or tutorials around selected methodological themes. An early example was the PhiCER (phenomenography in computing education research) workshop, which was organised twice in 2006–2007 and helped many researchers to learn this method [8]. ICER has frequently supported pre- or post-conference events on various themes. In addition, its work-in-progress workshops allow researchers to present ongoing research and get feedback and support from others. Recently significant initiatives have been the establishment of CSEdResearch.org, which especially supports K-12 level computing education, and the CSEdGrad.org project for supporting PhD students in the field. The former, for example, includes among its resources more than 200 instruments that can be used in research.

While all of the elements above are laudable initiatives helping to attain the goals of improving research quality, there remains room for critique. There is some tension between the requirements for rigorous research and the presentation

of innovations in the field, as pointed out by Nelson and Ko [49]. Theories are obviously strong assets for guiding research, but there is a risk that undue emphasis on theory may restrict the design of new educational innovations, as not all development and research in computing education is guided by existing theories. New domain-specific theories may emerge from the analysis of existing settings and collected data with an open mind rather than relying solely on existing theoretical lenses. These opportunities should not be ignored. Moreover, if all papers must include a rigorous empirical evaluation of new pedagogical innovations, learning tools, or learning resources, this may inhibit the presentation of innovations to the computing education audience. Overall, the work carried out in the field is targeted at improving computing education practice, and there should be space for early presentation of ongoing work. For example, the Koli Calling conference currently has two tracks for papers: research papers can be long, and have strict requirements; while short papers, often called discussion papers, provide an opportunity to report new developments with only preliminary results, as well as opinion pieces that discuss relevant themes supported only by argumentation. The CER field is rich and it is worthwhile to make this richness visible.

## 8 Recommendations

We conclude by briefly summarising the main recommendations that we have collected from the meta-analyses addressed in previous sections.

- There are as yet few theories targeted at designing new educational activities [39]. More work, as exemplified by Nikula et al. [50] and Xie et al. [87], is needed.
- When theoretical frameworks are cited in papers, their actual role in research design and in analysis and interpretation of results must be made clear. There is currently a reporting problem in that theoretical frameworks and how they are applied in the research are not clearly identified with citations in papers. This follows in part from the practice of considering theories only as related work.
- Research questions and/or hypotheses should be properly reported, as should the goals or purpose of the study itself.
- There are many shortcomings in reporting research, which should be addressed. Shortcomings in the following make replication of studies difficult or impossible and undermine interpretation of the results.
  - Contextual information—where the study was carried out—should where appropriate provide relevant information such as course syllabus, required prerequisite information, course requirements and schedule, and grading principles.
  - Participant demographics, their background, and recruitment practice or incentives should be reported accurately.
  - When using questionnaires, preference should be given to validated instruments. When new questionnaires are designed, some evidence of their

reliability and validity should be reported. The survey questions should ideally be made available for readers.
  - Reasons should be given for selecting the analysis methods, such as statistical tests, their variations, and their applicability in the setting.
  - In qualitative settings, the methods used for building categories should be explained.

- Publication venues have an important role in supporting research quality in terms of the instructions they provide for authors, the page limits, and improving the quality of the review process and the competency of reviewers.
- Various research training activities, such as pre- or post-conference workshops and doctoral consortia, can play an important role in supporting and further developing the competencies of community members.

## References

1. Ahadi, A., Hellas, A., Ihantola, P., Korhonen, A., Petersen, A.: Replication in computing education research: researcher attitudes and experiences. In: Proceedings of the 16th Koli calling international conference on computing education research, pp. 2–11 (2016)
2. Alaqsam, A., Ghabban, F., Ameerbakhsh, O., Alfadli, I., Fayez, A.: Current trends in online programming languages learning tools: a systematic literature review. Journal of Software Engineering and Applications **14**(7), 277–297 (2021)
3. Atchison, W.F., Conte, S.D., Hamblen, J.W., Hull, T.E., Keenan, T.A., Kehl, W.B., McCluskey, E.J., Navarro, S.O., Rheinboldt, W.C., Schweppe, E.J., Viavant, W., Young, D.M.: Curriculum 68: recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science. Communications of the ACM **11**(3), 151–197 (1968)
4. Austing, R.H., Barnes, B.H., Bonnette, D.T., Engel, G.L., Stokes, G.: Curriculum recommendations for the undergraduate program in computer science: a working report of the ACM committee on curriculum in computer sciences. ACM SIGCSE Bulletin **9**(2), 1–16 (1977)
5. Bandura, A.: Self-efficacy mechanism in human agency. American Psychologist **37**(2), 122 (1982)
6. Becker, B.A., Quille, K.: 50 years of CS1 at SIGCSE: a review of the evolution of introductory programming education research. In: 50th Technical Symposium on Computer Science Education, pp. 338–344 (2019)
7. Ben-Ari, M., Bednarik, R., Levy, R.B.B., Ebel, G., Moreno, A., Myller, N., Sutinen, E.: A decade of research and development on program animation: the Jeliot experience. Journal of Visual Languages & Computing **22**(5), 375–384 (2011)
8. Berglund, A., Box, I., Eckerdal, A., Lister, R., Pears, A.: Learning educational research methods through collaborative research: the PhICER initiative. In: Tenth Australasian Computing Education Conference, pp. 35–42 (2008)
9. Boustedt, J.: Students' understanding of the concept of interface in a situated context. Computer Science Education **19**(1), 15–36 (2009)
10. Brown, N.C., Altadmri, A., Sentance, S., Kölling, M.: Blackbox, five years on: an evaluation of a large-scale programming data collection project. In: 14th International Computing Education Research Conference, pp. 196–204 (2018)
11. Chahal, K.K., Kaur, A., Saini, M.: Empirical studies on using pair programming as a pedagogical tool in higher education courses: a systematic literature review. Research and Evidence in Software Engineering, pp. 251–286 (2021)

12. Decker, A., McGill, M.M.: A topical review of evaluation instruments for computing education. In: 50th Technical Symposium on Computer Science Education, pp. 558–564 (2019)
13. Denning, P.J., Tedre, M.: Computational Thinking. MIT Press (2019)
14. Dorn, B., Elliott Tew, A.: Empirical validation and application of the computing attitudes survey. Computer Science Education **25**(1), 1–36 (2015)
15. Du, Y., Luxton-Reilly, A., Denny, P.: A review of research on parsons problems. In: Proceedings of the Twenty-Second Australasian Computing Education Conference, pp. 195–202 (2020)
16. Dweck, C.S.: Self-theories: their role in motivation, personality, and development. Psychology Press (2013)
17. Fensham, P.J.: Defining an Identity: The Evolution of Science Education as a Field of Research. Springer Science & Business Media (2004)
18. Fincher, S., Petre, M.: Computer Science Education Research. CRC Press (2004)
19. Fincher, S.A., Robins, A.V.: The Cambridge Handbook of Computing Education Research. Cambridge University Press (2019)
20. Gregor, S.: The nature of theory in information systems. MIS Quarterly, pp. 611–642 (2006)
21. Guzdial, M.: Exploring hypotheses about media computation. In: Ninth International Computing Education Research Conference, pp. 19–26 (2013)
22. Guzdial, M., du Boulay, B.: The history of computing. The Cambridge Handbook of Computing Education Research (2019) **11** (2019)
23. Heckman, S., Carver, J.C., Sherriff, M., Al-Zubidy, A.: A systematic literature review of empiricism and norms of reporting in computing education research literature. ACM Transactions on Computing Education **22**(1), 1–46 (2021)
24. Hundhausen, C.D., Douglas, S.A., Stasko, J.T.: A meta-study of algorithm visualization effectiveness. Journal of Visual Languages & Computing **13**(3), 259–290 (2002)
25. Kaplan, A., Maehr, M.L.: The contributions and prospects of goal orientation theory. Educational Psychology Review **19**(2), 141–184 (2007)
26. Kelleher, C., Pausch, R.: Lowering the barriers to programming: a taxonomy of programming environments and languages for novice programmers. Computing Surveys (CSUR) **37**(2), 83–137 (2005)
27. Kinnunen, P., Meisalo, V., Malmi, L.: Have we missed something? Identifying missing types of research in computing education. In: Sixth International Computing Education Research Workshop, pp. 13–22 (2010)
28. Kinnunen, P., Simon, B.: My program is ok – am I? Computing freshmen's experiences of doing programming assignments. Computer Science Education **22**(1), 1–28 (2012)
29. Kölling, M., Quig, B., Patterson, A., Rosenberg, J.: The BlueJ system and its pedagogy. Computer Science Education **13**(4), 249–268 (2003)
30. Kong, S.C., Chiu, M.M., Lai, M.: A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. Computers & Education **127**, 178–189 (2018)
31. Lewis, C.M.: Exploring variation in students' correct traces of linear recursion. In: Tenth International Computing Education Research Conference, pp. 67–74 (2014)
32. Lishinski, A., Good, J., Sands, P., Yadav, A.: Methodological rigor and theoretical foundations of CS education research. In: 12th International Computing Education Research Conference, pp. 161–169 (2016)
33. Lister, R.: The Randolph thesis: CSEd research at the crossroads. SIGCSE Bulletin **39**(4), 16–18 (2007)
34. Lopez, M., Whalley, J., Robbins, P., Lister, R.: Relationships between reading, tracing and writing skills in introductory programming. In: Proceedings of the Fourth International Workshop on Computing Education Research, pp. 101–112 (2008)
35. Lukkarinen, A., Malmi, L., Haaranen, L.: Event-driven programming in programming education: a mapping review. ACM Transactions on Computing Education **21**(1), 1–31 (2021)
36. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory programming: a systematic literature review. In: ITiCSE 2018 Working Group Reports, pp. 55–106 (2018)

37. Malmi, L., Sheard, J., Kinnunen, P., Simon, Sinclair, J.: Computing education theories: what are they and how are they used? In: 15th International Computing Education Research Conference, pp. 187–197 (2019)
38. Malmi, L., Sheard, J., Kinnunen, P., Simon, Sinclair, J.: Theories and models of emotions, attitudes, and self-efficacy in the context of programming education. In: 16th International Computing Education Research Conference, p. 36–47 (2020)
39. Malmi, L., Sheard, J., Kinnunen, P., Simon, Sinclair, J.: Development and use of domain-specific learning theories, models and instruments in computing education. ACM Transactions on Computing Education, **23**(1), Article 6, pp. 1–48 (2023)
40. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Kinnunen, P., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Theoretical underpinnings of computing education research: what is the evidence? In: Tenth International Computing Education Research Conference, pp. 27–34 (2014)
41. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Characterizing research in computing education: a preliminary analysis of the literature. In: Sixth International Computing Education Research Workshop, pp. 3–12 (2010)
42. Margulieux, L., Ketenci, T.A., Decker, A.: Review of measurements used in computing education research and suggestions for increasing standardization. Computer Science Education **29**(1), 49–78 (2019)
43. McCauley, R., Grissom, S., Fitzgerald, S., Murphy, L.: Teaching and learning recursive programming: a review of the research literature. Computer Science Education **25**(1), 37–66 (2015)
44. McGill, M.M., Decker, A.: Construction of a taxonomy for tools, languages, and environments across computing education. In: 16th International Computing Education Research Conference, pp. 124–135 (2020)
45. McGill, M.M., Decker, A.: A gap analysis of statistical data reporting in K-12 computing education research: recommendations for improvement. In: 51st Technical Symposium on Computer Science Education, pp. 591–597 (2020)
46. McGill, M.M., Decker, A.: Tools, languages, and environments used in primary and secondary computing education. In: 25th Conference on Innovation and Technology in Computer Science Education, pp. 103–109 (2020)
47. Morrison, K., van der Werf, G.: Editorial. Educational Research and Evaluation **18**(5), 399–401 (2012)
48. Naps, T.L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., Velásquez-Iturbide, J.Á.: Exploring the role of visualization and engagement in computer science education. In: ITiCSE 2002 Working Group Reports, pp. 131–152 (2002)
49. Nelson, G.L., Ko, A.J.: On use of theory in computing education research. In: 14th International Computing Education Research Conference, pp. 31–39 (2018)
50. Nikula, U., Gotel, O., Kasurinen, J.: A motivation guided holistic rehabilitation of the first programming course. ACM Transactions on Computing Education **11**(4), 1–38 (2011)
51. Papert, S.A.: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books (2020)
52. Park, T.H., Saxena, A., Jagannath, S., Wiedenbeck, S., Forte, A.: Towards a taxonomy of errors in HTML and CSS. In: Ninth International Computing Education Research Conference, pp. 75–82 (2013)
53. Parsons, D., Haden, P.: Parson's programming puzzles: a fun and effective learning tool for first programming courses. In: Eighth Australasian Computing Education Conference, pp. 157–163 (2006)
54. Peng, J., Yuan, B., Spector, J.M., Wang, M.: Integrating technology in programming learning and instruction: a critical review. International Journal of Smart Technology and Learning **1**(4), 323–343 (2019)
55. Petre, M., Sanders, K., McCartney, R., Ahmadzadeh, M., Connolly, C., Hamouda, S., Harrington, B., Lumbroso, J., Maguire, J., Malmi, L., McGill, M.M., Vahrenhold, J.: Mapping the landscape of peer review in computing education research. In: ITiCSE 2020 Working Group Reports, pp. 173–209 (2020)

56. Randolph, J., Bednarik, R., Silander, P., Gonzalez, J., Myller, N., Sutinen, E.: A critical analysis of the research methodologies reported in the full papers of the proceedings of ICALT 2004. In: Fifth International Conference on Advanced Learning Technologies, pp. 10–14 (2005)
57. Randolph, J.J.: Computer science education research at the crossroads: a methodological review of computer science education research, 2000–2005. Utah State University (2007)
58. Randolph, J.J.: A methodological review of the program evaluations in K-12 computer science education. Informatics in Education **7**(2), 237–258 (2008)
59. Randolph, J.J., Bednarik, R., Myller, N.: A methodological review of the articles published in the proceedings of Koli Calling 2001-2004. In: Fifth Finnish/Baltic Sea Conference on Computer Science Education, pp. 103–109 (2005)
60. Randolph, J.J., Julnes, G., Sutinen, E., Lehman, S.: A methodological review of computer science education research. Journal of Information Technology Education: Research **7**(1), 135–162 (2008)
61. Reeves, S., Albert, M., Kuper, A., Hodges, B.D.: Why use theories in qualitative research? British Medical Journal **337** (2008)
62. Research Council of Norway: The Role of Theory in Educational Research – Report from the March Seminar 2011 (2012)
63. Sanders, K., Sheard, J., Becker, B.A., Eckerdal, A., Hamouda, S., Simon: Inferential statistics in computing education research: a methodological review. In: 15th International Computing Education Research Conference, pp. 177–185 (2019)
64. Shaffer, C.A., Cooper, M.L., Alon, A.J.D., Akbar, M., Stewart, M., Ponce, S., Edwards, S.H.: Algorithm visualization: the state of the field. ACM Transactions on Computing Education **10**(3), 1–22 (2010)
65. Sim, T.Y., Lau, S.L.: Online tools to support novice programming: a systematic review. In: Second Conference on e-Learning, e-Management and e-Services (IC3e), pp. 91–96 (2018)
66. Simon: A classification of recent Australasian computing education publications. Computer Science Education **17**(3), 155–169 (2007)
67. Simon: Koli Calling comes of age: an analysis. In: Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), pp. 119–126 (2008)
68. Simon: Informatics in Education and Koli Calling: a comparative analysis. Informatics in Education **8**(1), 101–114 (2009)
69. Simon: Ten years of the Australasian Computing Education Conference. In: 11th Australasian Computing Education Conference, p. 157–164. AUS (2009)
70. Simon, Sheard, J.: Twenty-four years of ITiCSE papers. In: 25th Conference on Innovation and Technology in Computer Science Education, p. 5–11. Association for Computing Machinery, New York, NY, USA (2020)
71. Simon, S., Sheard, J., Carbone, A., De Raadt, M., Hamilton, M., Lister, R., Thompson, E.: Eight years of computing education papers at NACCQ. National Advisory Committee on Computing Qualifications (2008)
72. Soloway, E., Ehrlich, K.: Empirical studies of programming knowledge. IEEE Transactions on Software Engineering (5), 595–609 (1984)
73. Sorva, J., Karavirta, V., Malmi, L.: A review of generic program visualization systems for introductory programming education. ACM Transactions on Computing Education **13**(4), 1–64 (2013)
74. Szabo, C., Falkner, N., Petersen, A., Bort, H., Cunningham, K., Donaldson, P., Hellas, A., Robinson, J., Sheard, J.: Review and use of learning theories within computer science education research: primer for researchers and practitioners. In: ITiCSE 2019 Working Group Reports, pp. 89–109 (2019)
75. Tedre, M., Simon, Malmi, L.: Changing aims of computing education: a historical survey. Computer Science Education **28**(2), 158–186 (2018)
76. Tedre, M., Sutinen, E.: Three traditions of computing: what educators should know. Computer Science Education **18**(3), 153–170 (2008)
77. Thomas, L., Eckerdal, A., McCartney, R., Moström, J.E., Sanders, K., Zander, C.: Graduating students' designs: through a phenomenographic lens. In: Tenth International Computing Education Research Conference, pp. 91–98 (2014)

78. Thompson, E., Kinshuk: The nature of an object-oriented program: how do practitioners understand the nature of what they are creating? Computer Science Education **21**(3), 269–287 (2011)
79. Thota, N., Berglund, A., Clear, T.: Illustration of paradigm pluralism in computing education research. In: 14th Australasian Computing Education Conference (2012)
80. Tsai, M.J., Wang, C.Y., Hsu, P.F.: Developing the computer programming self-efficacy scale for computer literacy education. Journal of Educational Computing Research **56**(8), 1345–1360 (2019)
81. Umapathy, K., Ritzhaupt, A.D.: A meta-analysis of pair-programming in computer programming courses: implications for educational practice. ACM Transactions on Computing Education **17**(4), 1–13 (2017)
82. Urquiza-Fuentes, J., Velázquez-Iturbide, J.A.: Pedagogical effectiveness of engagement levels–a survey of successful experiences. Electronic Notes in Theoretical Computer Science **224**, 169–178 (2009)
83. Valentine, D.W.: CS educational research: a meta-analysis of SIGCSE Technical Symposium proceedings. ACM SIGCSE Bulletin **36**(1), 255–259 (2004)
84. Vessey, I., Ramesh, V., Glass, R.L.: A unified classification system for research in the computing disciplines. Information and Software Technology **47**(4), 245–255 (2005)
85. Weinberg, G.M.: The Psychology of Computer Programming. Van Nostrand Reinhold New York (1971)
86. Wigfield, A., Eccles, J.S.: Expectancy-value theory of achievement motivation. Contemporary Educational Psychology **25**(1), 68–81 (2000)
87. Xie, B., Loksa, D., Nelson, G.L., Davidson, M.J., Dong, D., Kwik, H., Tan, A.H., Hwa, L., Li, M., Ko, A.J.: A theory of instruction for introductory programming skills. Computer Science Education **29**(2-3), 205–253 (2019)
88. Yuen, T.T., Robbins, K.A.: A qualitative study of students' computational thinking skills in a data-driven computing class. ACM Transactions on Computing Education **14**(4), 1–19 (2014)

# Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research


**Sonsoles López-Pernas, Mohammed Saqr, and Mikko Apiola**

## 1 Introduction

Science productivity has grown steadily over time and is expected to continue to do so as the number of publishers, journals, scientific events, and disciplines is on the rise [1]. Today's scientists are faced with an exponentially larger number of papers and less time to read given the accumulating academic duties. The growth of literature has greatly outpaced scientists' capacity to read the relevant literature or follow the latest developments in their fields [2]. Such accelerated growth has also impacted librarians, policymakers, students, and the public at large and, therefore, scientometrics was conceptualized to help understand, map, and summarize scientific research as well as evaluate scientists, institutions, or science productivity in general [3, 4].

Scientometrics is a quantitative method for the study of "science of science" and science communication at large [4–6]. Scientometrics as a field is deeply interdisciplinary at the intersection of philosophy, history, mathematics, sociology, information sciences, and statistics [3]. Bibliometrics, a closely related field, which often overlaps with scientometrics, is commonly defined as "the application of mathematics and statistical [methods] to books and other media of communication" [7, 8]. Today, bibliometric methods are increasingly used to study the literature's bibliographic data. Within the context of this book and the methods implemented here, bibliometric methods will be used within the larger more encompassing scope of scientometrics that encompasses bibliometrics, but also extends to the epistemology, structure, process, interrelationships, and dynamics of computing education research in our case [7–9].

S. López-Pernas (✉) · M. Saqr · M. Apiola
University of Eastern Finland, Joensuu, Finland
e-mail: sonsoles.lopez@uef.fi; mohammed.saqr@uef.fi; mikko.apiola@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research* ,
https://doi.org/10.1007/978-3-031-25336-2_5

Traditional methods of literature evaluation such as expert or peer review have several shortcomings. First, the peer-review process is time-consuming, slow, and costly. Second, the peer-review process is subject to human bias and distortion and lacks transparency [5]. On the other hand, scientometrics provides a cost-effective, more objective, and informative mode of analysis [4, 10]. An evaluation of peer-review in Italy's national research assessment found that scientometric methods are preferable to peer-review. The authors concluded that scientometrics "would allow much better, cheaper and more frequent national research assessments" [10]. In fact, the results of bibliometric analyses have been found to be correlated with other indicators of research quality, such as peer review or scientific awards [11, 12]. For research synthesis, systematic reviews—the gold standard for scientific evidence—are always performed; while immensely useful, they are time and resource exhaustive, focus on specific research questions, and limited in scope to a selected number of papers. However, it should be noted that scientometric methods are complementary to both methods, i.e., peer review for research evaluation and literature review for research synthesis. Although scientometric analysis offers a cost-effective, objective, data-driven method to look at research production, a proper scientometric analysis cannot be complete without a nuanced qualitative view inspired by the other methods [5].

The main aim of scientometrics is to *measure* science, map scientific impact, create indicators of productivity, as well as offer guidance to policy and management [5]. While scientometrics relies to a large extent on a diverse set of literature meta-data (i.e., bibliographic information), citations remain the central most important piece of information that drives most scientometric indicators. When an article cites another, a linkage is created between the two articles, bridging the authors, scientific concepts, publications, and even fields. Citing also creates a temporal continuity that builds on past ideas to create modern knowledge [5, 13]. A wide array of networks make use of such linkage. For instance, citations of articles are used to build co-citation networks or keyword networks. Citations are also used to build the Hirsch index (or simply H-index), one of the most important—yet very controversial—scientific indicators [5, 14].

## 2 Networks

In scientometrics, there is a long tradition of using networks to study collaboration among authors, institutions, or countries [4, 15, 16]. Networks enable researchers to map relationships, interactions, and connections between networked elements [16]. Similarly, researchers have also used networks to find similarities between papers, references, and publication venues. The power of networks affords researchers a rich toolset of visualization, mathematical analysis of relationships as well as algorithms for finding patterns of relationships, or groups of frequent interactions or relationships, i.e., communities [17]. Several methods exist for the definition of relationships, and each result in different network configurations, mathematical

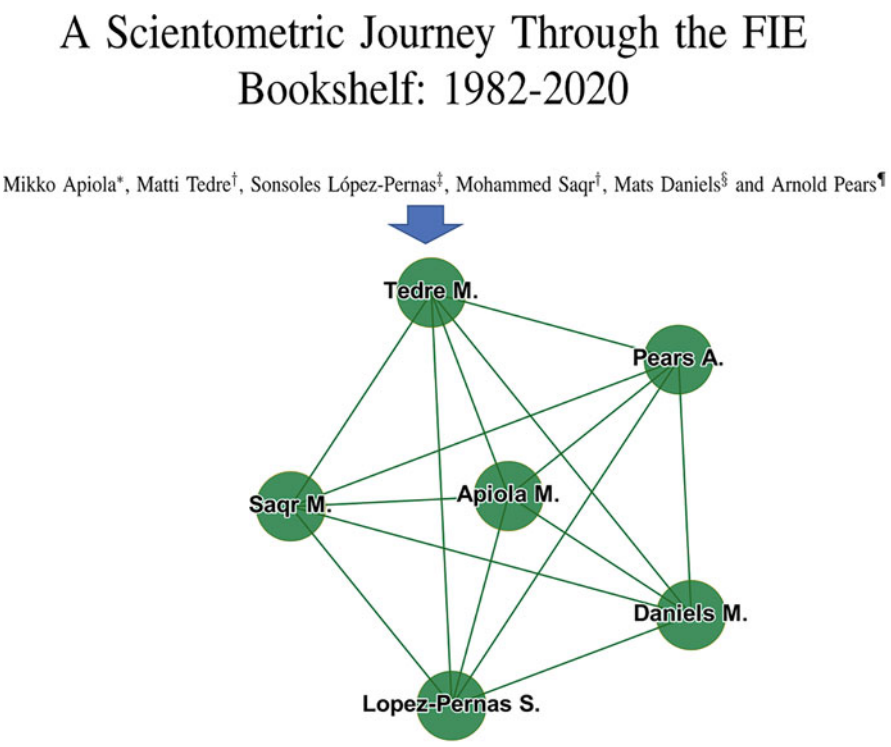


**Fig. 1** A co-authorship network of a single article constructed by considering each author connected to all other co-authors

parameters, and relationships. In this section, we start by offering a simplified overview of the basic methods for constructing a network. The following paper which was published in IEEE Frontiers in Education Conference had six authors, every author sharing a "co-authorship relationship with every other author" [18]. A network of co-authorship could be constructed by considering every author connected to every other author in the paper similar to Fig. 1. In the authorship network visualization, every author (often referred to as a node in networks terminology) is represented as a circle, and every relationship (often referred to as edge in networks terminology) is represented as a line connecting the two related authors.

A network of several papers is constructed in the same way, i.e., by aggregating all the relationships between the co-authors of the given papers. For instance, Fig. 2 shows a network created from two papers [1, 18]. An edge (a relationship) between every pair of co-authors is established and the authors who are shared between the two papers as a bridge between the authors of the two papers.

- Apiola, M., Tedre, M., López-Pernas, S., Saqr, M., Daniels, M., & Pears, A. (2021, October). A Scientometric Journey Through the FIE Bookshelf: 1982–2020. In *2021 IEEE Frontiers in Education Conference (FIE)* (pp. 1–9). IEEE.

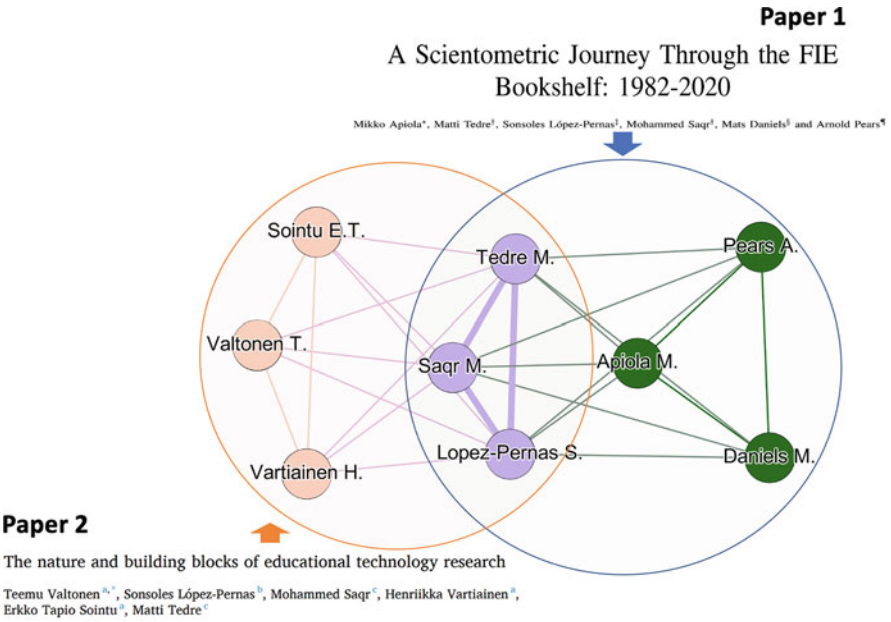


**Fig. 2** A co-authorship network constructed from two articles by considering each author connected to all other co-authors

- Valtonen, T., López-Pernas, S., Saqr, M., Vartiainen, H., Sointu, E. T., & Tedre, M. (2022). The nature and building blocks of educational technology research. *Computers in Human Behavior, 128*, 107123.

The networks presented in the previous examples are called co-authorship networks and are constructed by considering every author connected to all other co-authors in the same paper. Yet, the question is how to weigh every connection between co-authors. Several methods exist, the simplest of which is to weigh every connection equally, a method commonly referred to as full counting. Nonetheless, papers with more authors will have an advantage over papers with fewer authors, leading to inflation of their degree of connectivity [4, 17, 19]. For instance, in a full counting network created from a paper with six authors (e.g., Fig. 1), each author will have five connections. While in a paper with only two authors, (e.g., [20]), each author will have just one connection. Inflating the metrics of papers with higher number of authors results in biased and skewed inferences and therefore several methods were developed to balance the credit assigned to authors. A well-established alternative is to fractionally allocate the credit of authorship between authors, i.e., divide credit by the number of authors, a method commonly referred to as *fractional counting*. For instance, in the paper in Fig. 1, each of the six authors receives 1/6th of the credit i.e., 0.166 [4, 17]. There is fair evidence that fractional counting methods are preferable in constructing networks of countries, institutions and co-authorships

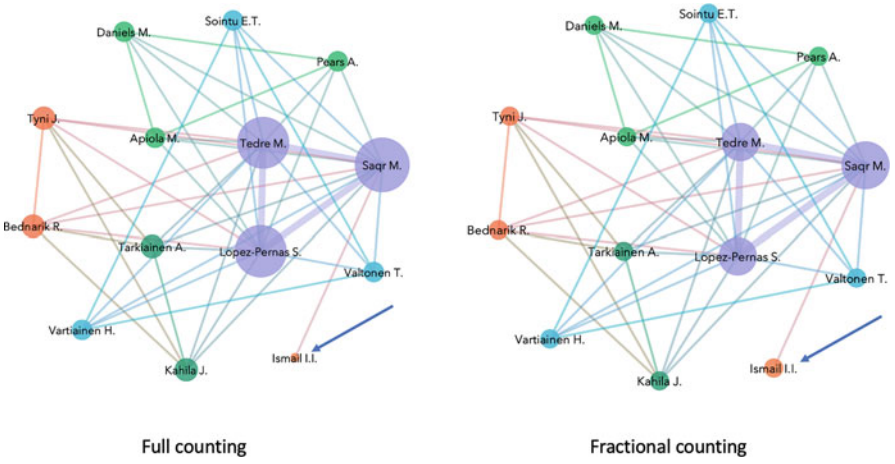


**Fig. 3** Comparison between full-counting and fractional counting co-authorship networks. In a full counting network, the more co-authors, the higher the degree (left). Node "Ismail I.I." (pointed at with an arrow) is used as a reference to explain full vs. fractional counting in the text

[15, 17]. Figure 3 shows a comparison between a network created from four papers using full counting and using fractional counting [1, 18, 20, 21]:

- Valtonen T, López-Pernas S, Saqr M, Vartiainen H, Sointu ET, Tedre M (2022) The nature and building blocks of educational technology research. *Computers in Human Behavior* 128:107123
- Apiola M, Tedre M, López-Pernas S, Saqr M, Daniels M, Pears A (2021) A Scientometric Journey Through the FIE Bookshelf: 1982–2020. In*: Proceedings of the 2021 IEEE Frontiers in Education Conference (FIE)*
- Ismail II, Saqr M (2022) A Quantitative Synthesis of Eight Decades of Global Multiple Sclerosis Research Using Bibliometrics. *Frontiers in Neurology* 13:845539
- Tyni J, Tarkiainen A, López-Pernas S, Saqr M, Kahila J, Bednarik R, Tedre M (2022) Games and Rewards: A Scientometric Study of Rewards in Educational and Serious Games. *IEEE Access* 10:31578–31585

The network was configured so that authors with a higher degree (number of co-authors) have a larger node size. The node labeled "Ismail II" has the smallest size in the full counting network with a degree of 1, although he is a co-author in a paper with only another author. Other authors of multi-authored papers had a higher degree and larger node size, e.g., Daniels M has a degree of 5. In other words, full counting awards more credit to the authors of papers with more authors, which is contrary to common sense: that authors of papers with less authors deserve more credit. The fractional counting network (right side), shows a more balanced allocation of credit where Ismail II has comparable size to other nodes.

## 3 The Philosophy of the Methods Followed in This Chapter

It is naive to assume that quantitative metrics can be read in isolation. There is a wide agreement that the quantitative evaluation of science should be supported by a qualitative approach with expert assessment [5]. Assessment of research should be viewed within the mission and context of the institution or researcher. Some research may not be globally influential, yet, it could be important to local communities, marginalized minorities, or contributing to novel areas [22]. Some research areas or disciplines attract more attention than others, and therefore, "filed norms" should be taken into account. The Leiden Manifesto for research metrics provides a great discussion on such issues and readers are encouraged to consult [22].

Understanding computing education research requires a multidisciplinary approach that combines scientometric methods with the latest advances in statistics, visualization, and network science as well as a nuanced qualitative review of the literature. Our previous scientometric research (e.g., [1, 16, 23, 24]), literature review (e.g., [22]), and interaction with several experts in the field have helped build our strategy for data cleaning and analysis on five main principles:

- **Data accuracy**: Bibliometric data are far from accurate with several problems regarding, e.g., keywords, author names, institutions, affiliations and conference names. We perform exhaustive steps to manually select each article included in the analysis, clean each field using state-of-the-art data processing methods as well as manual cleaning, verification, and quality assessment by multiple researchers to reach a consensus on the accuracy of the results and the methods.
- **Qualitative synthesis**: A nuanced qualitative synthesis has been applied when necessary in all of our analyses and interpretation of the quantitative results. We avoid making judgments, ranking, or being tempted by the numbers as concrete truth but rather use numbers as a guide to understanding the complex realities of science [22].
- **State-of-the-art analysis**: We use a large array of interdisciplinary methods that come from different fields which include visualization, statistical methods, network science, as well as modern bibliometric analysis.
- **Accurate, simple, and transparent indicators**: We use several methods for our analysis that we make sure are straightforward, transparent, and accurate. To ensure that our analysis is sound, we allowed several researchers who are involved in the editorial process to review and audit our analysis including, e.g., scientific output, collaboration networks, and historical data concerning conferences. For instance, almost 20 researchers were consulted about their position in the collaboration networks and their connections, their opinion helped us choose the best algorithm.
- **Priority for relevance, legitimacy, context, and expert judgment:** While metrics can be tempting to compute, apply or visualize, we have used qualitative evaluation where relevant, and verifiable by experts, which tells an important part of the story.

Therefore, the methods described in this chapter, and applied in the remaining chapters of the book in which bibliometric results are presented, apply these principles as operationalized in a five-step process of data collection, processing, analysis, and verification.

## 4 Methods

A five-step framework was followed in this chapter (Fig. 4):

1. **Data retrieval**: The first step includes searching the appropriate database retrieval of the metadata, and verifying the integrity of the retrieved data. By "appropriate" database, we mean one that has the best coverage of the subject matter, clear indexing criteria, and rich and consistent metadata.
2. **Screening for eligibility**: Search queries may include noise (i.e., data that are not relevant to the intended search) and therefore manual analysis may be needed to make sure that only the relevant records retrieved from the search are included in the analysis.
3. **Data pre-processing**: Bibliometric data is far from perfect as publications differ from one another in the way they record the fields in the database. Therefore, it is necessary to disambiguate the authors, institutions, as well as venues of publication. Keywords may differ vastly among publications, which require processing using custom dictionaries, cleaning, or combining.
4. **Data analysis**: The goal of analyzing bibliometric data is often gaining insights into a particular field of research. Some common analyses are author productivity, topic trends, geographical distribution, and top cited articles. Networks have a crucial role in mapping author collaboration (or collaboration between countries or institutions), co-citation of articles, or co-occurrence of keywords.
5. **Integrity check**: A group of experts verifies the results of data analysis and make sure that the findings align with the status of the field.

### *4.1 Data Retrieval*

The first step in the process was to obtain the metadata of all CER research that are relevant, comprehensive, noise-free and up-to-date. We followed the PRISMA-S (Preferred Reporting Items for Systematic reviews and Meta-Analyses) literature search extension [25] in order to find the relevant literature on computing education research. The outline of the whole searching, screening and cleaning process can be seen in Fig. 4. We performed the search on the Scopus database on 24th of January 2022. In addition to including almost all the venues that the Web of Science database includes, Scopus offers a larger coverage of conferences and journals relevant to our study [26]. Combining several sources is possible but it adds many difficulties to

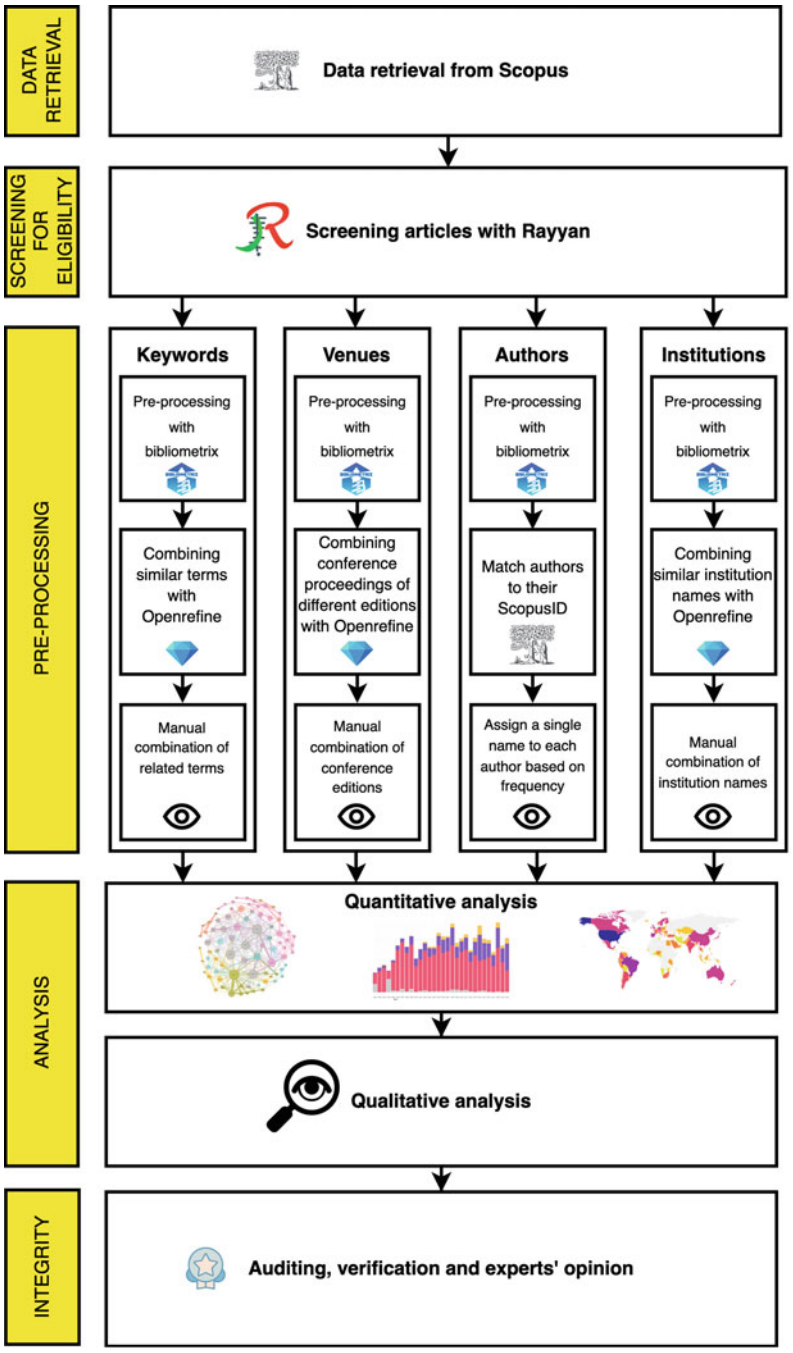


**Fig. 4** A five-step framework for data retrieval and analysis

the whole pre-processing and analysis, since metadata are coded in different ways in different databases, and there are big differences in the number of citations as each database indexes different publications. Besides its wide coverage of CER, we chose the Scopus database since it is well-maintained and has a rigorous quality assurance procedure for the indexed scientific journals or conferences [27, 28]. Moreover, Scopus was selected over the more "modern" databases (e.g., Dimensions or Lens), since the inclusion and coverage criteria in these databases are not as well-documented as in Scopus.

Two sources of data were retrieved: (1) all research published in venues dedicated to CER research that exclusively publish CER, and (2) research published in other venues (non-dedicated) which was obtained through a keyword search query.

1. **Dedicated venues**: The first step in our data retrieval process was to extract the data from the dedicated venues (conferences and journals) that are exclusively dedicated to computing education research. This approach is commonly used in bibliometrics analysis since it ensures a comprehensive coverage of the relevant articles with no noise (i.e., articles about other subjects) [1]. The dedicated list was compiled after consensus of seven researchers (the editorial team of this book). It included two major journals dedicated to computing education research:

   - Computer Science Education (CSE)
   - ACM Transactions on Computing Education (TOCE)
   - Journal on Educational Resources in Computing (JERIC), which was TOCE's former name

   The list of conferences included the following:

   - SIGCSE (Special Interest Group in Computer Science Education) Technical Symposium
   - Innovation and Technology in Computer Science Education (ITiCSE)
   - Koli Calling International Conference on Computing Education Research (Koli Calling)
   - International Computing Education Research conference (ICER)
   - Global Computing Education Conference (CompED)
   - Computer Science Education Research Conference (CSERC)
   - International Conference on Informatics in Schools (ISSEP)
   - Workshop in Primary and Secondary Computing Education (WiPSCE)

2. **Search terms**: The second step was to retrieve CER papers published in non-dedicated venues. Since computing education research lies in the intersection between education and computing research, authors often publish in venues that are specific to either of these fields, as well as in venues that belong to areas like engineering education or data science. We conducted a search that looked into the title, abstract, and author keywords of the articles using a search query that includes a combination of the terms "computing", "computer science", and "informatics" with the education-related terms: "education", "learn*", "teach*",

"curricul*", "course*", and "introductory". Some combinations resulted in substantial noise (i.e., unrelated articles) so we removed them from our search (e.g., "learning computing" brought many articles related to machine learning that were not about education). Three researchers met several times to agree on this set of search terms. The terms were also discussed with several other researchers and consensus was reached, after trying several combinations, that this combination retrieves the majority of relevant papers available. Yet, to make sure that every article in the dataset is relevant, a manual selection process was performed where each article was verified to belong to CER, which is explained below. The search query performed was as follows:

```
TITLE-ABS ( "COMPUTING EDUCATION" ) OR
AUTHKEY ( "COMPUTING EDUCATION" ) OR
AUTHKEY ( "COMPUTING LEARN*" ) OR
AUTHKEY ( "TEACHING COMPUTING" ) OR
AUTHKEY ( "COMPUTING TEACH*" ) OR
AUTHKEY ( "COMPUTING COURSE*" ) OR
AUTHKEY ( "COMPUTING CURRICUL*" ) OR
TITLE-ABS ( "COMPUTER SCIENCE EDUCATION" ) OR
TITLE-ABS ( "COMPUTER SCIENCE LEARN*" ) OR
TITLE-ABS ( "LEARNING COMPUTER SCIENCE" ) OR
TITLE-ABS ( "TEACHING COMPUTER SCIENCE" ) OR
TITLE-ABS ( "COMPUTER SCIENCE TEACH*" ) OR
TITLE-ABS ( "COMPUTER SCIENCE CURRICUL*" ) OR
TITLE-ABS ( "INTRODUCTORY COMPUTER SCIENCE" ) OR
AUTHKEY ( "COMPUTER SCIENCE EDUCATION" ) OR
AUTHKEY ( "COMPUTER SCIENCE LEARN*" ) OR
AUTHKEY ( "LEARNING COMPUTER SCIENCE" ) OR
AUTHKEY ( "TEACHING COMPUTER SCIENCE" ) OR
AUTHKEY ( "COMPUTER SCIENCE TEACH*" ) OR
AUTHKEY ( "COMPUTER SCIENCE COURSE*" ) OR
AUTHKEY ( "COMPUTER SCIENCE CURRICUL*" ) OR
AUTHKEY ( "INTRODUCTORY COMPUTER SCIENCE" ) OR
TITLE-ABS ( "INFORMATICS EDUCATION" ) OR
AUTHKEY ( "INFORMATICS EDUCATION" ) AND
NOT TITLE-ABS ( "HEALTH INFORMATIC*" ) AND
NOT AUTHKEY ( "HEALTH INFORMATIC*" )
```

This search resulted in 8935 articles, of which 3177 were repeated from the previous search, adding a total of 5758 new articles.

## 4.2 Screening Articles for Eligibility

The next step in our workflow (Fig. 4) was to manually screen the results from our search to ensure that every included article is relevant. Two of the authors manually screened all the 5758 records resulting from our search in non-dedicated venues, using Rayyan.ai (a free online service for conducting systematic literature reviews). Both authors first rated 898 items in common with a high interrater agreement (Cohen's kappa = 0.85). All disagreements were discussed until agreement was reached by the three authors. After that, the remaining articles were divided among the authors to screen. A total of 450 records were excluded after manual screening (7.8% of the articles screened).

An exploratory analysis of the data revealed that some articles are duplicates, and therefore, we removed duplicate articles that were indexed as different records in Scopus. For example, some venues published papers in their conference proceedings and in special issues from journals, e.g., SIGCSE Technical Symposium proceedings and ACM SIGCSE Bulletin. We further excluded all articles that were published in 2022 or later. Articles that were indexed as original articles, conference papers, or book chapters were included, while editorials, errata, reviews, notes, short surveys, and retracted papers were excluded as they do not report on research in findings. The final dataset included 16,863 records (see Fig. 5).

## 4.3 Pre-Processing Bibliometric Data

In a third step (Fig. 4), the bibliometric dataset was processed with the R library bibliometrix [29]. Bibliometrix is an open-source tool for performing comprehensive science mapping analysis. Bibliometrix is compatible with most of the main scientific databases (Web of Science, Scopus, Cochrane, Lens, Pubmed and Dimensions) as well as with generic BibTeX formatted files. It creates a bibliographic data frame with rows corresponding to publications and columns corresponding to the bibliographic fields provided by each database (authors' names, title, and keywords, among other information). All such elements constitute the bibliographic attributes of a document, also called metadata. Since articles' metadata are not consistent across publication venues, or even within the same publication, additional cleaning was required before analysis.

### 4.3.1 Keywords

Author keywords are chosen to best reflect the content of an article. Yet, there is no standard on how to select keywords for a manuscript. Authors often choose keywords freely or have to choose from a list of keywords provided by the journal or the conference. As a result, author keywords have grown to be inconsistent and

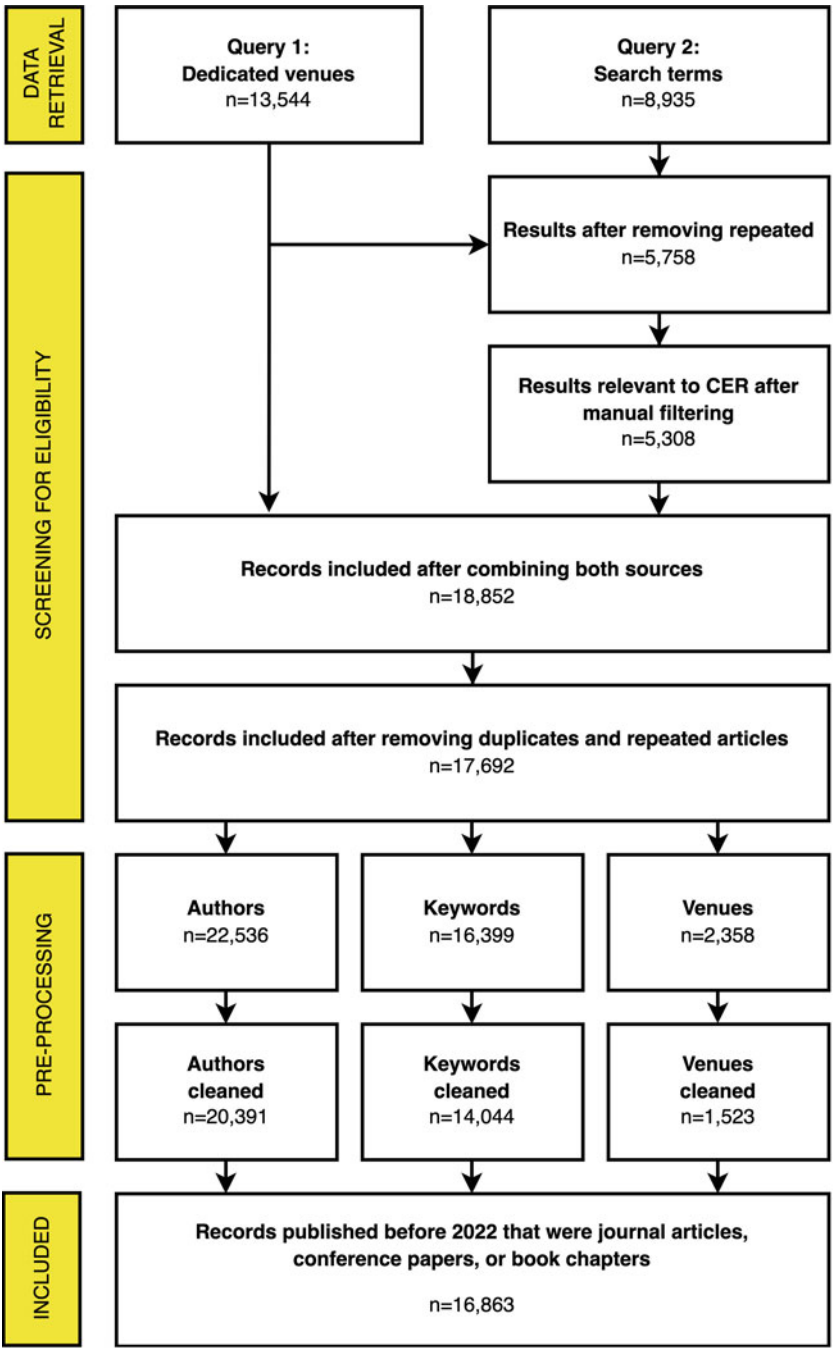


**Fig. 5** Outline of the CER metadata retrieval, screening and pre-processing process

require cleaning before they can be analyzed to extract publication themes and trends. Three stages of cleaning were applied:

1. **Keywords processing with bibliometrix software**: As a first step in the cleaning process we used the bibliometrix library which has a standard cleaning process. First, the library converts all keywords to uppercase to avoid considering the same keyword with different cases as different. Then, the library removes extra whitespaces. Lastly, the library removes special characters and non-printable characters which could make identical keywords look different.
2. **Cleaning with Openrefine**: In the second step, several algorithms are used in tandem to clean the keywords that are very similar to each other. For instance, different spellings of the same keyword, single and plural, keywords with hyphens, British and American spellings of the same keyword. Some examples are "computer" and "computers", "technology enhanced learning" and "technology-enhanced learning", "meta-analysis" and "meta analysis", and "visualization" and "visualisation". This process is performed using Google OpenRefine [30]: a free, open source application designed for cleaning of "messy" data. OpenRefine offers several Natural Language Processing (NLP) algorithms that help cluster together similar words based on different criteria such as spelling or pronunciation. Since different variations of the keywords could exist, the process is usually performed by using all the available NLP algorithms to search for the keywords [31]. The process is repeated a few times until no similar keywords are identified. A researcher has to accept the identified keywords as similar or reject the software suggestion. We used all of the following algorithms to identify similar keywords:
    - **Key collision algorithms**: These algorithms are based on the creation of an alternative representation of a value (a keyword in our case) that contains the most meaningful part. These methods include the fingerprint algorithm or the generic n-gram fingerprint. An example of two keywords that are suggested to be combined using these algorithms is "children and computers" and "computers and children"
    - **Nearest neighbor algorithms**: They detect words that differ from each other less than a given threshold or distance, e.g., Levenshtein and PPM (Prediction by Partial Matching). An example of a set of two words that are suggested to be combined using this method is "visualization" and "visualisation", as they only differ in one letter.
    - **Phonetic algorithms**: Phonetic algorithms detect when two words have similar pronunciation but different spellings. These are Metaphone3, Cologne-phonetic, Daitch-Mokotoff, and Beider-Morse. These algorithms often provide few useful suggestions that are not captured by other algorithms before, since words that are pronounced similarly are often also written similarly. Terms that should definitely not be combined are sometimes suggested such as "second year" and "secondary".
3. **Manual cleaning**: Clustering algorithms cannot identify that two keywords (e.g., "object-oriented programming" and OOP) are the same if they do not share

spelling or phonetic similarities. Moreover, there are different variations of the same keyword (i.e., synonyms). Therefore, a third step has to be performed using spreadsheet software to search for such possible keywords. Some examples of keywords that should be combined are "automated grading" and "automated assessment"; "massive open online courses" and "MOOCs"; "block-based programming" and "block coding". Since it is infeasible to perform such a step in every keyword, it was only performed with keywords that had a frequency of five or more.
4. **Stop words**: The last step was the removal of the stop keywords that were part of the search strings, or were essentially redundant, offering little distinction and the removal of which does have negative consequences on the results. Thus, we removed keywords such as "computer science education" or "education".

### 4.3.2 Venues

Publication venues are recorded in different fields which include "Source", "Conference.name", "Conference.code" and "ISSN". **Journal names** are (more often than not) consistent across the years: they are checked for completeness of data, and missing data are fixed. Yet, in our case, all journal data were consistent and required no further cleaning. **Conferences** are essentially inconsistent with lower quality regarding the proceedings' title and year of publication. For instance, the proceedings' title of the ITiCSE conference was "ITiCSE 2005: 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education" in 2005, while in 2021 it was "26th ACM Conference on Innovation and Technology in Computer Science Education, ITICSE 2021". Moreover, some proceedings are published with a generic title that encompasses many conferences (e.g., "ACM International Conference Proceeding Series" or "CEUR Workshop Proceedings"). To fix these names, we relied on triangulating information from additional fields (e.g., "Conference.name") which helped group articles from the same conference together. The final result was a unified field that had all journals and conferences from all versions under the same journal title or full name of the conference regardless of the year or proceedings' official title. Conference cleaning was performed with Openrefine using a process similar to the keyword cleaning, which was followed by manual checking and fixing of the instances that automated algorithms missed.

### 4.3.3 Authors

Scopus metadata includes the author list of each manuscript which allows to group together all papers by the same author. However, author names may vary throughout their career (e.g., the author Linda Grandell changed her name to Linda Mannila) or even the same name can appear in different ways: for example, when the author has a middle name (e.g., Peter Denning appears both as Peter Denning and as

Peter J. Denning) or when the name has special characters like accent marks or hyphens (e.g., J. Ángel Velázquez-Iturbide). Furthermore, the *authors* field in Scopus metadata only includes last name and initials. Thus, it is likely that different authors may have the same last name and initials despite not sharing the exact same name, which makes it impossible to distinguish them just by looking at the author name field. Nonetheless, Scopus metadata includes a Scopus ID field that allows to track author names that refer to the same person. To clean the author names from our dataset, we followed two steps:

1. We first used the bibliometrix library to process the author names. Bibliometrix unifies the format of the author names into a semicolon-separated list. Bibliometrix also removes extra whitespaces and non-printable characters.
2. Then, each author was matched to their corresponding Scopus ID using a custom script. The script checks the Scopus ID, verifies the integrity of the matching, and flags papers where matching fails. Cases where the match fails were manually checked and fixed. The most common causes were authors with a suffix (e.g., Jr. or III), where the suffix was wrongly identified as a separate author. For authors with more than one author name for the same Scopus ID, we kept only the most frequent spelling. For example, the author Peter Denning more frequently appeared as "DENNING PJ" rather than "DENNING P", so we used the former spelling for all his appearances in our dataset.

### 4.3.4 Institutions

Authors' affiliations (which are used to retrieve the institutions) are often inconsistent in articles' metadata, due to their many variations, given the different items that compose an affiliation (department, institution, address, etc.) and can be in any order, if at all present. To unify the affiliations in our dataset, we aimed at extracting only the main institution (mostly universities). This poses a great challenge since institutions can be written in their original form (e.g., "Helsingin yliopisto") or translated into English. Moreover, translations do not always conform to official English institution names (e.g. "University of Helsinki"), but are sometimes freely translated (e.g. "Helsinki University"). Some institutions even have different spellings of their own original name (e.g., by including the word "the" at the beginning: "(The) Open University of Israel"), may include the campus in their name (e.g. "UCLA" vs. "University of California"), or may have changed their name throughout their history (e.g., "University of Joensuu" vs. "University of Eastern Finland"). The institution cleaning was performed in a three-step process:

1. **Bibliometrix**: First, we used the bibliometrix library to extract authors' institutions from the Scopus affiliation field. Bibliometrix extracts the institution name in uppercase, removes special characters, and uses heuristics to clean the institution names. However, bibliometrix is far from perfect and a large number of inconsistencies remain.

2. **OpenRefine**: We then used OpenRefine to detect similar spellings of the same institution using the same algorithms presented in the Keywords cleaning section.
3. **Manual cleaning**: Lastly, we manually cleaned the remaining institutions that could not be fixed using the clustering algorithms.

### *4.4 Data Analysis*

The cleaned dataset was analyzed using the R statistical language with the bibliometrix package [29]. Bibliometrix offers an extensive toolset for the extraction, analysis, and visualization of bibliometric metadata (e.g., authors, keywords, citations, and countries). In addition to Bibliometrics, several R packages were used to plot and analyze the data. For correlations, comparisons across groups (ANOVA or Student's *t*-test), plots of statistical comparisons we used ggstatplot [32] (e.g., difference in keyword usage between journals and conferences [33]). For other plots (e.g., the trend of keywords [34]) ggplot2 was used [35]. One of the most relevant data analysis tools for bibliometric data is the construction of networks, as described in the first section of this chapter. We used the software Gephi [36] for plotting the networks, with the Fruchterman Reingold layout algorithm [37]. To map communities of nodes that are strongly connected (e.g., commonly collaborating authors or co-occurring keywords), we used Louvain Modularity [38]. Each community is colored with a different color for easy readability. Below are the most common network construction methods.

- **Co-authorship**: Networks afford researchers a powerful summarization tool visualizing the collaboration among researchers and mapping the communities that shaped the field. To build co-authorship networks, we used a fractional counting method where co-authorship credit was functionally divided among authors [17], i.e., edge weights are inversely proportional to the number of authors of the paper. We used community detection to map the communities of co-authorship and find the authors who frequently collaborate together. A co-authorship network of the whole dataset can be found in chapter "The Hands that Made Computing Education Research: Top Authors, Networks, Collaboration and Newcomers" [39], Fig. 5, while other chapters include specific subsets, e.g., for the Finnish collaboration network (Fig. 3 in chapter "Computing Education Research in Finland") [40].
- **Country network**: There is no country field in bibliographic meta-data, and therefore, we used the authors' affiliation at the time of article publication to extract the countries. Two countries were considered connected if authors from each country co-authored the same article. Since country networks were based on co-authorship, they were constructed using fractional counting (similar to co-authors). An example of country collaboration can be found in chapter "Computing Education Research in the UK & Ireland" [41], Fig. 5.

- **Institutions network**: Similar to country and authorship networks, the institutional networks were created from the affiliations of the authors. Two institutions were considered connected if two authors from the two institutions co-authored a document. The networks were constructed with fractional counting. An example of institutional collaboration can be found in chapter "Computing Education Research in the UK & Ireland" [41], Fig. 4.
- **Co-citation networks**: The cited references of a paper underpin the theoretical, methodological and overall grounding of the article. The aggregation of all references can constitute the building blocks of a field as whole e.g., [1]. A co-citation network was constructed by considering two articles connected if they are cited by the same paper [29].
- **Keyword networks**: Keyword networks were created from the keywords of each article by considering a keyword is connected to another if they both were listed (co-occurred) in the same article. Unlike the co-authorship networks which have a credit issue that require special weighting, keyword networks were created with the full counting method. An example of a keyword network can be found in chapter "The Evolving Themes of Computing Education Research: Trends, Topic Models, and Emerging Research" [34], Fig. 2 devoted to studying the main themes in computing education research.

Quantitative metrics by themselves are not sufficient for the reader to understand the full picture of the research field under investigation—computing education research in our case. Therefore, for the interpretation and reporting of the results, we have relied on qualitative analysis by domain experts in the field. Such experts have analyzed the findings and built a narrative around them that tells the true story of the field of research.

## *4.5 Integrity Check*

Analysis is incomplete without quality assurance and verifying the integrity of the obtained results. Although it is depicted as the last step in our workflow (Fig. 4), quality assurance is an iterative process performed all along the study. It includes, inter alia, code auditing and exploratory data analysis of the obtained results, e.g., checking that author names do not present different forms, checking for duplicate articles, or checking for conference names in different forms. Integrity check was performed throughout the whole process several times: when inaccuracies were fixed, the process was repeated until all inaccuracies were resolved. In the same vein, all author and institutional networks in the study were audited by several authors who are in the network and they gave their feedback about the accuracy of representation which was all taken into account when building the networks.

## 5 Conclusions and Limitations

Scientometrics offers an overarching view of a research field, as well as a way to study the temporal trends of the past and the possible evolution of the future. The recent advances in scientometric methods take advantage of a vast array of statistics, visualization, machine learning, network science, and qualitative methods. As such, the transdisciplinary nature of the field allows a rigorous and nuanced evaluation and offers a wealth of potential for connecting different insights and perspectives.

In this chapter, we have outlined a step-by-step methodology for scientometric analysis. Throughout the remaining chapters of the book, the inclusion of scientometric analyses helps delineate the field of computing education research, the main players, the scientific venues, and the main themes of research, as well as the study of specific communities and subfields. Such analyses are complemented with different perspectives such as a meta-analytical view [42], case studies [43], or the study of the impact of CER research outside of academia [44], which together show the complete outlook of the field.

We have set the focus of the present chapter on the whole process that bibliometric data undergoes to be suitable for analysis, starting from a comprehensive search query and placing great emphasis on data pre-processing. Bibliometric data often contains inconsistencies in author names', keyword usage, publishing source titles, etc. Our methodology uses a combination of several tools to overcome such limitations, with the aim of obtaining the most reliable dataset possible. However, our methodology is not without limitations. First of all, although Scopus is the scientific database that has the best combination of coverage and metadata quality, it does not include all the existing papers relevant to our field of study. Although combining data from several databases is a possibility, it would only create more inconsistencies, such as the difference in citation count, or the impossibility of univocally identifying the authors by their unique database identifier. Another limitation is that the search query used might not capture all the relevant articles to computing education research. Computing education research is divided into multiple disciplines such as software engineering, cybersecurity, etc. [45]. Authors may use more granular keywords instead of the overarching term "computing education", which may have been missed by our search. Nonetheless, we believe by combining a keyword query with a venue query we obtained a representative sample, if not comprehensive. The import of data from dedicated venues, especially conferences, was also vulnerable to common issues such as missing fields, incomplete metadata that biased the query, mistakes in publication venue names, and full missing years especially in the earlier times. Therefore, the import of dedicated venue data, especially conference proceedings, may have been affected by inconsistent and incomplete metadata, and missing records. This given, scientometric analysis does not require a comprehensive dataset, but a representative sample of data.

Our methodology is not limited to computing education research, but rather transferrable to other fields of research. We hope the insights that our scientometric analysis results inspire researchers in other fields to conduct similar analyses, in a way that allows comparing scientific production and trends across fields.

## References


1. Valtonen T, López-Pernas S, Saqr M, et al (2022) The nature and building blocks of educational technology research. Comput Human Behav 128:107123
2. Priem J, Hemminger BH (2010) Scientometrics 2.0: New metrics of scholarly impact on the social Web. FMRI Tech Rep. https://doi.org/10.5210/fm.v15i7.2874
3. Thijs B (2019) Science Mapping and the Identification of Topics: Theoretical and Methodological Considerations. In: Glänzel W, Moed HF, Schmoch U, Thelwall M (eds) Springer Handbook of Science and Technology Indicators. Springer International Publishing, Cham, pp 213–233
4. van Raan A (2019) Measuring Science: Basic Principles and Application of Advanced Bibliometrics. In: Glänzel W, Moed HF, Schmoch U, Thelwall M (eds) Springer Handbook of Science and Technology Indicators. Springer International Publishing, Cham, pp 237–280
5. Mingers J, Leydesdorff L (2015) A review of theory and practice in scientometrics. Eur J Oper Res 246:1–19
6. Ivancheva L (2008) Scientometrics today: A methodological overview. Collnet j scientometr inf manag 2:47–56
7. Alan P (1969) Statistical Bibliography or Bibliometrics. Journal of Documentation 25:348–349
8. Broadus RN (1987) Toward a definition of "bibliometrics". Scientometrics 12:373–379
9. Hood WW, Wilson CS (2001). The literature of bibliometrics, scientometrics, and informetrics. Scientometrics 52:291–314
10. Abramo G, D'Angelo CA (2011) Evaluating research: from informed peer review to bibliometrics. Scientometrics 87:499–514
11. Diekmann A, Näf M, Schubiger M (2012) The impact of (Thyssen)-awarded articles in the scientific community. Kolner Z Soz Sozpsychol 64:563–581
12. Aksnes DW, Taxt RE (2004) Peer reviews and bibliometric indicators: a comparative study at a Norwegian university. Res Eval 13:33–41
13. Garfield E (1955) Citation Indexes for Science: A New Dimension in Documentation Through Association of Ideas. American Association for the Advancement of Science
14. Hirsch JE (2005) An index to quantify an individual's scientific research output. Proceedings of the National Academy of Sciences of the United States of America, 102(46):16569–16572. https://doi.org/10.1073/pnas.0507655102
15. van Eck NJ, Waltman L (2014) Visualizing Bibliometric Networks. In: Ding Y, Rousseau R, Wolfram D (eds) Measuring Scholarly Impact: Methods and Practice. Springer International Publishing, Cham, pp 285–320
16. Saqr M, Poquet O, Lopez-Pernas S (2022) Networks in education: A travelogue through five decades. IEEE Access 10:32361–32380
17. Perianes-Rodriguez A, Waltman L, van Eck NJ (2016) Constructing bibliometric networks: A comparison between full and fractional counting. J Informetr 10:1178–1195
18. Apiola M, Tedre M, López-Pernas S, et al (2021) A Scientometric Journey Through the FIE Bookshelf: 1982-2020. In: Proceedings of the 2021 IEEE Frontiers in Education Conference (FIE). pp 1–9
19. Waltman L (2016) A review of the literature on citation impact indicators. J Informetr 10:365–391
20. Ismail II, Saqr M (2022) A Quantitative Synthesis of Eight Decades of Global Multiple Sclerosis Research Using Bibliometrics. Front Neurol 13:845539
21. Tyni J, Tarkiainen A, López-Pernas S, et al (2022) Games and Rewards: A Scientometric Study of Rewards in Educational and Serious Games. IEEE Access 10:31578–31585
22. Hicks D, Wouters P, Waltman L, et al (2015) Bibliometrics: The Leiden Manifesto for research metrics. Nature 520:429–431

23. Apiola M, Saqr M, López-Pernas S, Tedre M (2022) Computing Education Research Compiled: Keyword Trends, Building Blocks, Creators, and Dissemination. IEEE Access 10:27041–27068
24. Apiola M, López-Pernas S, Saqr M, et al (2022) From a National Meeting to an International Conference: A Scientometric Case Study of a Finnish Computing Education Conference. IEEE Access 10:66576–66588
25. Rethlefsen ML, Kirtley S, Waffenschmidt S, et al (2021) PRISMA-S: an extension to the PRISMA statement for reporting literature searches in systematic reviews. J Med Libr Assoc 109:174–200
26. Norris M, Oppenheim C (2007) Comparing alternatives to the Web of Science for coverage of the social sciences' literature. J Informetr 1:161–169
27. Singh VK, Singh P, Karmakar M, et al (2021) The journal coverage of Web of Science, Scopus and Dimensions: A comparative analysis. Scientometrics 126:5113–5142
28. Baas J, Schotten M, Plume A, et al (2020) Scopus as a curated, high-quality bibliometric data source for academic research in quantitative science studies. Quantitative Science Studies 1:377–386
29. Aria M, Cuccurullo C (2017) bibliometrix: An R-tool for comprehensive science mapping analysis. J Informetr 11:959–975
30. OpenRefine. https://openrefine.org/. Accessed 2 May 2022
31. Clustering methods in-depth. https://docs.openrefine.org/next/technical-reference/clustering-in-depth. Accessed 2 May 2022
32. Patil I (2021) Visualizations with statistical details: The 'ggstatsplot' approach. Journal of Open Source Software 6(61):3167. https://doi.org/10.21105/joss.03167
33. Apiola M, López-Pernas S, Saqr M (2023) The Venues that Shaped Computing Education Research: The Gatekeepers Under the Lens. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer
34. Apiola M, López-Pernas S, Saqr M (2023) The Evolving Themes of Computing Education Research: Trends, Topic Models, and Emerging Research. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer
35. Wickham H, Wickham MH (2007) The ggplot package. URL: https://cran r-project org/web/packages/ggplot2/index html
36. Bastian M, Heymann S, Jacomy M (2009) Gephi: an open source software for exploring and manipulating networks. In: Proceedings of the third international AAAI conference on weblogs and social media. AAAI Press, Menlo Park, CA, pp 361–362
37. Fruchterman TMJ, Reingold EM (1991) Graph drawing by force-directed placement. Softw Pract Exp 21:1129–1164
38. De Meo P, Ferrara E, Fiumara G, Provetti A (2011) Generalized Louvain method for community detection in large networks. In: 2011 11th International Conference on Intelligent Systems Design and Applications. IEEE
39. Apiola M, López-Pernas S, Saqr M (2023) The Hands that Made Computing Education Research: Top Authors, Networks, Collaboration and Newcomers. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer
40. Malmi L, Hellas A, Ihantola P, et al (2023) Computing Education Research in Finland. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer
41. Becker B, Bradley S, Sentance S, et al (2023) CER in the UK and Ireland. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer
42. Malmi L, Simon, Sheard J, et al (2023) The Evolution of CER: A Meta-Analytic Perspective. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer
43. Daniels M, Berglund A, Pears A (2023) A Case Study: The Uppsala Computing Education Research Group (UpCERG). In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer

44. Saqr M, López-Pernas S, Apiola M (2023) The Impact and Chatter About Computing Education Research across the Social Media, News, Patents and Blogs: The Case of Altmetrics. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer
45. Association for Computing Machinery ACM Curricula recommendations. https://www.acm.org/education/curricula-recommendations. Accessed 14 Aug 2022

# The Hands that Made Computing Education Research: Top Authors, Networks, Collaboration and Newcomers


**Mikko Apiola, Mohammed Saqr, and Sonsoles López-Pernas**

## 1 Introduction

The history and evolution of computing education research (CER) involves a number of milestones, activities, efforts, and curiosities [4]. Especially in the past 20–25 years, CER has matured from being mainly an interest of computing educators for sharing experience reports and course descriptions, into a research specialisation with established venues of dissemination, methodological rigor, theoretical viewpoints and seminal publications [12, 29]. Trends, expectations and concerns change. Technologies rise and fall. But the fundamental forces that drive the evolution of a scientific discipline originate in people, and their networks of collaboration. The evolution of CER is determined by people and their intuitions, ideas, and actions. Then, it becomes relevant to ask, who are the influential people of CER and how do their collaboration shape the discipline of CER.

In academia, researchers are associated with specific institutions, they belong to research groups, such as the UpCERG-group [9], and in addition they have their own evolving networks of collaborators within and outside of their immediate environments. As authors proceed in their careers, they often change institutions and research groups, become parts of collectives of authors, and their academic trajectories involve varying kinds of collaborations with new and old authors, within their personally evolving networks of collaborators. Sense of belonging to a community is often defined, e.g., through attributes, such as a shared sense of personal relatedness, a sense of mattering through a feeling of having an influence

M. Apiola (✉) · M. Saqr · S. López-Pernas
University of Eastern Finland, Joensuu, Finland
e-mail: mikko.apiola@uef.fi; mohammed.saqr@uef.fi; sonsoles.lopez@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_6

in the community and the community having an influence to the member, sense of fulfilment of needs related to what the group does, and an emotional connection [20]. Belonging to communities is linked to one's identity, and members of a community share an identity-forming narrative.

One meaningful form of collaboration between two researchers is that of co-authoring a publication. Publications are often co-authored with colleagues of varying kinds, who maybe strongly or loosely connected with each other. They may or may not belong to the same groups and communities. While a co-authorship may or may not indicate belonging to a community, a co-authorship is a meaningful indicator of a shared agreement to conduct a research: a shared responsibility on selecting the methods, collecting data, analysing the data, and writing a research report. Authors also share the impact to their reputation that a publication may cause. Therefore, authorships and co-authorships are an important form of collaboration. In this chapter, we investigate the authorship patterns in CER from the perspectives of author productivity, clusters of co-authorships and international collaboration between authors. By doing this, we contribute one interesting viewpoint into authors and their collaboration. This view may be used as the basis for more in-depth analyses of author communities of CER.

## 2 Related Research

A number of previous studies have used CER publication data to investigate the authorship patterns and collaborations of CER. Previous research have investigated publications from the major well-known publication outlets, such as SIGCSE, ITiCSE, Koli Calling, ACE, FIE, and ICER [4]. Many of the analyses provide a narrative of each venue, positioning them geographically, with regard to their special focus areas, and by outlining main historical events, such as changes in conference chairs. In addition, previous research has used publication data to identify key characteristics of authorships, including patterns of collaboration, geographical concentration and diversity, internalisation, maturity of the field, patterns of newcomers versus old-timers, and networks of core authors.

A much used metric to investigate author productivity in CER publications is Lotka's Law (developed by Alfred James Lotka, 1880–1949), according to which, in a sufficiently large set of publications within a scientific discipline, the numbers of author contributions will follow a pattern, where e.g. some 60% of authors will contribute to one paper, 15% to two, 7% to three, and more generally, the number of contributions to a given number of papers bears an inverse square relationship to that number of papers [22]. Lotka's Law has become one of the basic laws of informetrics, and has been widely applied since its anniversary in 1926, especially after 1960s, when it became a hotspot along with development of informatics theory [23]. In the context of CER, research has revealed a comfortable fit between the observed and expected numbers of author contributions with regard to Lotka's Law

in a set of articles published in the Olympiads in Informatics journal [29, pp. 54–55], as well as in sets of publications from Koli Calling [31] and ICER [30]. It is argued that both Koli Calling and ICER display productivity characteristics of a research discipline, while ITiCSE less so as it does not appear to satisfy Lotka's Law [29, p. 54], [32]. Recent analyses of authorship patterns in publications in ACE [33] seem to conform with patterns observed in Koli Calling and ICER. In all, these analyses have been used to support the claim that authorship patterns in many, but not all, venues of CER, resemble patterns of authorship found in other disciplines of science [29–33].

Another metric about CER communities is that of the size of a *Giant Component*, which refers to the largest subgraph within a network of authors, where nodes represent individual authors and edges represent co-authorships between authors [18]. The giant component has been argued to represent the "essence" of the community [21]. In an analysis of papers from six CER conferences, the results show size of giant component ranging between 14% and 46% in CER outlets [21]. With regard to ICER, a significantly higher giant component size of 49% was observed [21]. The giant component size for ITiCSE working groups (1996–2016) was found to be 97% [18]. The sizes of giant components in CER communities were found to be significantly smaller when compared with giant components found in CS communities [21]. This has been partly explained by the patterns of author entrance.

The MEIN classification [21] was designed to classify author entrance in CER. In MEIN, each article is classified to a category of M (merge), if the paper includes authors who were previously not part of a same cluster; E (extend), if a paper includes both old-timers and newcomers, and all old-timers were part of a same cluster; I (internal), if all authors are old-timers who are also part of a same cluster; N (newcomers) if all of the authors have zero previous publications in the data [21]. Researchers categorised publications from six CER venues in 2012, and compared the results with general CS conferences, revealing significantly higher proportions of I and N types of papers as compared to computer science (CS) conferences [21]. More recent analyses of ITiCSE (1996–2019) show a portion of internal papers of 18% and new papers 39% [32], and low patterns for ITiCSE Working Groups (1.0% for internal papers and 5.1% for newcomers in 1996–2016 data)[18]. In all, the MEIN analyses suggest that CER communities are "introverted"; researchers communicate less than computer science researchers, and newcomers do not integrate that well. With the exception of the extroverted profile of the ITiCSE Working Group [18], CER papers seem to be often written by either a group of newcomers or a group of old-timers, who are already connected with each other [18, 21, 32].

Other analyses of collaboration in CER includes the analysis of collaboration patterns among institutions that produce CER [4]. A *rich club* analysis refers to detection of subsets of connected nodes that interact primarily among themselves, indicating dominance in collaboration [8, 16, 38]. A rich club analysis revealed a subset of 52 institutions, which were involved in some 9% of all articles in a dataset of CER publications from major publication outlets that publish CER

[4]. In other research, authors identified author collaboration clusters in Frontiers in Education (FIE) publications [5]. Other metrics and indicators of CER authors and collaboration include identification of top contributors and most cited authors, average papers per author, average collaborators per author, geographical origins of authors, international versus national co-authorship patterns, and others. In an analysis of the collaboration networks of ITiCSE, ICER and SIGCSE, it was found that while collaboration between authors from different institutions is increasing, collaboration mostly happens within countries rather than between countries [37].

An analysis from 2016 [31] investigated the authors and authorships of Koli Calling, by zooming into most contributing authors, returning authors, authors with most co-authors, top contributing countries and numbers of papers with a multinational author team [31]. Results show a solid core of returning authors, and a strong dominance of authors based in Finnish institutions, followed by authors from Germany, Sweden, UK, and Australia [31]. Similar findings were found in a recent scientometric analysis of Koli Calling publications [3]. Analysis of author affiliations show that a quarter of papers entailed collaboration between institutions, and a half of that quarter included authors from multiple countries [31]. An earlier analysis from 2009 [28] compared publications of Koli Calling and Informatics in Education (IiE), and the results show that Koli Calling had more repeat authors than IiE in 2009 [28] and that in both Koli Calling and IiE, which is a journal based in Lithuania, Finland dominated both with regard to authorships and articles [28]. In ACE, most papers have originated from Australia or New Zealand [33]. In other venues, dominance of USA and other high-income countries is seen [5, 7, 26, 37].

To summarise, previous analyses have revealed diverse collaboration patterns between authors, profiles of author productivity, and increasing international collaboration. On the other hand, analyses have also shown an introverted nature of collaboration, where newcomers might not always integrate that well. With regard to research of authors and their collaboration, a number of avenues for future research exist. Most previous analyses have concentrated on one or a few venues of dissemination at a time, lacking a more holistic analysis of authorship patterns and collaboration. In addition, there seems to be much room to extend the current metrics with modern scientometric methods. Future analyses could also be better connected to theoretical understanding about collaboration and communities.

## 3 Methods and Data

The metadata of publications in all central venues of CER, as well as metadata of CER publications published elsewhere, identified by a keyword search, were retrieved from Scopus, which has the most comprehensive metadata of CER papers [14]. The full process of obtaining the data, including procedures of selecting keywords and venues, database search, data screening and data cleaning are explained in Chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" of this book [14].

**Table 1** Descriptive statistics

| Data set statistics | |
|---|---|
| *Authorship data* | |
| Authors | 20,391 |
| Authors (after additional filtering) | 20,144 |
| Author appearances | 45,698 |
| Authors of single-authored documents | 3125 |
| Authors of multi-authored documents | 17,019 |

The dataset contains a total of 16,863 articles. The cleaned data included 20,391 authors. Additional filtering reduced the total number of authors to 20,144. The descriptives of the dataset used for authorship analysis are presented in Table 1.

## 4 Productive Authors

Out of all the 20,144 different people who have authored or co-authored articles in CER within our dataset, some 13,330 (66.2%) have authored or co-authored only one paper, while some 3089 (15.3%) have authored or co-authored two papers. Some 1251 (6.2%) have authored or co-authored three papers. The observed values for number of authorships in our data fit well with the expected values suggested by Lotka's Law of author productivity [22] (see Fig. 1) with constant parameters ($p = 2.258678$, $C = 0.6998023$). $C$ and $p$ are constants that vary according to the discipline but are generally expected to be close to $C = 0.60$ and $p = 2$ [30]. The goodness of fit was confirmed by a Kolmogorov-Smirnov (K-S) test, showing that the data fits Lotka's Law at the 0.01 level of significance.

Within our dataset, a total of 20,144 people have authored or co-authored publications in CER. Figure 2 shows the top 20 authors' production over time pre 2000, while Fig. 3 shows the 20 authors with the highest numbers of paper authorships after 2000. Before 2000, the most contributing 20 authors wrote 15 or more publications, and the highest number of contributions was 28. After 2000, all authors in the top 20 list contributed some 49 or more articles. The largest number of contributions post 2000 was 82. The largest number of contributions over all of the years in the dataset was 87 by Mark Guzdial. It is considered that 2000 marks the time after which CER became more established as a research discipline with new venues of dissemination, increase in scientific rigor, publication of influential books, and establishment of professional positions [4], which is why 2000 was chosen as the cutpoint.

The list of most productive authors features a number of prominent CER researchers. The largest number of authorships in the data before 2000 was 28 by Robert M. Aiken, an ACM Fellow with various received awards including ACM Outstanding Contribution Award (1996), IEEE Computer Society Golden Core Award (1996), IFIP's Silver Core (1992), ACM SIGCSE's Outstanding Contributions to Computer Science Education (1995) and Lifetime Service Awards (1999).

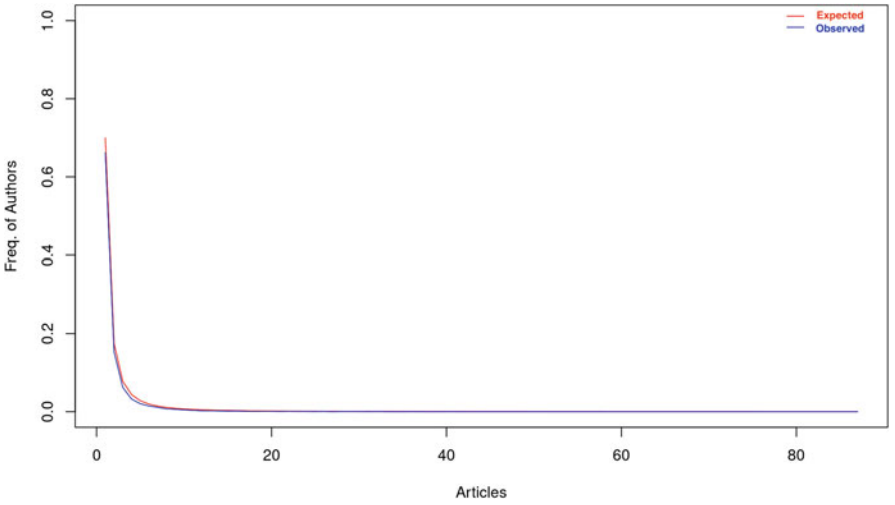


**Fig. 1** Author productivity and Lotka's Law. The y-axis represents the frequency of authors. The x-axis represents the expected number of contributions according to Lotka's Law. The red line represents the expected pattern of authorship according to Lotka's Law, while the blue line represents the observed pattern of authorship in CER

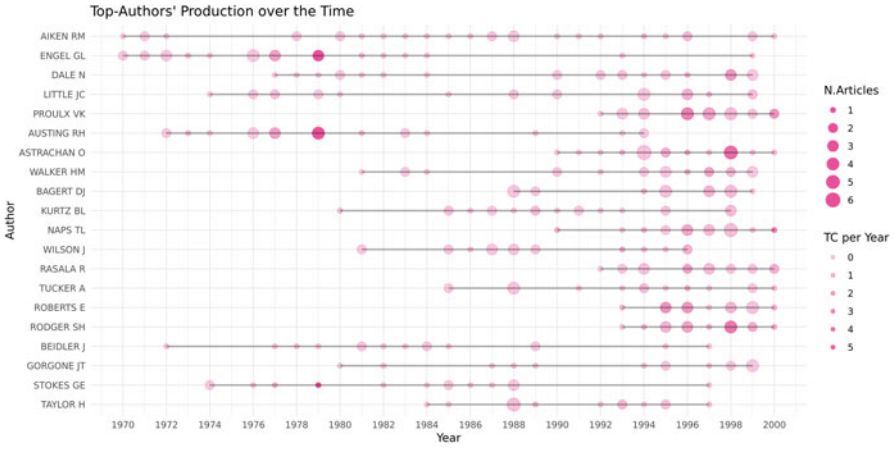


**Fig. 2** Top authors' production (until 2000). The size of the circle indicates number of articles. TC (total citations) is indicated by the color, with stronger red resembling more citations while lighter red resembles less citations. The range is 0–5 citations, which is affected by the fact that citation data was recorded less systematically in the times before 2000s [14]

His most cited work in the data is a review of using animations in understanding algorithms, co-authored with Judith Wilson [35]. Top contributors also include Gerald L. Engel with 25 contributions and his most cited work being the CC'78 curriculum recommendations [6]; Nell Dale with 24 authorships (most cited work on teaching recursion [36]), Joyce Currie Little (23 authorships, most cited work on

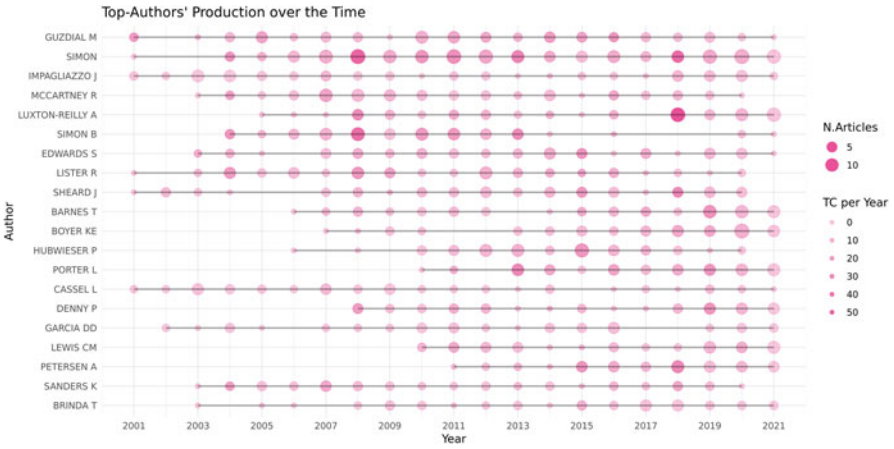


**Fig. 3** Top authors' production (post 2000). The size of the circle indicates number of articles. TC (total citations) is indicated by the color, with stronger red resembling more citations while lighter red resembles less citations, with a range from 0 to 50 citations

evaluation [1]), and Richard H. Austing (22 authorships, most cited work on CC'78 recommendations [6]).

After 2000, the largest number of paper authorships is 82 by Mark Guzdial, a receiver of multiple awards and a key figure in the CER field. His most cited work in the dataset is a multi-institutional assessment of programming skills [19]. Top contributors after 2000 also include Simon (81 contributions, most cited work is a review of introductory programming [15]), John Impagliazzo (75, most cited work is an overview report on computing curricula 2005 [27]), Robert McCartney (73, most cited work: [13]), and Andrew Luxton-Reilly with 70 contributions, and his most cited work being a review on introductory programming [15].

## *4.1 Newcomers and Old-Timers*

Figure 4 shows all articles in the dataset divided to categories based on whether the authors of the paper are all appearing for the first time in CER (newcomers), all authors have published before (recurring), or if the paper includes both newcomers and oldtimers (mixed). The total number of articles written by an all-newcomer author team was 5446 (32.3%), an old-timer author team 5921 (35.1%), and a team mixed with old-timers and newcomers 5496 (32.6%). Before 2000, the largest category of papers was the group of papers written by all-newcomer author teams (47.7%). After 2000, the situation started to change, and by 2020s, papers authored by a mixed author team became the largest group of papers. Figure 4 shows the decline of the newcomer-category, with the category becoming the smallest after 2010, and the mixed-category becoming the largest. The numbers of citations are represented in the figure by the colored balls. We also investigated

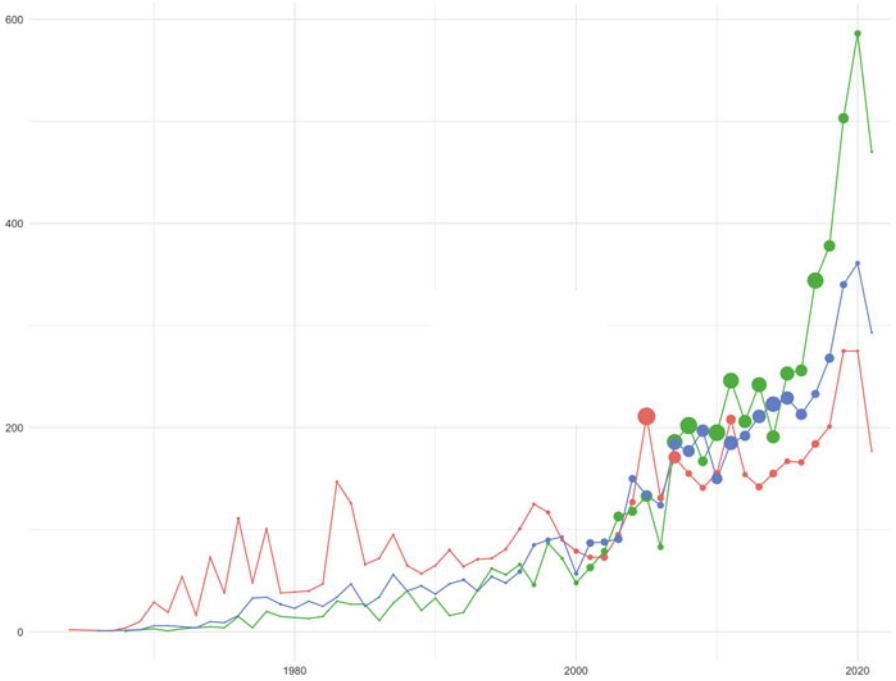


**Fig. 4** All papers divided to (**a**) all authors are newcomers (red) (**b**) authors include a mixture of newcomers and old-timers (green) (**c**) all authors are old-timers (blue). The y-axis represents number of articles. The x-axis represents the year of publication. The decline after 2020 is caused by not all metadata being recorded at the time of data retrieval (see [14])

the amount of citations to papers in the three different categories. The mean numbers of citations to papers written by all-newcomer author teams was (5.93), for papers written by oldtimers (8.55), and for papers written by a mixed team (8.98). The analysis shows that papers written by all-newcomer author teams received significantly less citations as compared to papers written by oldtimers or papers written in a mixed author team. A Welch's Analysis of Variance (ANOVA) confirms that at least one of the means in this analysis differs significantly from the others [$F_{Welch}(2, 10{,}855.97) = 41.4$, $p < 0.001$]. Post-hoc tests confirmed significant differences between all-newcomers and oldtimers, as well as between all-newcomers and mixed-teams ($p_{Holm\text{-}corrected} < 0.0001$ for both).

## 5 Clusters of authorship

Figure 5 presents co-authorships in the papers published in CER venues. The nodes represent authors with the most co-authors, and the edges represent co-authorships between the authors. Unconnected nodes are active collaborators, but whose co-

authors do not belong to the group of most active collaborators in Fig. 5. Some clearly identifiable clusters have formed around the authors in CER. For each cluster of authors, we also inspected the top five keywords from a subset of all articles written by members of the cluster in our data (after removing keywords: computing education, computer science education, education). The top five keywords of each cluster are presented in parenthesis when introducing the clusters.

**The orange cluster** *(CS1/CS2; novice programmers; programming education; threshold concepts; debugging)* down from the center is formed around well-known authors and influential contributors, including Beth Simon (University of California, San Diego), a teaching professor, a top contributor in CER, working, among other things, in technology-enhanced learning and K-12 computing education; Robert McCartney (University of Connecticut), awarded CER researcher and the former

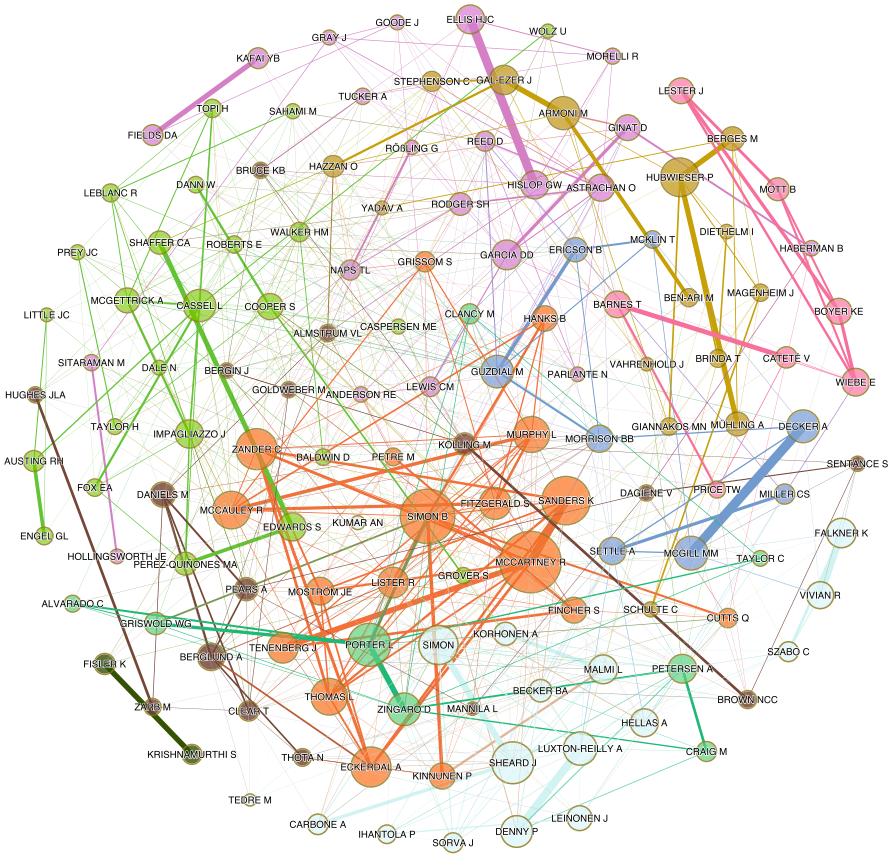


**Fig. 5** Co-author network of CER authors with most collaborators. Node size indicates the number of unique co-authors, edge thickness indicates number of co-authorships, and colors indicate communities of researchers who frequently collaborate together (as determined by Louvain modularity)

editor of ACM TOCE; Kate Sanders (Rhode Island College), focusing on empirical CER, and a participant in the bootstrapping CER workshop in 2002–2003, and a top author in many venues of dissemination on CER; Raymond Lister (University of Technology, Sydney), a top contributor in CER, specialising in teaching of computer programming; Josh Tenenberg (University of Washington), and others.

**The light blue cluster** *(CS1/CS2; programming education; novice programmers; assessment; LA-EDM)* at six o'clock is built around top contributing CER researchers from Australasia and Finland, including Simon (University of Newcastle, Australia), a pioneer in meta-research in CER and inventor of Simon's system of classifying publications; Judy Sheard (Monash University, Australia), a top contributor in CER focusing in educational technology and assessment; Andrew Luxton-Reilly (University of Auckland); and Finnish CER researchers Lauri Malmi, Ari Korhonen, Juha Sorva, Arto Hellas (Aalto University, Finland); Petri Ihantola (University of Helsinki, Finland); Paul Denny (University of Auckland). **The light green cluster** *(curriculum; CS1/CS2; object-oriented programming; K-12; visualization)* is centered around Lillian Cassel (Villanova University); Clifford A. Shaffer (Virginia Tech); Eric Roberts (Stanford University), a many-times awarded contributor in CER; John Impaggliazzo (Hofstra University) who is an author of many books, articles, contributor to CER in diverse topics, including history of computing, and receiver of many IEEE and ACM/SIGCSE awards; Stephen H. Edwards (Virginia Tech), with focus on many aspects in CER including automatic assessment, metrics, and gamification; Richard H. Austing (University of Maryland); and many others.

**The pink cluster** *(K-12; gender and diversity; pedagogy; computational thinking; e-textiles)* at the upper side of Fig. 5 is centered around prominent CER authors Heidi Ellis (Western New England University), who focuses e.g. in software engineering education, open-source software, and tools for biological data analysis; Owen Astrachan (Duke University); Dan Garcia (UC Berkeley), founder of the "CSforALL" movement, and winner of numerous awards and frequent SIGCSE contributor; Yasmin Kafai (University of Pennsylvania); Gregory Hislop (Drexel University), a contributor to numerous initiatives and efforts in computing and CER; and others. **The mint green cluster** *(CS1/CS2; peer instruction; assessment; novice programmers; data structures)* is formed around the influential CER authors Leo Porter (University of California); Daniel Zingaro (University of Toronto); Andrew Petersen (University of Toronto); and Christine Alvarado (University of California). **The brown cluster** *(K-12; CS1/CS2; programming education; pedagogy; computational thinking)* includes CER authors, many from Sweden or the Baltic Countries, including Anders Berglund (Uppsala University, Sweden); Arnold Pears (KTH Royal Institute of Technology, Sweden); Mats Daniels (Uppsala University, Sweden); Valentina Dagiene (Vilnius University, Lithuania); Neena Thota (University of Massachusetts); Sue Sentance (Raspberry Pi Foundation).

**The blue cluster** *(K-12; CS1/CS2; gender and diversity; assessment; programming education)* carries top CER authors such as Mark Guzdial (University of Michigan); Adrienne Decker (University of Buffalo); and Briana B. Morrison (University of Nebraska). **The gold cluster** *(K-12; gender and diversity; program-*

*ming education; CS1/CS2; object-oriented programming)* includes CER researchers from Israel and Germany: Peter Hubwieser (TU Munich), with focus on empirical investigation of learning processes in computer science; Mordechai Ben-Ari (Weizmann Institute of Science); Judith Gal-Ezer (The Open University of Israel); Marc Berges (TU Munich). **The light pink cluster** *(K-12; computational thinking; block-based programming; game-based learning; gender and diversity)* includes, among others, CER researchers James C. Lester (North Carolina State University); Bradford W. Mott (North Carolina State University); Tiffany Barnes (University of North Carolina); Veronica Cateté (North Carolina State University). The smaller **dark green cluster** *(plan composition)* is formed around the CER authors Shriram Krishnamurthi (Brown University); Kathi Fisler (Brown University). In all, the co-author network of authors with the most collaborators reveals an impressive set of clusters with prominent authors from different parts of the globe, pushing the boundaries of CER and working on a diverse set of topics. Similarities and differences are found in the top five keywords of each cluster.

## 6 Authors Who Build Bridges Between Communities

The importance of integrating disciplines and research cultures is acknowledged well, e.g. the need to integrate liberal arts and social sciences with engineering is advocated by many [25]. One known issue is that closely connected disciplines or research groups within a discipline may place stronger emphasis on some preferred research goals, paradigms, methods, and approaches over others. Therefore, bridging communities may be valuable to bring new ideas and shake up outdated ways of thinking. In efforts to build bridges between the communities of SIGHCI (Special Interest Group in Human-Computer Interaction) and SIGCHI (Special Interest Group in Computer-Human Interaction), emphasis was put on bridging communities with their unique orientations towards "making real-world contributions", or on emphasis on high standards in scientific rigor [11], a common distinction also in CER, with its varying preferences for tools-research or empirical research [10]. Figure 6 presents a network of authors based on co-authorships in papers. The figure highlights authors who bridge separate or isolated communities. A group of authors who are highlighted as authors who bridge communities, in no specific order, include: Simon, Arnold Pears, Judithe Sheard, Owen Astrachan, Beth Simon, Lillian Cassell, John Impaggliazzo, Dan Garcia, Mark Guzdial, Colleen M. Lewis, Susan H. Rodger, Amruth Kumar, Stephen H. Edwards, Tiffany Barnes, Joyce Currie Little, Steve Cooper, Henry M. Walker, Tony Clear, Briana B. Morrison, and Joel C. Adams.

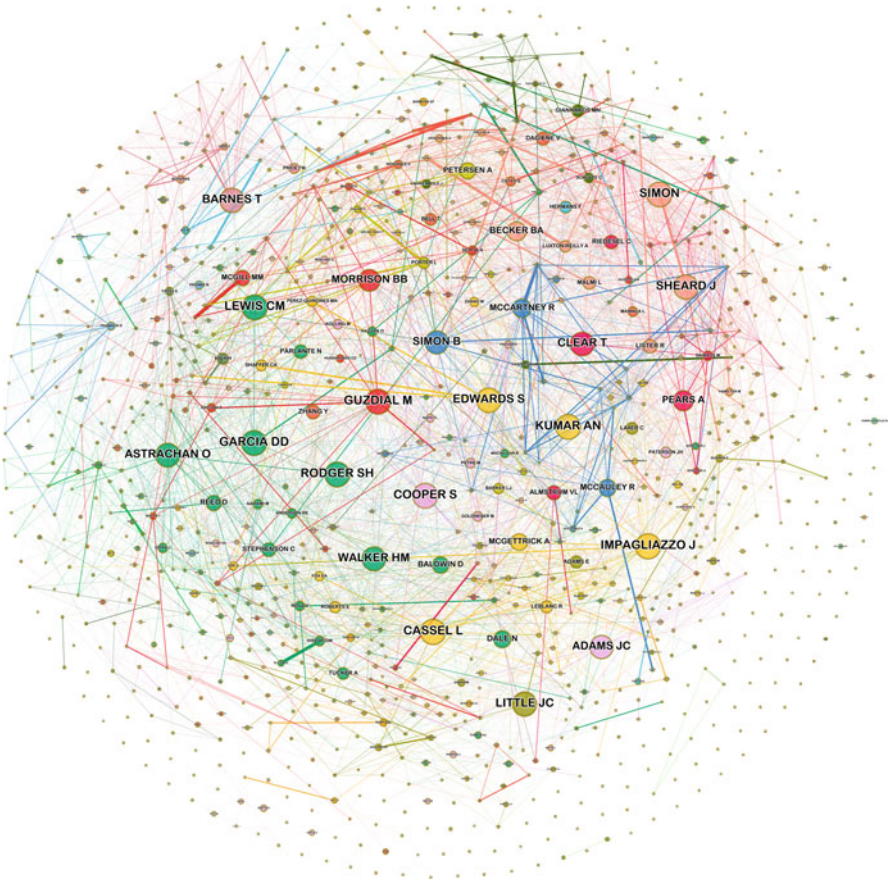


**Fig. 6** Authors who build bridges. The figure highlights authors, based on co-authorship patterns in data, who have acted as bridges between separate or isolated communities of co-authorship

## 7 International Collaboration and Venues

With regard to international collaboration, Fig. 7 shows the trend of articles divided to ones written by author teams from a single country versus those written by author teams from multiple countries. The analysis shows that while a good number of articles are authored in international teams, by far the largest share of papers are still written by authors from a same country. One characteristic of CER as a research discipline is the tendency to publish a lot in conferences, a tendency inherited from the CS discipline (see Chap. 7 [2]). Authoring and co-authoring articles deepens collaboration among authors. Attending conferences fundamentally shapes author networks as people meet and network with each other. Analyses of authorship patterns with regard to the most popular publication venues of CER reveals co-attendance patterns in conferences and other publication venues (see

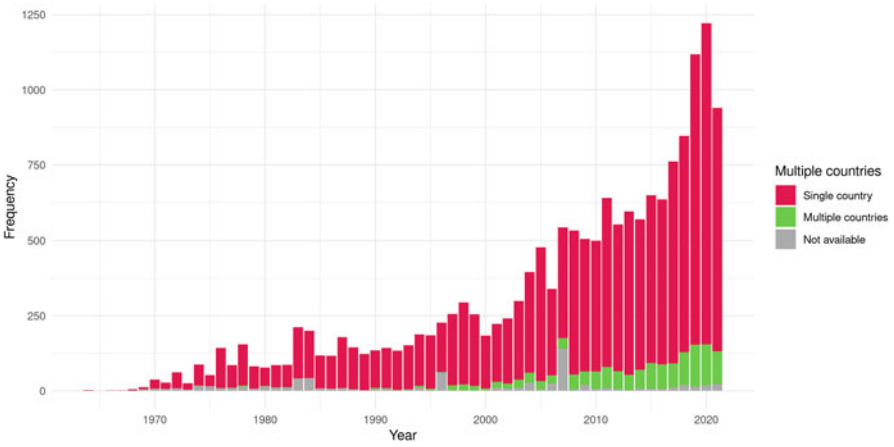


**Fig. 7** International collaboration. The x-axis shows the year of publication, while the y-axis represents the frequency of publications. The red category includes papers with authors teams from a single country, while the green category includes articles with authors from multiple countries. The gray category represents articles with no country metadata available

Fig. 8). There are clear patterns of attending multiple publication venues within CER. For example, many authors have published both in SIGCSE as well as its European sibling ITiCSE, as revealed by the strong connection among them in Fig. 8. While a number of previous research have analysed publication and authorship patterns in specific venues of CER, future research could dig deeper into building a combined understanding about the different publication venues and their networks of authors and communities, and what impact this has on co-authoring patterns.

## 8 Discussion

### *8.1 Author Productivity and Productive Authors*

Analysis of author productivity shows a good fit between author productivity data in the CER discipline with regard to Lotka's Law [22]. Previous research have investigated author productivity patterns within publications in one or a few CER venues at a time, showing that authorship patterns in many, but not all, venues, conform to Lotka's Law [29–33]. What does this mean? It has been argued that development of a scientific discipline progresses through the stages of incubation, growth, and maturity, and the activities of authors within these stages vary [23]. Lotka's Law may not be able to fully explain the distribution of authors in earlier stages than maturity [23]. The fact that some venues of CER satisfy Lotka's Law while others do not, could, perhaps, be one indication that some venues of CER are

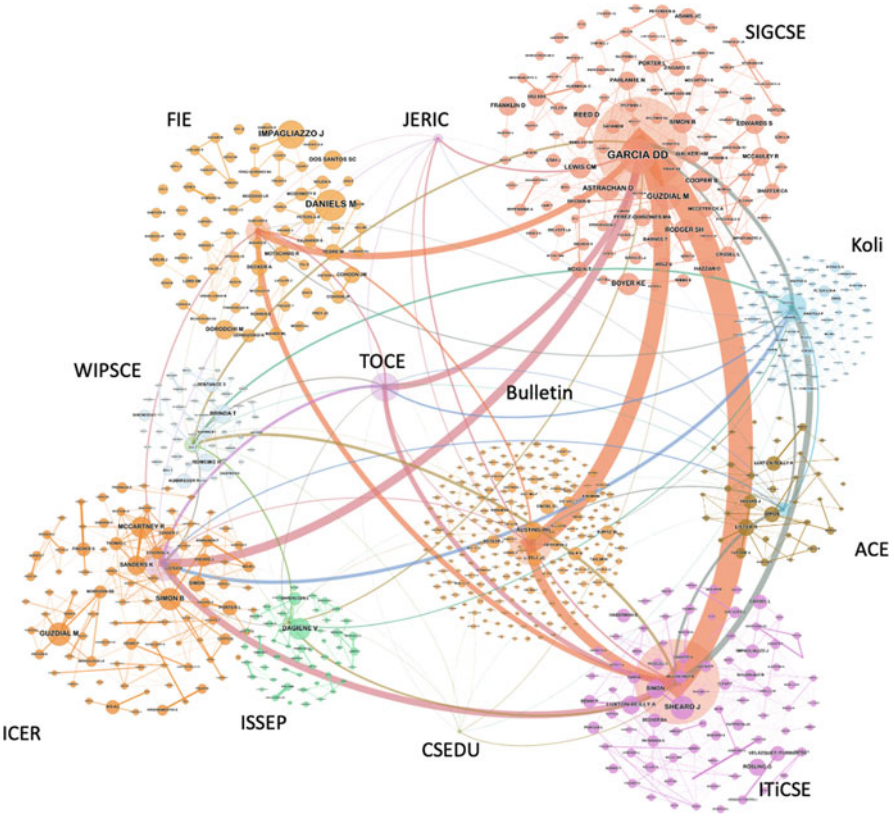


**Fig. 8** Network of co-authorship patterns between top publication venues of CER. If an author has authored papers in two of the venues, then they are connected. TOCE refers to ACM Transactions on Computing Education, while CSEDU refers to Computer Science Education. The other acronyms are explained in Chap. 7 [2]

still in the earlier stages of incubation and growth, while others are more mature with regard to author productivity. In all, when evaluating the development of CER as a research discipline, Lotka's Law has, and can be, used as one indication of progress, together with e.g. Fensham's two sets of criteria for evaluating a scientific field (see Chap. 4 [17]), and the framework of scholarship by Glassick (see Chap. 2 [10]). Our analysis confirmed and extended the previous analyses by including a broader range of venues of CER, and the results indicate that CER publications do follow authorship patterns that resemble those found in other disciplines of science, too.

Analysis of publication data reveals a cohort of highly productive authors. Since the CER field was significantly different prior and after 2000, and to shed light to the previous times, we analysed the most contributing authors prior and post to 2000. The identified highly productive authors and their contributions shed light on

works with high impact from recent and previous times, and may act as inspiration for researchers looking for innovative ideas, topics to work on, or partners for collaboration.

## 8.2 Newcomers and Recurring Authors

With regard to author entrance patterns, previous research has shown that, as compared to general CS publication venues, in many, but not all [18], publication venues of CER, there is a significantly high proportion of papers written by all-newcomer author teams or oldtimers [21, 32]. Thus, previous analysis suggests that CER researchers have a tendency to stick more with nearby colleagues as compared to CS researchers, and also newcomers have harder time integrating as compared to CS researchers [21]. While a comparison between CER and CS was done, no thresholds were determined for introverted or extroverted profiles [21]. Previous findings suggest an introverted profile of CER, where papers are often authored either by a group of newcomers or a group of old-timers, who are already connected with each other, suggesting that newcomers might not integrate that well, at least if compared to general CS [18, 21, 32]. Our investigation sheds new light into these previous findings. With regard to times before 2000, papers written by newcomers clearly dominated the CER, with some 47.7% of papers written by newcomers. However, after 2000s, especially after around 2005, papers authored by a mixture of newcomers and old-timers became the largest category. After 2000, the shares of papers in these three categories were: all newcomers (30%), all recurring (33.7%), and mixed (36.2%). Future research could do a comparison of the authorship patterns in CER with other scientific disciplines. Our findings reveal that papers with all-newcomer author teams receive significantly less citations than papers in the other categories. When interpreting the findings it is good to note that, especially with regard to certain venues of CER, citation rates have been shown to be low [4].

## 8.3 Collaboration and Building Bridges

Analysis of the co-author network reveals clusters of authorships between many highly influential authors, many of which are based in North America, but also including clusters that are based in the Nordic Region, Europe, Australasia, Israel, and many other parts of the globe. An analysis of top five keywords visible in all the published works in our dataset by the authors in each constellation shows similarities but also differences in the top keywords. Teaching programming, introductory courses, and K-12 seem to dominate. Other top five keywords of the constellations include gender and diversity, learning analytics and data mining, visualisations, data structures, block-based programming, game-based learning and others, giving hints about the focus areas in each of the constellations. However,

such analysis is shallow without a proper inspection about the actual underlying research groups and communities and topics of research that are pursued in those. Co-authorship analyses or analyses of top keywords per author constellations do not reveal the deeper essence of the underlying author communities. Therefore, deeper analyses and research are needed to gain a more comprehensive picture about the influential research groups and underlying community structures. Many such analyses are provided in other chapters of this book.

Our analyses of bridging communities highlighted, based on co-authorship patterns in the data, a specific group of authors, who have acted as bridges between separate communities of authors. It is important to build bridges between communities, which may have their own and unique specific and preferred goals, methods, ways of working, research cultures and values. Future research needs to investigate this issue with more depth.

## *8.4 International Collaboration*

Our analysis shows that while a significant share of articles have been authored by international teams of authors, the major share of articles are still written by authors from a single country. These findings add to analyses of single publication venues of CER, such as that of ITiCSE, ICER and SIGCSE, which showed collaborations mostly happening within countries rather than between countries [37], with similar findings found with regard to Koli Calling [31], ACE [33], and other venues [5, 7, 26, 37]. The tendency of collaborating mostly with authors from one's own country raises the question about the impact of co-publishing in many venues of CER to co-authorship patterns. In a brief analysis about how the venues of dissemination are linked in co-publishing, a clear pattern emerges from authors who tend to publish their research in several of the known venues. Those venues of co-publishing may often be geographically located far away from each other, and attending requires international travels. A deeper analysis of the communities formed around these venues and the impact of conference attendance to publication habits is suggested as a future research.

## *8.5 Limitations and Future Research*

Analysing publication data can reveal many previously hidden aspects of authors and scientific communities. Analyses may bring visible systemic challenges, such as a tendency of newcomers to not integrate well in CER communities [21], patterns of collaboration, authorships, and citation habits. While there is a lot of research and concrete action around topics of diversity, inclusion, equity and broader participation in computing [24, 34], many aspects of participation are still hard to reach by analysing publication metrics. The deeper essence about the cultures of CER

communities still remains beyond scientometric analyses, including communities' social structures, norms, values, management styles, attitudes, and hidden agendas, which together contribute to whether the members of the community and the community as a whole is healthy and thriving. To what extent do communities of CER provide a culture of challenge, inspiration and freedom that pushes its members to go beyond what is already known, tackle challenges, and grow as independent and innovative researchers? Additional data collection and measures are needed for deeper analyses of CER communities.

## 9 Conclusions

The evolution of CER as a research field is fundamentally driven by people and their networks. In this research, we contribute to previous analyses of CER authors, which have often been conducted in the context of one or two venues of dissemination. In our analysis we use a larger set of data from all central publication outlets of CER, and we have added to the previous analyses of authors by providing new insights to author productivity patterns, most productive authors, clusters of co-authorships between authors, and international collaboration. Our results have revealed a healthy author productivity pattern among CER, healthy pattern of involving more newcomers as co-authors in papers, patterns of publication among top constellation of authors, and patterns of international collaboration and co-attendance to venues of publication. In all, in this research, we have offered a unique viewpoint to the authors and authorship patterns in CER, and paved the way for more in-depth analyses of authors and their communities in CER.

## References

1. Almstrum, V.L., Dale, N., Berglund, A., Granger, M., Little, J.C., Miller, D.M., Petre, M., Schragger, P., Springsteel, F.: Evaluation: Turning technology from toy to tool: Report of the working group on evaluation. SIGCSE Bull. **28**(SI), 201–217 (1996). https://doi.org/10.1145/237477.237648
2. Apiola, M., López-Pernas, S., Saqr, M.: The Venues that Shaped Computing Education Research: Dissemination Under The Lens. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
3. Apiola, M., López-Pernas, S., Saqr, M., Pears, A., Daniels, M., Malmi, L., Tedre, M.: From a National Meeting to an International Conference: A Scientometric Case Study of a Finnish Computing Education Conference. IEEE Access **10**, 66576–66588 (2022). https://doi.org/10.1109/ACCESS.2022.3184718
4. Apiola, M., Saqr, M., López-Pernas, S., Tedre, M.: Computing education research compiled: Keyword trends, building blocks, creators, and dissemination. IEEE Access **10**, 27041–27068 (2022). https://doi.org/10.1109/ACCESS.2022.3157609
5. Apiola, M., Tedre, M., López-Pernas, S., Saqr, M., Daniels, M., Pears, A.: A Scientometric Journey Through the FIE Bookshelf: 1982–2020. In: 2021 IEEE Frontiers in Education Conference (FIE), pp. 1–9 (2021). https://doi.org/10.1109/FIE49875.2021.9637209

6. Austing, R.H., Barnes, B.H., Bonnette, D.T., Engel, G.L., Stokes, G.: Curriculum '78: Recommendations for the undergraduate program in computer science — a report of the ACM curriculum committee on computer science. Communications of the ACM **22**(3), 147–166 (1979). DOI http://doi.acm.org/10.1145/359080.359083
7. Becker, B.A., Settle, A., Luxton-Reilly, A., Morrison, B.B., Laxer, C.: Expanding opportunities: Assessing and addressing geographic diversity at the SIGCSE Technical Symposium. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, pp. 281–287. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3408877.3432448
8. Colizza, V., Flammini, A., Serrano, M.A., Vespignani, A.: Detecting rich-club ordering in complex networks. Nature Physics **2**(2), 110–115 (2006). https://doi.org/10.1038/nphys209
9. Daniels, M., Berglund, A., Pears, A.: A case study: The Uppsala Computing Education Research Group (UPCERG). In: Past, Present and Future of Computing Education Research: A Global Perspective. Springer (2023)
10. Daniels, M., Pears, A., Malmi, L., Simon: What is Computing Education Research (CER)? In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
11. Djamasbi, S., Galletta, D.F., Nah, F.F.H., Page, X., Robert Jr., L.P., Wisniewski, P.J.: Bridging a bridge: Bringing two HCI communities together. In: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, CHI EA '18, pp. 1–8. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3170427.3170612
12. Fincher, S.A., Robins, A.V.: An important and timely field. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, pp. 1–8. Cambridge University Press (2019). https://doi.org/10.1017/9781108654555.001
13. Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B., Thomas, L.: A Multi-national Study of Reading and Tracing Skills in Novice Programmers. SIGCSE Bull. **36**(4), 119–150 (2004). http://doi.acm.org/10.1145/1041624.1041673
14. López-Pernas, S., Saqr, M., Apiola, M.: Scientometrics: A Concise Introduction and a Detailed Methodology for the Mapping of the Scientific Field of Computing Education Research. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
15. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory programming: A systematic literature review. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2018 Companion, pp. 55–106. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3293881.3295779
16. Ma, A., Mondragón, R.J., Latora, V.: Anatomy of funded research in science. Proceedings of the National Academy of Sciences **112**(48), 14760–14765 (2015). https://doi.org/10.1073/pnas.1513651112.
17. Malmi, L., Simon, Sheard, J., Kinnunen, P., Sinclair, J.: The Evolution of Computing Education Research: A Meta-Analytic Perspective. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
18. McCartney, R., Sanders, K.: ITiCSE working groups and collaboration in the computing education community. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2018, pp. 332–337. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3197091.3197143
19. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., Laxer, C., Thomas, L., Utting, I., Wilusz, T.: A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students. In: Working group reports from ITiCSE on Innovation and technology in computer science education, ITiCSE-WGR '01, pp. 125–180. ACM, New York, NY, USA (2001). http://doi.acm.org/10.1145/572133.572137
20. McMillan, D., Chavis, D.: Sense of community: A definition and theory. Journal of Community Psychology **14**, 6–23 (1986). https://doi.org/10.1002/1520-6629(198601)14:13.0.CO;2-I

21. Miró Julià, J., López, D., Alberich, R.: Education and research: Evidence of a dual life. In: Proceedings of the Ninth Annual International Conference on International Computing Education Research, ICER '12, pp. 17–22. Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2361276.2361281
22. Nicholls, P.T.: Bibliometric modeling processes and the empirical validity of Lotka's law. Journal of the American Society for Information Science **40**(6), 379–385 (1989)
23. Qiu, J., Zhao, R., Yang, S., Dong, K.: Author Distribution of Literature Information: Lotka's Law, pp. 145–183. Springer Singapore, Singapore (2017). https://doi.org/10.1007/978-981-10-4032-0_6
24. Rankin, Y.A., Thomas, J.O., Erete, S.: Real talk: Saturated sites of violence in cs education. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, pp. 802–808. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3408877.3432432
25. Root-Bernstein, R.: STEMM education should get "HACD". Science **361**(6397), 22–23 (2018). https://doi.org/10.1126/science.aat8566.
26. Settle, A., Becker, B.A., Duran, R., Kumar, V., Luxton-Reilly, A.: Improving Global Participation in the SIGCSE Technical Symposium: Panel. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20, pp. 483–484. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3328778.3366979
27. Shackelford, R., McGettrick, A., Sloan, R., Topi, H., Davies, G., Kamali, R., Cross, J., Impagliazzo, J., LeBlanc, R., Lunt, B.: Computing curricula 2005: The overview report. In: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '06, pp. 456–457. Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1121341.1121482
28. Simon: Informatics in Education and Koli Calling: a Comparative Analysis. Informatics in Education **8**(1), 101–114 (2009)
29. Simon: Emergence of computing education as a research discipline. Ph.D. thesis, Aalto University School of Science (2015)
30. Simon: A Picture of the Growing ICER Community. In: Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16, pp. 153–159. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2960310.2960323
31. Simon: The Koli Calling Community. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli Calling '16, pp. 101–109. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2999541.2999562
32. Simon: Twenty-four years of ITiCSE authors. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, pp. 205–211. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3341525.3387387
33. Simon: Twenty-two years of ace. In: Proceedings of the Twenty-Second Australasian Computing Education Conference, ACE'20, pp. 203–210. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3373165.3373188
34. Williams, T.L.: The lives of hidden figures matter in computer science education. Communications of the ACM **65**(2), 20–22 (2022). https://doi.org/10.1145/3506577
35. Wilson, J., Aiken, R., Katz, I.: Review of animation systems for algorithm understanding. SIGCSE Bull. **28**(SI), 75–77 (1996). https://doi.org/10.1145/237477.237534
36. Wu, C.C., Dale, N.B., Bethel, L.J.: Conceptual models and cognitive learning styles in teaching recursion. SIGCSE Bull. **30**(1), 292–296 (1998). https://doi.org/10.1145/274790.274315
37. Zhang, J., Luxton-Reilly, A., Denny, P., Whalley, J.: Scientific collaboration network analysis for computing education conferences. In: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE '21, pp. 582–588. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3430665.3456385
38. Zhou, S., Mondragon, R.: The rich-club phenomenon in the internet topology. IEEE Communications Letters **8**(3), 180–182 (2004). https://doi.org/10.1109/LCOMM.2004.823426

# The Venues that Shaped Computing Education Research: Dissemination Under the Lens


**Mikko Apiola, Sonsoles López-Pernas, and Mohammed Saqr**

## 1 Introduction

A crucial part of science is dissemination. All scientific disciplines have their own cultures and habits of publication, and established venues of dissemination, with their own rules and practices. These dissemination venues act as gatekeepers to steer the development of a scientific discipline by deciding which research is valued and which is not. In the highly competitive and metrics-oriented academic milieu of today, dissemination and publication habits are more important than ever. In computing education research (CER), previous research of dissemination practices have focused mostly on the highly dedicated and most influential venues [5]. In this chapter, we provide a holistic exploration into the dissemination practices of CER, by zooming in on a total of 1523 publication venues where CER typically gets published, in the four categories of dedicated and non-dedicated journals and conferences.

By using Scopus metadata and modern scientometrics methods [20], our analysis focuses on central characteristics of publication venues: a small core of highly-dedicated venues, which together publish a remarkable share of CER, and a diverse range of publication venues in neighboring disciplines of education, engineering education, and general computer science. Our analysis demonstrates the conference-oriented publication tendency of CER, where the highly dedicated publication outlets dominate the numbers of publications, while journal articles are in significantly lower numbers, in direct contrast with other disciplines such as engineering education research (EER). Analysis shows that while authors of CER tend to publish much more often in conferences, journal articles receive significantly more citations, with the exception of ITiCSE Working Groups, a special form of collaborative

M. Apiola (✉) · S. López-Pernas · M. Saqr
University of Eastern Finland, Joensuu, Finland
e-mail: mikko.apiola@uef.fi; sonsoles.lopez@uef.fi; mohammed.saqr@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_7

publishing [27]. This chapter provides insights into the diversity and differences of preferred research topics both between and within the dedicated and non-dedicated categories. For example, while the non-dedicated venues focus more on game-based learning, the dedicated venues are stronger in research on introductory courses. We conclude the chapter with a discussion about the future of dissemination in CER. More specifically, in this chapter, we answer the following research questions:

1. In which publication outlets does CER get published?
2. How do the themes of research differ between dedicated and non-dedicated publication outlets of CER?

The chapter is structured as follows. First, Sect. 2 shows how dissemination of CER has evolved in the four categories of publication outlets, and provides basic characteristics of the four categories with regards to numbers of publications, citation rates, topics of research, and top authors. Second, Sect. 3 zooms in on the dedicated venue's category, introduces the central venues inside the category, their publication metrics, top keywords, and geography. Third, Sect. 4 introduces the non-dedicated venues in a similar fashion. Discussions are provided in Sect. 5.

## *1.1 A Brief Summary of Data and Methods*

This chapter is based on a scientometric analysis of a comprehensive dataset of CER publications, obtained from Scopus database in January 2022. The entire process of data retrieval, screening, pre-processing, analysis and integrity checking is explained in Chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" of this book [20]. The data and methods can be briefly summarised as follows. The principles of PRISMA-S (Preferred Reporting Items for Systematic reviews and Meta-Analyses) literature search extension was applied to retrieve data from two sources. First, a full download from a list of dedicated publication outlets (conference series and journals) that focus exclusively on CER was done. The list of dedicated venues was compiled after meetings between seven researchers (the editorial team of this book), until a consensus was reached. Second, a keyword query search from all other publication venues was performed, after which these two sets of articles were joined. After exhaustive screening, cleaning, and pre-processing, the result set contains a total of 16,863 articles.

Due to inconsistencies in recording publication outlet names in Scopus, extensive cleaning and manual checkups were required to reliably recognise different publication outlets. For example, many publication outlets, especially conference series, use slightly different names in different years, requiring much manual work to merge all instances from different years into a complete series. The cleaning process made use of the OpenRefine tool to automatically detect similar publication venue names, which was followed by exhaustive manual checking and fixing of venue names. This

**Table 1** Four categories of CER publication outlets with total number of venues and number of articles in each category

| | N of venues | N (%) of papers |
|---|---|---|
| Dedicated journals | 3 | 776 (4.6%) |
| Dedicated conference series and magazines | 13 | 11,148 (66.1%) |
| Non-dedicated journals | 572 | 1407 (8.3%) |
| Non-dedicated conference series | 935 | 3532 (21%) |
| | 1523 | 16,863 (100%) |

process reduced the amount of distinct publication venues from 2358 into 1523. For the purposes of this chapter, publication venues were further divided to four broad categories, including (1) journals exclusively dedicated in publishing CER, (2) conference series exclusively publishing CER, (3) journals that publish CER together with other topics of research, and (4) conference series that publish CER together with other topics of research. The magazines SIGCSE Bulletin and ACM Inroads are included in the category 2, because they publish CER papers, and setting a category with only two venues would not have made sense. It is good to note that in earlier times, SIGCSE Bulletin republished earlier proceedings of the SIGCSE Technical Symposium. At the inception of ACM Inroads, in 2010 the SIGCSE bulletin transformed into an electronic newsletter. Identification of the dedicated journals and conferences was done based on expert opinions of the editorial team of this book [20]. The rest of the venues were categorised to non-dedicated journals and non-dedicated conferences by using keyword-based identification, and examining all venue names manually. This process resulted in four categories of publication outlets, summarised in Table 1. Appendix lists all dedicated venues, and all those non-dedicated venues, which have published five or more articles. The data is not without limitations, which are discussed in Sect. 5.3.

## 2 Evolution of Central Venues in Time: Four Categories

Some of the early venues for airing ideas about computing education were International Federation for Information Processing's (IFIP) TC-3 (established in 1963) and Association for Computing Machinery (ACM)'s SIGCSE Technical Symposium (Special Interest Group in Computer Science Education, established in 1968), both of which arranged their first conferences in 1970. Common journals for dissemination have been, e.g. International Journal of Man-Machine Studies, the Computer Journal, Computers & Education, and Communications of the ACM. Other common early venues include the Psychology of Programming Interest Group (PPIG) and the Empirical Studies of Programmers (ESP) [15, 36]. In the decades before 2000s, countless course descriptions were published, but also distinct examples of empirical research started to appear [15, 32].

Over the years, CER evolved from primarily being a forum for educators to exchange their best practices, towards a methodologically and theoretically-driven, established discipline of science [44]. Especially after the 2000s, several capacity-building initiatives were launched to train CER researchers, whose backgrounds were often in computer science rather than in learning sciences [8, 9, 12, 30, 31]. Before the 2000s, venues such as the SIGCSE Technical Symposium primarily published experience reports, and the gatherings were often called *swap-meets*, referring to the exchange of pedagogical ideas between the attending computing teachers [19]. Especially after the 2000s, venues of dissemination started to place more strict requirements to the papers they accepted [33, 44], and new venues such as ICER (the International Computing Education Research Conference), established in 2005, started to expect theory-informed and methodologically rigorous articles [45].

Since the main focus of CER is the study of learning and teaching of computing, it is natural that CER has been influenced by the learning sciences [25]. Integration with learning sciences has brought fresh perspectives and ideas, and a repertoire of new methods and means to conduct previously unseen types of research [25]. Building research designs that are informed by learning-theoretical contributions have enabled new research approaches on educational, social, cultural, and historical contexts in CER, involving e.g., factors of motivation, attitudes, values, dispositions, or mindsets, and the means to design and evaluate related practical innovations and interventions [25]. Such new approaches have enriched CER of describing specific instructional designs and studying their impact to e.g. course grades. Many reviews and meta-analyses have started to focus on how learning theories are used and applied in CER [22–24, 51]. On the other hand, while CER researchers are often computer scientists and engineers, CER has always had the unique strength to design and develop new tools and learning technologies: tools-research has always been a fundamental part of CER [44].

In an editorial from 2015, editors of ACM Transactions on Computing Education (TOCE), a premier venue to disseminate CER, noted that an overwhelming majority of articles received a "major revisions" decision, resulting from CS educators lacking the expertise in required educational inquiry to go beyond experience reports, and insufficient rewards from their home institutions for the required time and effort [55]. In a recent conversation between the current and immediate past editors, the "gatekeepers", of the two premier publication outlets in CER: ACM TOCE and Computer Science Education (CSE), the editors highlighted several points [13]. Many articles are still submitted—and get rejected—on the basis that they are essentially experience reports with anecdotal evidence, or papers that introduce a technological tool without a proper evaluation, often with the assumption that: *"This is a cool idea, surely the contribution is obvious"*. In addition, a substantial proportion of articles are on first courses, mostly CS1 or CS2 [13]. While experience reports without proper evaluation often get desk rejected, the audience is also shifting from educational practitioners to active researchers on the field of CER. The editors wished to see more papers that build CER-specific theories, papers on evaluation instruments and innovations, papers that connect to socio-cultural theories, embodied cognition, and in general, approaches from other fields [13].

One future desire was to widen the repertoire of research methods in CER, by incorporating, e.g., deep ethnographies and other new avenues for research [13].

To summarize the development, in recent decades, CER has made significant advances in broadening its research culture by integrating with learning sciences, learning to appreciate theoretical contributions, increasing its methodological rigor, strengthening its branch of tools-research, and by bringing in a wider repertoire of research designs [18, 23, 25]. Over the years, the evolution of the culture of CER has been influenced by the CER researchers' roots in computer science, increased integration with learning sciences, close connections with EER [21], and the unique ability of CER to design and implement new technologies. These characteristics of CER reflect directly into the evolution in dissemination practices of CER.

Over the time, CER has developed its own unique set of dedicated conferences and journals, which have experienced their own evolution from first publishing mostly experience reports, and then later, especially after the 2000s, by starting to demand increased research rigor. Meanwhile, while CER has been increasingly building bridges with disciplines such as EER and educational research, this also reflects in diversity in the publication outlets. Over time, CER has been published in conferences and journals that are dedicated exclusively to CER, and conferences and journals in neighboring disciplines of e.g. learning sciences and engineering education (non-dedicated). Figure 1 shows the trends of CER published in these four broad categories over time.

## *2.1 Shares of Articles and Top Venues*

Figure 1 shows all of the distinct publication venues of CER ($N = 1523$) in our data, categorised into (1) conferences and magazines that publish exclusively CER, (2) journals that publish exclusively CER, (3) conferences that are not dedicated to CER only, but publish also in other areas, such as learning sciences, EER, or general computer science, (4) non-dedicated journals in other fields, such as EER, education, or general CS. The largest category by far where CER gets disseminated are the dedicated conferences and magazines ($N = 13$), accounting for 66.1% of articles in the dataset. The second largest category is the non-dedicated conferences ($N = 935$), which together publish some 21% of articles. Dedicated journals ($N = 3$) publish some 4.6% of CER, while the Non-Dedicated Journals ($N = 572$) publish some 8.3% of articles in our dataset (see Table 1). The publication outlets in the four categories are listed in Appendix. These four categories resemble the essential venues of dissemination in CER. While a small number of core dedicated venues publish exclusively CER, a large number of non-dedicated venues publish CER together with research in neighboring disciplines, such as education, EER, and computing.

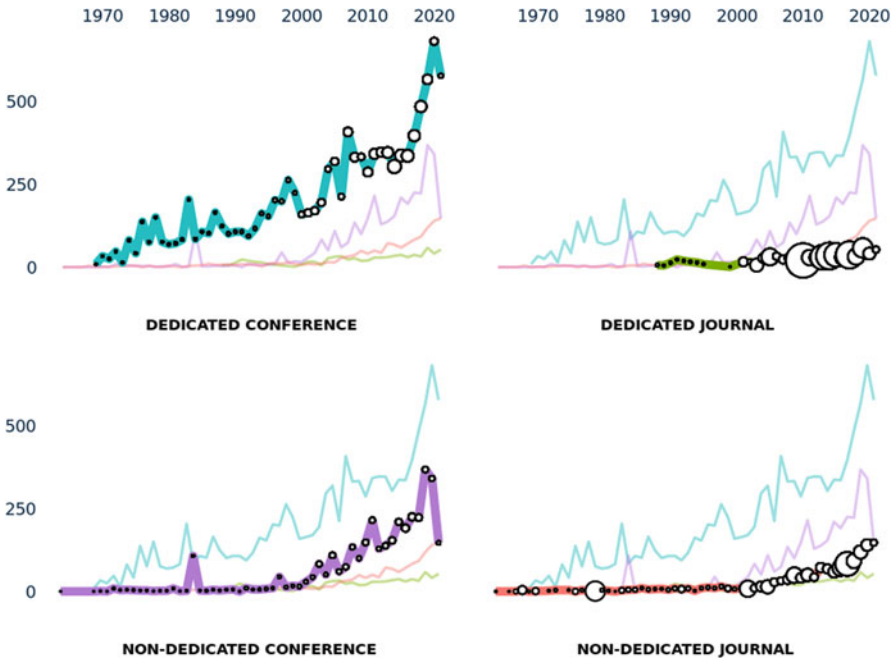


**Fig. 1** Four categories of CER venues in Time: Dedicated Conferences and Magazines, Dedicated Journals, Non-Dedicated Conferences, Non-Dedicated Journals. The y-axis represents the number of publications, and white bubbles represent numbers of citations. Each of the four categories has been assigned a unique color

## *2.2 Citations*

We investigated the mean number of citations between the four categories. The results show that journal articles in the dedicated venues category are the most cited ($\bar{x} = 16.77$), followed by articles published in the non-dedicated journals ($\bar{x} = 10.85$). For articles published in the dedicated conferences and magazines, the mean number of citations was: ($\bar{x} = 7.99$), and for the non-dedicated conferences: ($\bar{x} = 4.18$). All differences between the categories were statistically significant (Kruskall-Wallis, $p < .01$ for all comparisons). The findings confirm previous findings that while CER researchers tend to publish in conferences, journal articles in CER are significantly better cited [5]. The findings also add to previous research that research published in the dedicated venues is, on average, significantly better cited than research in the non-dedicated venues.

### 2.3 Research Topics (Keywords)

Second, we analysed differences in the top keywords between the dedicated venues (journals and conferences combined) and the non-dedicated venues (journals and conferences combined). While many previous works have investigated research in the dedicated CER venues (for a summary, see: [5, 44]), no previous research that we are aware of has investigated the research themes published outside of the dedicated CER venues. The top 20 keywords per category are listed in Table 2. We conducted a closer inspection into a combined set of top keywords from both the dedicated and non-dedicated venues to look into dominance of top keywords between these two categories. In Fig. 2, the proportion of top keyword use between the dedicated and non-dedicated venues is shown, with the color blue indicating the stronger category, and color red indicating the weaker category. The strength of the color represents the statistical significance of the relative proportion of keyword use between the categories, as determined by the $\chi^2$ test. For example, the relative frequency of *informatics education* as a keyword is significantly higher in the non-dedicated category, than in the dedicated category, and this difference is significant as observed by the high residual between the observed and expected values in a $\chi^2$ test (see Fig. 2).

In the non-dedicated venues, stronger keywords with highest significance include *higher education*, *educational technology*, *informatics education*, *STEM*, *computer science curriculum*, *engineering*, *information technology*, and *game-based learning*. This resembles the nature of the non-dedicated venues in publishing broadly on educational technology, informatics, information technology, and engineering. On the other hand, in the dedicated venues, stronger keywords include research in introductory courses (*CS1/CS2*), *object-oriented programming*, *novice programmers*, *java*, *design*, and *automatic assessment*. Table 2 shows that programming education, CS1/CS2, and K-12 computing education are strong in all four categories.

### 2.4 Authors

The top 10 authors per each of the four categories are listed in Table 3. Many authors in the dedicated venues, including journals and conferences, are well-known prominent CER authors (see Chapter "The Hands that Made Computing Education Research: Top Authors, Networks, Collaboration and Newcomers" [3]). Many top authors in the non-dedicated venues are also well-known contributors in the dedicated CER venues, but differences are also seen with tendencies to publish more on engineering education, or mixing CER with, e.g., ICT4D in the case of Jarkko Suhonen from the University of Eastern Finland, or in the diverse works of Andreas Zendler (Top 1 in Non-dedicated Journals category). The top authors in the dedicated venues are analysed with more depth in Chapter "The Hands that Made Computing Education Research: Top Authors, Networks, Collaboration and Newcomers" of this book. One must note that Scopus, as well as ACM Digital

**Table 2** Most frequently used (top) 20 keywords per category

| | Dedicated conf. and magaz. | Non-dedicated conference | Dedicated journal | Non-dedicated journal |
|---|---|---|---|---|
| 1 | CS1/CS2 | PROGRAMMING EDUCATION | K-12 | K-12 |
| 2 | K-12 | K-12 | PROGRAMMING EDUCATION | GENDER AND DIVERSITY |
| 3 | PROGRAMMING EDUCATION | GAME-BASED LEARNING | COMPUTATIONAL THINKING | PROGRAMMING EDUCATION |
| 4 | GENDER AND DIVERSITY | SOFTWARE ENGINEERING EDUCATION | E-LEARNING | DESIGN |
| 5 | COMPUTATIONAL THINKING | COMPUTATIONAL THINKING | GENDER AND DIVERSITY | CS1/CS2 |
| 6 | PEDAGOGY | GENDER AND DIVERSITY | GAME-BASED LEARNING | ASSESSMENT |
| 7 | ASSESSMENT | CS1/CS2 | INFORMATICS EDUCATION | COMPUTATIONAL THINKING |
| 8 | SOFTWARE ENGINEERING EDUCATION | E-LEARNING | SOFTWARE ENGINEERING EDUCATION | COLLABORATIVE LEARNING |
| 9 | VISUALIZATION | ASSESSMENT | EDUCATIONAL TECHNOLOGY | HUMAN FACTORS |
| 10 | E-LEARNING | ENGINEERING | HIGHER EDUCATION | EXPERIMENTATION |
| 11 | ACTIVE LEARNING | COMPUTER SCIENCE CURRICULUM | COMPUTER SCIENCE CURRICULUM | SOFTWARE ENGINEERING EDUCATION |
| 12 | OBJECT-ORIENTED PROGRAM-MING | VISUALIZATION | VISUALIZATION | BROADENING PARTICIPATION) |
| 13 | COLLABORATIVE LEARNING | COLLABORATIVE LEARNING | COLLABORATIVE LEARNING | PEDAGOGY |
| 14 | NOVICE PRO-GRAMMERS | PEDAGOGY | CS1/CS2 | OBJECT-ORIENTED PROGRAMMING |
| 15 | JAVA | STEM | ENGINEERING | LANGUAGE |
| 16 | AUTOMATIC ASSESSMENT | EDUCATIONAL TECHNOLOGY | STEM | HIGHER EDUCATION |
| 17 | GAME-BASED LEARNING | ACTIVE LEARNING | PROGRAMMING LANGUAGES | NOVICE PROGRAMMERS |
| 18 | DESIGN | GAME DESIGN/DEVELOPMENT | ASSESSMENT | VISUALIZATION |
| 19 | MOTIVATION | INFORMATICS EDUCATION | COMPUTER-AIDED INSTRUCTION | PAIR PROGRAMMING |
| 20 | SCRATCH | INFORMATION TECHNOLOGY | ROBOTICS | MISCONCEPTION |

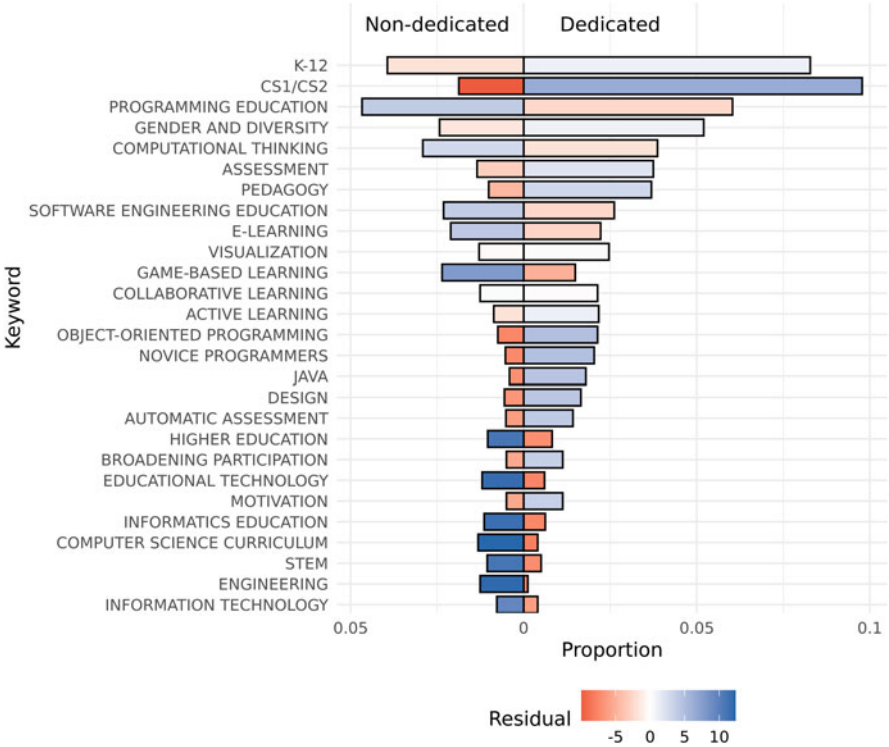


**Fig. 2** Strength of top keywords in non-dedicated and dedicated venue-categories. The list includes top 20 keywords from both of these categories. The x-axis represents the strength of keyword presence in frequency (blue indicates stronger keyword presence than red). The strength of color indicates the size of the residual between observed and expected values in a $\chi^2$ test (stronger color means more statistical significance). For example, "informatics education" is a significantly stronger keyword in the non-dedicated venues than in the dedicated venue's category

Library, may classify editorials as journal papers, which may have an influence in the results in Table 3. This, and other limitations, are discussed with more depth in Sect. 5.3.

## 3 Dedicated Venues

The core dedicated venues resemble the essence of CER. Many previous analyses have targeted the core venues of dissemination of CER, and focused on collaboration and geographical distribution [2, 7, 39, 46, 48, 59], tendencies of accepting newcomers [3, 26, 29, 47], patterns of authorship [44–48], core authors and communities [26], use of theories and learning, among other approaches [5].

**Table 3** Authors per category

| | Dedicated conference | Non-dedicated conference | Non-dedicated journal | Dedicated journal |
|---|---|---|---|---|
| 1 | SIMON (76) | IMPAGLIAZZO J (27) | ZENDLER A (14) | MCCARTNEY R (13) |
| 2 | SHEARD J (70) | BOYER KE (19) | TSAI C-W (12) | TENENBERG J (12) |
| 3 | GUZDIAL M (66) | HISLOP GW (16) | KARNALIM O (12) | ARMONI M (12) |
| 4 | LUXTON-REILLY A (65) | WIEBE E (13) | GUZDIAL M (11) | BEN-ARI M (9) |
| 5 | LISTER R (62) | DANIELS M (13) | LEE Y (10) | MALMI L (9) |
| 6 | SIMON B (61) | HUBWIESER P (13) | SUHONEN J (9) | EDWARDS S (8) |
| 7 | MCCARTNEY R (59) | LESTER J (12) | VELÁZQUEZ-ITURBIDE JÁ (9) | RYOO JJ (7) |
| 8 | EDWARDS S (56) | LEE J (12) | SOH L-K (9) | GAL-EZER J (7) |
| 9 | PORTER L (54) | KAKESHITA T (12) | VON WANGENHEIM CG (8) | KOLIKANT YB-D (6) |
| 10 | RODGER SH (53) | MOTT B (11) | KLAUDT D (8) | KAFAI YB (6) |

In the following we will briefly summarise the general characteristics of these top venues of dissemination and zoom in on the differences within these top venues.

### *3.1 Core Conferences and Magazines*

There are a total of $N = 13$ venues in the dedicated conferences and magazines category in our data. The top 10 are visualised in Fig. 3. The top venues include ACM's Special Interest Group in Computer Science Education (SIGCSE)'s Technical Symposium, which was started in 1970 [15, 53]. This category also includes SIGCSE Bulletin. Other venues in this category include ITiCSE (Innovation and Technology in Computer Science Education) which was founded in 1996, ACE (Australasian Computing Education Conference) (1996), the Finnish Koli Calling Conference (2001), and ICER (International Computing Education Research Conference) (2005) [53]. The premier CER outlets for K-12 computing education are the ISSEP (The International Conference on Informatics in Schools) which was launched in 2005 and WIPSCE (Workshop in Primary and Secondary Computing Education). The category also includes the new Computer Science Education Research Conference (CSERC), Conference on Computing Education Practice (CEP), ACM's new Global Computing Education Conference (COMPED), as well as the UK and Ireland Computing Education Research Conference (UKICER). These conferences together form the core conferences that publish CER.

Out of all the dedicated venues, the SIGCSE Technical Symposium is by far the largest. In 1970, SIGCSE launched its Technical Symposium, initially focused as a forum for teachers to exchange best practice [1, 15, 53]. SIGCSE Technical Symposium is by far the largest venue for publishing CER with regards to the volume of published papers, publishing some 34% of articles in the category of dedicated conferences and magazines. The scale of submissions has evolved from 18 papers in its first year [57] to some 297 papers in 2021, and in recent years the symposium has attracted more than 1000 annual attendees [15]. One early categorisation of papers in SIGCSE was that of Valentine [56], who invented the categories: *Marco Polo, Tools, Experimental, Nifty, Philosophy and John Henry*. For almost three decades since the inception of SIGCSE Technical Symposium, the paper submission and review process was handled through postal mail, and the reviewers were all from North America [57]. Electronic submission systems became common after 2000. Analyses have shown popularity of keywords related to introductory programming, and recent increase of K-12, computational thinking and gender diversity [6, 19]).

The Finnish Koli Calling International Conference for Computing Education Research (est. 2001) was organized for the first time in 2001 [46]. Koli Calling was formerly called The Baltic Sea Conference on Computing Education Research. The Koli Calling conference is arranged annually at a resort in the Koli National Park in Eastern Finland [4]. Since 2001, the community of Koli Calling has evolved and diversified, with increased international collaboration [34, 41, 42, 46]. Indeed,

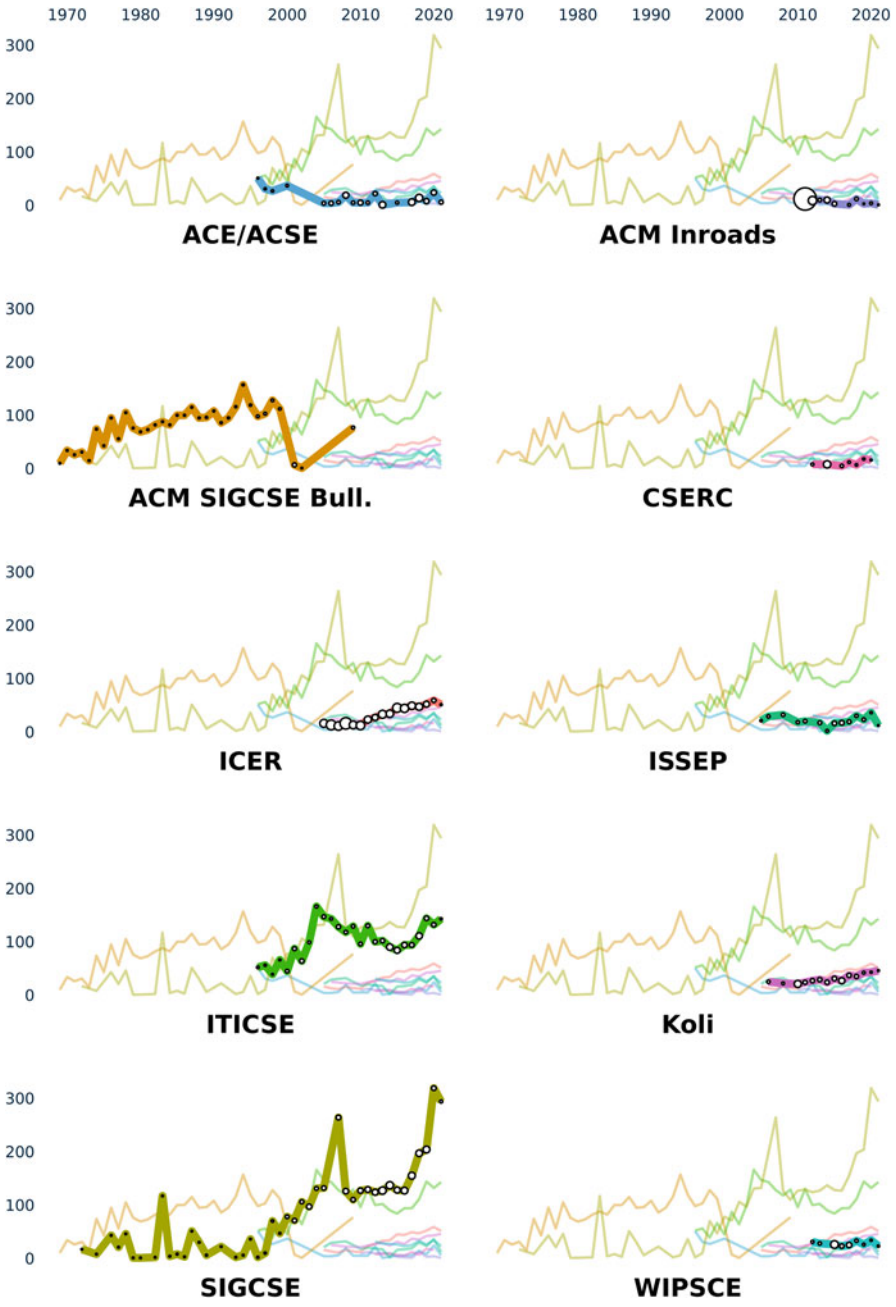


**Fig. 3** Top ten venues in the dedicated conferences and magazines -category. The y-axis represents volume of publications. Each figure shows the same venues, each assigned with a unique color, with the bolded one representing the venue in question. White bubbles represent numbers of citations

Koli Calling has evolved from a small gathering of local computing teachers to an internationally acknowledged conference; by 2006, the number of Finnish authors had shrunk by half, and in 2020, the largest share of authors were from USA [4].

Innovation and Technology in Computer Science Education (ITiCSE, est. 1996) conference is the European counterpart of the SIGCSE Technical Symposium, a premier international CER conference, arranged annually in varying locations in Europe since 1996 [47, 49]. The first ITiCSE was arranged in Barcelona, Spain, chaired by Boots Cassell. One specialty of ITiCSE is the ITiCSE Working Groups (ITiCSE-WGR), which have been part of the conference since its beginning in 1996 [26, 28]. The ITiCSE-WGR provides a unique opportunity for attendees to collaborate on a research project of common interest together with other conference attendees, and analysis shows that the ITiCSE-WGR, as a publication venue, is more "extroverted", more accepting to newcomers, as compared to some other CER venues [26]. A recent paper analysed all full papers (1996–2019, $N = 1295$), and Working Group Reports ($N = 129$) [49]. ITiCSE has also published shorter papers, posters, and doctoral consortium contributions [49]. Findings show, e.g. that nearly 40% of ITiCSE full papers have concerned programming education, and a clear increasing trend of research papers is seen, from less than 20% before 2000, to some 40–70% after 2010, and over 80% in 2019 [49].

International Computing Education Research Conference (ICER, est. 2005) became the third SIGCSE conference after SIGCSE Technical Symposium and ITiCSE [45]. The ad-hoc nature of CER, manifesting in a lot of articles being about describing course contents in an anecdotal matter, also known as the Marco Polo paper "I went there and I saw this" [56], started to attract a lot of attention after the turn of the millennium [30, 31]. The leading idea in launching ICER was to setup a venue, which would accept, in contrast to venues such as SIGCSE Technical Symposium, and ITiCSE, only rigorous research papers [45]. The first ICER in 2005 was arranged in Seattle, WA, chaired by Richard Anderson, Sally Fincher and Mark Guzdial, hosting some 16 accepted papers [45]. The Australasian Computing Education Conference (ACE, est. 1996) was launched in 1996, and has published papers during the timespan 1996–2020 [43, 48]. The first ACE was arranged in 1996 in Sydney, Australia, chaired by John Rosenberg and Alan Fekete of Sydney University. In the beginning, ACE was mostly a regional forum for presenting papers that before its existence might have been submitted to ITiCSE or SIGCSE Technical Symposium [43]. Classifications according to Simon's system [40] show e.g. that the proportion of research papers has risen from below 20% in 1996 to some steady 60–80% during 2009–2019, and over 80% in 2020 [48]. Australia and New Zealand have been the dominant countries in ACE.

The International Conference on Informatics in Schools (ISSEP) was launched in 2005 in Klagenfurt, Austria, as a forum for researchers and practitioners of CER both in primary and secondary education. No previous research that we are aware of, has reviewed or scientometrically analysed the publications in ISSEP. Workshop in Primary and Secondary Computing Education (WiPSCE) has its roots in the German computing education community, and aims to exchange research and practice of CER in primary and secondary education. The 2012 workshop (7th WiPSCE),

arranged in Hamburg, Germany, is the first one with data available in Scopus. WiPSCE is run in cooperation with ACM SIGCSE, and is hosted in countries across Europe. New conferences in CER include computer science education research conference (CSERC), conference on computing education practice (CEP), ACM Global Computing Education Conference (COMPED), and UK and Ireland Computing Education Research Conference (UKICER).

## 3.2 Core Journals

The two premier journals that are dedicated to publishing CER are the *Computer Science Education (CSE)*, established in 1988 [11, p. 1], and *ACM Transactions on Computing Education (TOCE)*, which was launched in 2009 [54]. CSE is a premier venue that publishes CER, with a focus on all aspects of CER, welcoming variety of research methods, but requiring rigorous use of methods to address research questions posed by submitting authors. ACM TOCE was formerly known as *Journal on Educational Resources in Computing (JERIC)*, which was launched in 2001. JERIC had a special focus on educational resources such as educational technology in computing education [54]. The editorial of JERIC from 2001 reads: *"ACM JERIC is an electronic publication providing access to high quality, archival resources suitable for support of computing education. Resources include scholarly articles with wide applicability and potential impact as well as multimedia and visualization works, laboratory materials, and other digital objects of practical use supporting learning in the computing field."* [10]. In 2009, the ACM JERIC journal was rebranded as ACM Transactions on Computing Education (TOCE) [54]. With the name change in 2009, the editorial board wished to broaden the focus from computing-based teaching tools to make the journal appeal to wider range of computing educators [54]. Figure 4 visualises the publication metrics and citations in these three core dedicated journals. Out of these three, ACM Transactions on Computing Education is the best cited, with an average of 21.52 citations per article in Scopus.

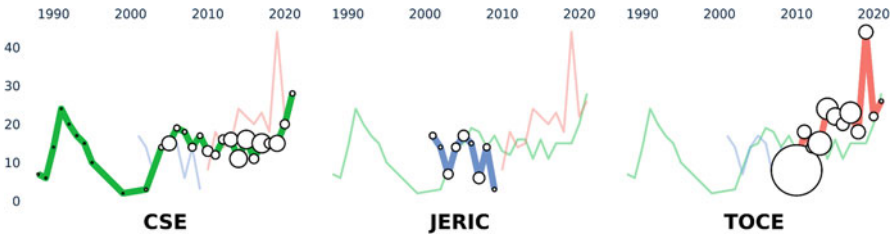


**Fig. 4** The three top dedicated journals (with highest numbers of publications). The y-axis represents volume of publications. Each figure shows the same three venues, each assigned with a unique color, with the bolded one representing the venue in question. The white bubbles represent numbers of citations

### 3.3 Keywords and Countries

Previous analyses of keywords in the central dedicated venues of CER reveal that leading themes of research have been learning and teaching programming, introductory courses in computer science, K-12 and computational thinking [5]. Other top researched areas of research in CER include learning analytics and educational data mining, gender and diversity, automatic assessment, software engineering, e-learning, visualisation, collaborative learning, and others [5]. The dominance of teaching programming as a topic in CER has been observed in numerous previous analyses, too [2, 13, 44, 48, 49, 56], as well as the high increase of research on computational thinking within CER [38]. Figure 5 presents top keyword use in the common dedicated venues that publish CER, including the three dedicated journals and top 10 conferences listed in Appendix.

Figure 5 shows the coverage of the top 20 keywords among the top dedicated venues, with darker color showing higher coverage. Figure 5 clearly shows that CS1/CS2, referring to research on introductory computer science courses, is within

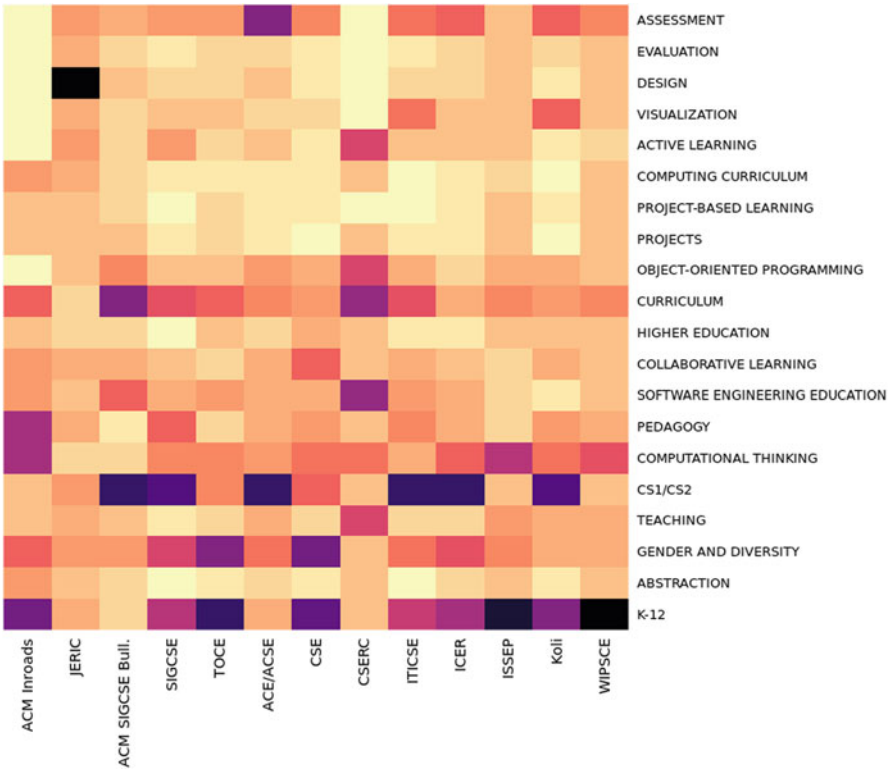


**Fig. 5** Heatmap of Top Dedicated Venues (Top 10 Conferences, and Top 3 Journals) and Top 20 Keywords. Darker color indicates stronger keyword presence

the top keywords in many venues, including SIGCSE Bulletin, SIGCSE Technical Symposium, ACE, ITiCSE, ICER, and Koli Calling. K-12 is a top ranking keyword in ISSEP, WIPSCE, ACM Transactions on Computing Education, and Computer Science Education. The top five keywords in these central venues are as follows:

- ICER (*CS1/CS2, K-12, Programming Education, Gender and Diversity, Assessment*),
- ITiCSE (*CS1/CS2, programming education, K-12, curriculum, e-learning*),
- SIGCSE Technical Symposium (*CS1/CS2, K-12, gender and diversity, curriculum, pedagogy*),
- Koli Calling (*programming education, CS1/CS2, K-12, assessment, visualization)*,
- ISSEP (*K-12, Informatics Education, Computational Thinking, Programming Education, Bebras*),
- ACE (*CS1/CS2, programming education, assessment, novice programmers, gender and diversity*),
- WIPSCE (*K-12,Programming Education,Computational Thinking, Scratch, Curriculum*),
- CSE (*K-12, programming education, gender and diversity, CS1/CS2, collaborative learning*),
- ACM TOCE (*K-12, gender and diversity, programming education, curriculum, CS1/CS2*),
- ACM JERIC (*design, education, experimentation, human factors, language*), and
- SIGCSE Bulletin (*CS1/CS2, Curriculum, Programming Education, Java, Software Engineering Education*).

With regards to top countries, many of the dedicated venues are dominated by North America. The largest single contributing country with regards to author contributions is USA in the following venues: SIGCSE Technical Symposium, SIGCSE Bulletin, ITiCSE, ACM Inroads, Journal of Educational Resources in Computing (JERIC), ACM Transactions on Computing Education (TOCE), Computer Science Education (CSE), and ICER. The exceptions to USA-dominance are: ACE, with the largest share of authors from Australia, CSERC (Netherlands), ISSEP (Germany), WIPSCE (Germany), and Koli Calling (Finland).

## 4 Non-Dedicated Venues

### *4.1 Conferences*

The category of non-dedicated conferences includes some 935 distinct publication venues, which have together published some 3532 CER articles, accounting for a share of 21% of CER in our dataset. The top 10 publication outlets (Fig. 6) in this category include the ASEE/IEEE Frontiers in Education, launched in 1970

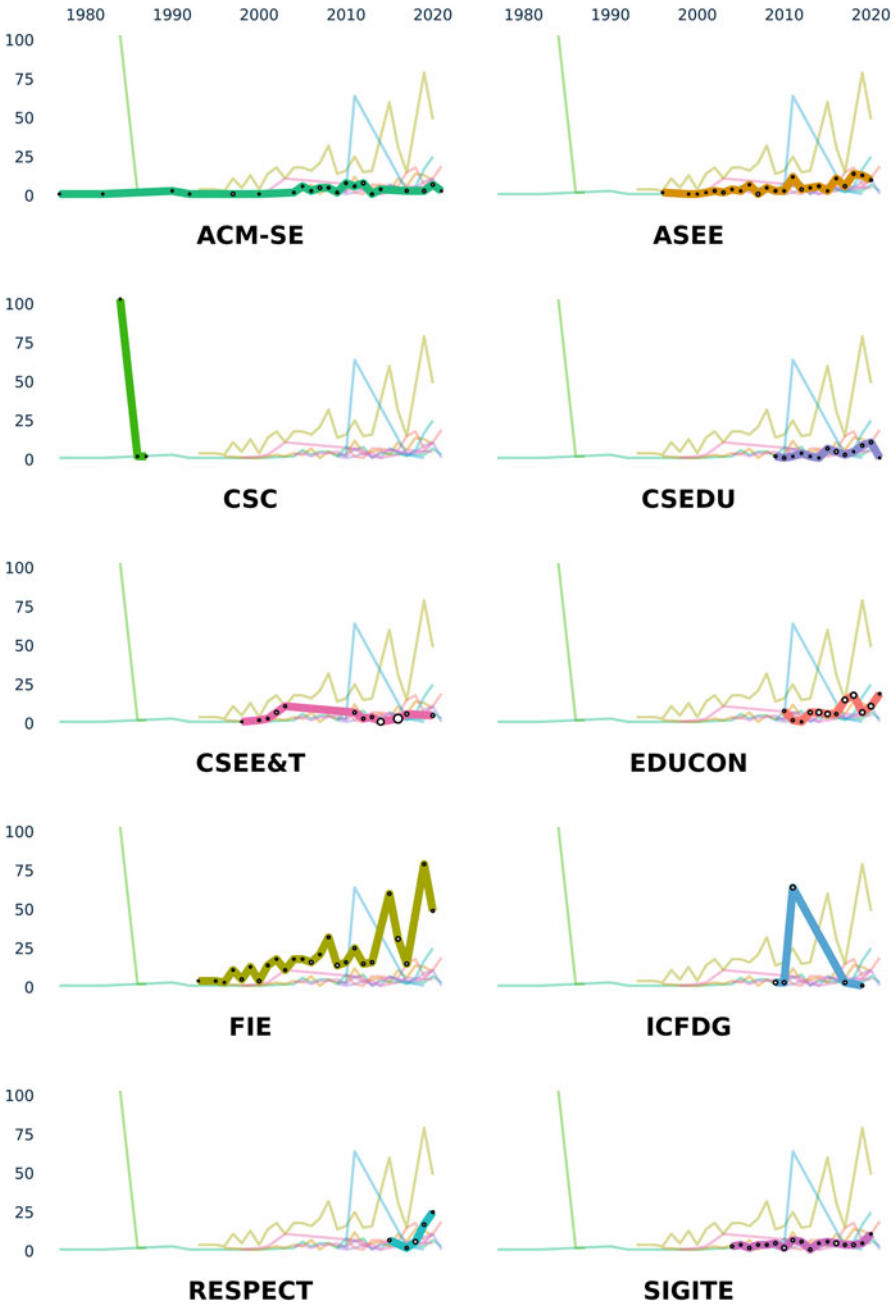


**Fig. 6** Non-dedicated conferences (Top 10). The y-axis represents the number of publications, and white bubbles represent numbers of citations. Each venue has been assigned with a unique color

by the IEEE Education Society [16, 17], soon joined by the American Society for Engineering Education (ASEE) as a co-sponsor [35]. FIE is known to publish contributions both in EER and CER [2]. Other venues in the top 10 include those in general computer science: ASEE Annual Conference and Exposition, ACM Annual Conference on Computer Science (CSC), ACM Annual South-East Conference (ACM-SE), those on EER (EDUCON), information technology education (SIGITE), digital games (ICFDG), equity in computing and engineering (RESPECT), educational technology (CSEDU), and software engineering (International Conference on Software Engineering Education and Training, CSEE & T).

### *4.2 Journals*

The category of non-dedicated journals includes some 572 distinct publication venues, which have together published some 1407 CER articles, accounting for a total share of 8.4% of CER in our dataset. The top 10 publication outlets, which account for 29% of publications in this category (Fig. 7) include venues in EER: IEEE Transactions on Education, Computer Applications in Engineering Education, International Journal of Engineering Education; those in general computer science and computing: Communications of the ACM, Computer, IEEE Access; those in educational technology: Education and Information Technologies, Computers & Education, Journal of Educational Computing Research, and those that are split over CER and educational technology: Informatics in Education.

### *4.3 Keywords and Countries*

The coverage of the top 20 keywords within the CER published in the top 20 non-dedicated journals and conferences shows several things. First, the profile is more divided and diverse. While there are some clear identifiable trendy topics, such as programming, introductory courses and K-12 computing education being popular in many venues, there is also diversity in top topical areas, such as gender and diversity in RESPECT, curriculum in Communications of ACM, educational technology in IEEE Access, and game-based learning in ICFDG. Second, highly popular topics of research, such as CS1/CS2 and programming education, are clearly visible, but otherwise the focus areas in top venues are diverse. This is aligned with the previous findings about the similarities and differences in top keywords; CER published in the non-dedicated venues seem to have a more broad focus on a variety of topics, while the dedicated venues focus more tightly on traditional topics (Fig. 8).

With regards to top contributing countries of the top non-dedicated publication venues, the single largest contributing country has been the USA, including: Communications of the ACM, Frontiers in Education, ASEE Annual Conference and Exposition, RESPECT, Annual ACM Southeast Conference, Conference on Digital

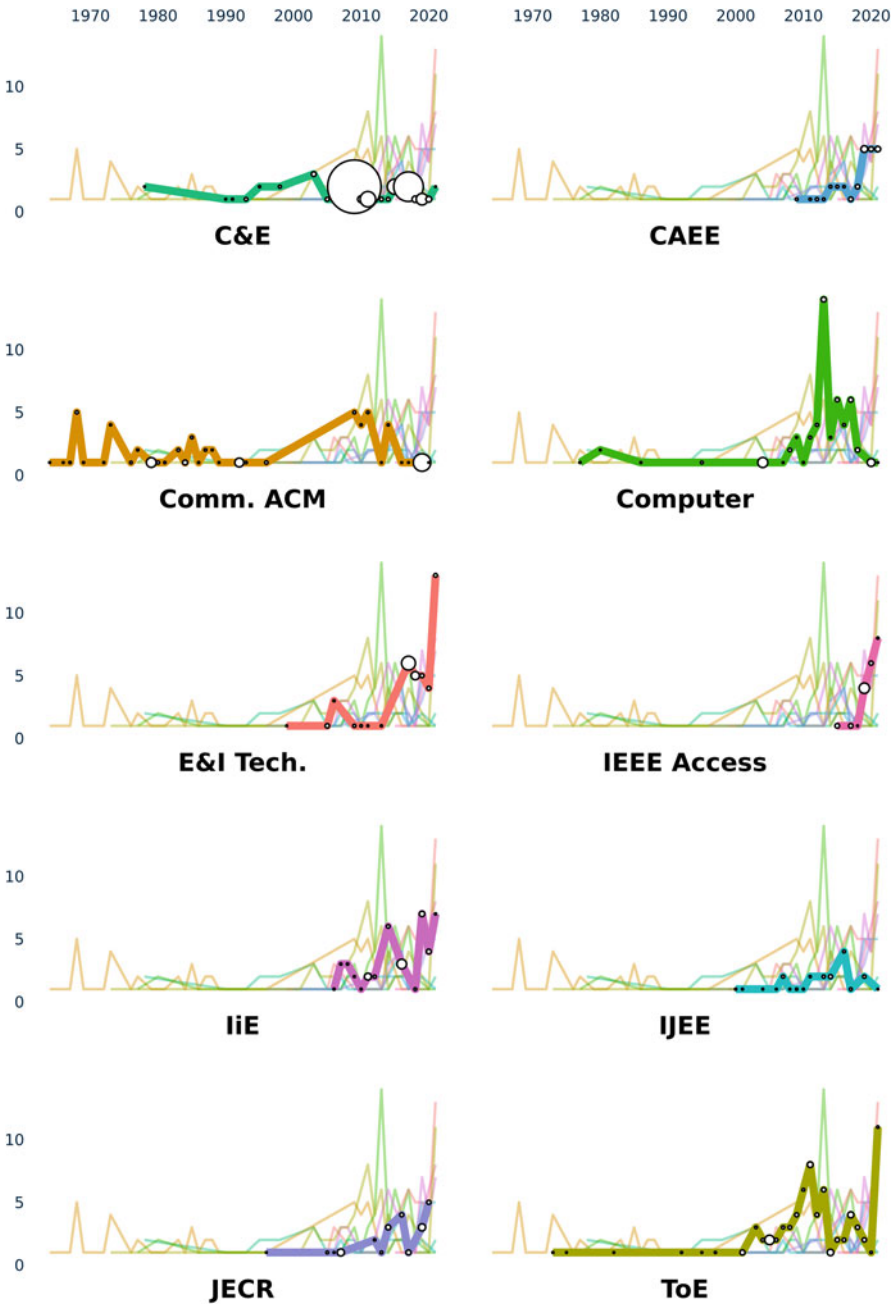


**Fig. 7** Non-dedicated journals (top ten), which account for 29% of articles in the category. The y-axis represents the number of publications, and white bubbles represent numbers of citations. Each venue has been assigned with a unique color

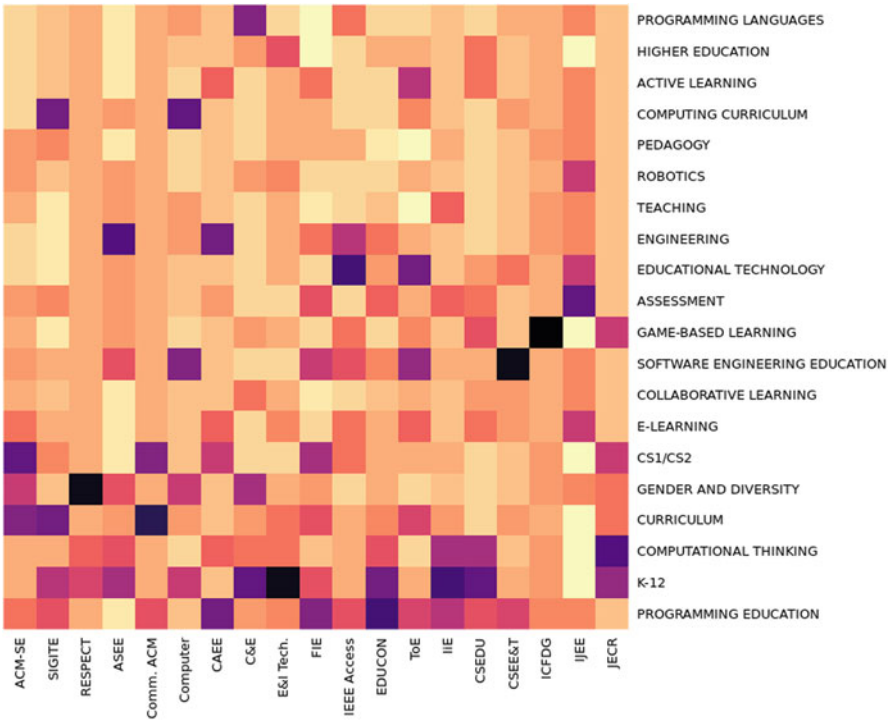


**Fig. 8** Heatmap of Top Non-Dedicated Venues (Top 9 Conferences, Top 10 Journals) and Top 20 Keywords. Darker color indicates stronger keyword presence. Note that ACM Annual Conference on Computer Science, (CSC) was terminated in 1984. Since keywords were not available during that time, CSC is excluded from this figure

Games, SIGITE, EDUCON, CSC, CSEE & T, IEEE Transactions on Education, Computers & Education, and Journal of Educational Computing Research. The exceptions to the USA-dominance are the following publication outlets: CSEDU with its largest share of authors from Brazil, Informatics in Education (Lithuania), International Journal of Engineering Education (Spain), IEEE Access (Spain), E&I Tech (Switzerland), and Computer Applications in Engineering Education (India).

## 5 Discussions

In this chapter we have explored the dissemination practices of CER by zooming in on a total of 1523 publication venues in four categories of dedicated and non-dedicated journals and conferences. We have investigated the volume of publications and citation metrics in four categories, as well as the diversity and differences in the

key topics of research both within and between the top publication venues in each of the four categories. In the following, we will summarise the findings.

## 5.1 Dissemination Practices of CER

The first research question of this chapter asked: *"In which publication outlets does CER get published?"* Our analyses reveal the following.

First, publishing in CER happens mostly in a tight core of 13 highly dedicated conferences and magazines, followed by a set of over 900 conferences scattered in neighboring fields of education, EER, computer science, and others. The third most common category is that of non-dedicated journals, including more than 500 journals in neighboring and related areas of research. The smallest category of CER with regards to number of publications includes the highly dedicated journals: two premier journals CSE and TOCE, and the former JERIC journal, which was terminated in 2009. While the top dedicated journals have published some 4.6% of CER in our dataset, the top dedicated conferences and magazines have published some 66% of CER in our dataset, with the rest of the articles scattered around hundreds of other venues, typically in EER, general computer science, educational technology, and the learning sciences, with some 21% in non-dedicated conferences, and 8.4% in non-dedicated journals. The conference-oriented tendency of dissemination in CER has been inherited from computer science, which also highly regards conferences as publication outlets. The tendency to publish in conferences is in direct contrast with many other disciplines, such as the closely connected discipline of EER. Researchers in EER publish primarily in journals [21]. The first periodical of EER: the *Bulletin of the Society for the Promotion of Engineering Education*, was launched in 1910 and published pedagogical practices and discussions [21]. The research cultures of EER differ also with regards to researcher training: while there are many departments for EER, offering doctoral programs, CER is often conducted within departments of computer science, human-computer interaction, or the learning sciences [21]. There are many similarities between CER and EER: both borrow theoretical and conceptual frameworks from the learning sciences and educational psychology; while EER studies engineering thinking, CER studies computational thinking; and both EER and CER devote significant efforts to first year undergraduate courses [21]. While CER research tends to get published significantly more in conferences than in journals, journal articles receive significantly better citations with journal articles in the highly dedicated journals receiving the most citations, followed by journal articles in non-dedicated venues, conference articles in dedicated venues and finally conference articles in non-dedicated venues, which receive the least of citations.

Second, previous meta-research and reviews [5, 44], narratives and interviews from leading journal editors [13, 55] highlight the process of CER maturing from experience reports and anecdotal evidence towards a rigorous field of research. Previous analyses of research rigor started from the early categorisations of Valen-

tine [56], with the categories *Marco Polo, Tools, Experimental, Nifty, Philosophy and John Henry*, and soon sparked a new track of meta-studies [44] that categorizes CER papers with regards to their research rigor and approach. The first decades of CER were mostly about sharing best practice through experience reports, followed by gradual increase of empirical research, integration with learning sciences, and especially after the 2000s, increased demands from dissemination outlets for research rigor. The new demands are seen in establishment of new venues, such as ICER in 2005, aiming to limit out experience reports. Demands for more rigorous research from top publication outlets have had a significant impact to CER: the share of research papers in ITiCSE has increased from less than 20% before 2000 to over 80% in 2019 [49], and similarly in ACE from less than 20% in 1996 to over 80% in 2020 [48]. While the premier journals CSE and TOCE nowadays desk reject experience reports, they still receive such papers in significant quantities [13]. Many papers are still written by CS educators, who may lack the expertise on the required educational inquiry [55]. Editors wish for a wider repertoire of methods and openings [13], and this is likely a direction where CER will evolve further. While the analysis of many dedicated venues of CER clearly show the trend of increasing research rigor, no previous analyses have investigated research rigor published outside the realm of the dedicated outlets. While it can be estimated that especially many popular journals where CER is typically published, such as *Computers & Education*, have strict requirements for research rigor, there may be many conferences without such requirements. While evaluating research rigor is beyond the scientometric method, this will remain an open research question for future studies. While research rigor is increasing, publication outlets of CER still receive primarily submissions, which introduce a technological intervention or innovation with the assumption that: *"This is a cool idea, surely the contribution is obvious."*, without any proper evaluation or evidence about the assumed positive impact of such technologies [13]. This may resemble overoptimistic beliefs among some that new technologies would be entirely beneficial [50, pp. 222].

## 5.2 Diversity in Dissemination

The second research question of this chapter asked: *"How do the themes of research differ between dedicated and non-dedicated publication outlets of CER?"* The results show the following.

First, we investigated the research topics and themes in dedicated and non-dedicated publication outlets. While most of the previous meta-studies and reviews [5, 44] have investigated topics and themes in publications in the dedicated CER venues, our analysis in this chapter is broader. Previous research shows that many of the known venues of CER are dominated by research in introductory computing, typically courses on computer programming [2, 5, 13, 44, 48, 49, 56], calling for more diversity in topics. An interview with the leading journal editors of the core dedicated CER journals also noted how almost all papers they receive

for review are about CS1 (introductory computer science). The editor of *Computer Science Education*, in an interview held in August 2017, noted: *"I think we have had only a couple of papers in the last year for CSE that were both undergraduate and outside of CS1"* [13]. The wish of editors has been that of more diversity in research topics, and methods, with e.g. wishes for use of deep ethnographies [13].

Our investigation shows differences in the top topics of research (keywords) between the dedicated venues (journals and conferences combined) and the non-dedicated venues (journals and conferences combined). While one of the trends of CER has been its integration with neighboring disciplines of the learning sciences, EER, general computer science, and others, it is interesting to observe the relationship between this trend and the selection of publication venues, and topics in published research. When looking at the top keywords, our analysis shows that e.g. research on introductory courses (CS1/CS2), object-oriented programming, novice programmers, Java, and automatic assessment all are significantly higher in the dedicated venue's category, while research e.g. on game-based learning, informatics education, educational technology, STEM, and information technology are significantly higher in the non-dedicated venues. These results show that the core CER venues center much on common core topics, but non-dedicated venues lean more towards applied topics, such as educational technology and games.

We investigated the diversity of research topics within the dedicated venues and the non-dedicated venues, separately. With regards to dedicated venues, previous research shows the dominance of teaching programming as a topic of CER [2, 13, 44, 48, 49, 56], and highly popular and emerging topics such as K-12 and computational thinking [5]. When analysing the top dedicated venues separately, first courses and programming were found to be strong in all of them. Differences were observed e.g. in the orientation towards *K-12 education* in WIPSCE, ISSEP, and ACM TOCE, *visualization* in Koli Calling, *assessment* in ACE and ACM JERIC, *gender and diversity* in ACM TOCE, and *curriculum* in SIGCSE Bulletin. Indeed, while core dedicated publication venues share many similarities in their top topics, they have their own specialisation areas, too. With regards to geography in the top dedicated venues, analysis of author affiliations in the top dissemination venues of CER show that in many venues (ICER, ITiCSE, SIGCSE Technical Symposium, TOCE, JERIC, SIGCSE Bulletin, CSE), North America is the single largest contributing country, while Koli Calling is dominated by authors from Finland; ISSEP has many contributions from Germany, Austria, Switzerland, and Italy; the largest country of WIPSCE is Germany; and the largest shares of authors of ACE are affiliated with institutions in Australia and New Zealand. While it is well-known that by far the largest share of CER has originated from North America [5], there are clear differences in the top contributing countries per top venues, too.

When it comes to the non-dedicated venues, top topics such as research in introductory courses or K-12 computing education is clearly visible, but not to the same extent as in the dedicated venues. Instead, there is a more diverse range of top topics. Within the non-dedicated venues, USA is also the single largest contributing country in most of the venues, with a few exceptions; CSEDU has its largest share of authors from Brazil, Informatics in Education from Lithuania, and International

Journal of Engineering Education, as well as IEEE Access, from Spain, E&I from Switzerland, and Computer Applications in Engineering Education has its largest share of articles originating from India.

### *5.3 Limitations*

The data is not without limitations. While Elsevier's Scopus is a well maintained database and more accurate than Web of Science [14], it is far from perfect. Some of the known problems with Scopus metadata include, but are not limited to the following. There are numerous issues with missing fields, inconsistent recording of keywords, references, mistakes in publication venues, missing data and even missing years in some venues. For example, titles of proceedings are often recorded inconsistently, which can not always be detected even with exhaustive cleaning operations (how to classify a publication venue, whose only title metadata is "IEEE"). Sometimes publications are misclassified as being published in completely different venues, misclassified as articles while indeed they are posters or editorials or vice versa, while some articles might be completely missing from the data, or their references or citation counts may be incorrectly recorded. Even with many manual and algorithmic methods for fixing inconsistencies and errors, correcting all mistakes is simply impossible. The limitations in the data are fully explained in Chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" of this book [20].

## 6 Conclusions

Each field of science have their own unique cultures of dissemination. The dissemination outlets determine what research is valued and what is not, while the evaluation criteria evolves in time. The hyper competitive nature of academic life is sometimes described by the aphorism "publish or perish." While publication metrics play an increasing role in the university life, it is important, every now and then, to turn a critical eye to the practices of dissemination. Measuring the quality of research is almost as difficult as objectively measuring the quality of songs, movies, cities, or loved ones. A fundamental problem with citation-based rankings is that they measure what other researchers at a given time are interested at [37, pp. 26–27]. Also, computing is partly a mathematical discipline, partly a scientific discipline, and partly an engineering discipline concerned with design and construction [52, 58]. Different venues and interest groups may also have their own emphases inside the branches of computing, which may also have an impact on how they conceive and conduct CER.

While previous research focusing on dissemination in CER has zoomed mostly into the top dedicated publication outlets in CER, in this research we extended

the scope of analysis to a number of other publication outlets, too. Our analysis revealed specific characteristics of CER: conference-orientation inherited from CS, high and low cited categories of publication venues, and varying diversity in topics of research. It seems that the conference orientation and low citations among conferences might pose a danger to career advancement in CER. While some conferences, such as ICER, are advancing in terms of their research rigor, the merit of conference publication may still be considerably low outside the CER and computing communities. While CER has currently only two active dedicated journals, this issue warrants for reflection among the CER community. More broadly, it becomes relevant to ask: how can the dissemination outlets of CER better serve career advancement of CER researchers? However, this does not mean that CER should necessarily become more journal-oriented, but parhaps that the conference-based tendency of CER and computing should be better accepted, also among other disciplines.

## Appendix: Publication Venues in Four Categories

**DEDICATED CONFERENCES AND MAGAZINES** ($N = 13$)

1. ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION, SIGCSE
2. ACM SIGCSE BULLETIN
3. INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION, ITiCSE
4. INTERNATIONAL COMPUTING EDUCATION RESEARCH CONFERENCE, ICER
5. KOLI CALLING INTERNATIONAL CONFERENCE ON COMPUTING EDUCATION RESEARCH
6. INTERNATIONAL CONFERENCE ON INFORMATICS IN SCHOOLS, ISSEP
7. AUSTRALASIAN COMPUTING EDUCATION CONFERENCE, ACE
8. WORKSHOP IN PRIMARY AND SECONDARY COMPUTING EDUCATION, WIPSCE
9. ACM INROADS
10. COMPUTER SCIENCE EDUCATION RESEARCH CONFERENCE, CSERC
11. CONFERENCE ON COMPUTING EDUCATION PRACTICE, CEP
12. ACM GLOBAL COMPUTING EDUCATION CONFERENCE, COMPED
13. UK AND IRELAND COMPUTING EDUCATION RESEARCH CONFERENCE, UKICER

**DEDICATED JOURNALS (3)**

1. COMPUTER SCIENCE EDUCATION
2. ACM TRANSACTIONS ON COMPUTING EDUCATION
3. ACM JOURNAL ON EDUCATIONAL RESOURCES IN COMPUTING

**NON-DEDICATED CONFERENCES WITH 5 OR MORE ARTICLES**

1. FRONTIERS IN EDUCATION CONFERENCE, FIE
2. ASEE ANNUAL CONFERENCE AND EXPOSITION
3. ACM ANNUAL CONFERENCE ON COMPUTER SCIENCE, CSC
4. IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE, EDUCON
5. ANNUAL CONFERENCE ON INFORMATION TECHNOLOGY EDUCATION, SIGITE
6. ANNUAL ACM SOUTHEAST CONFERENCE, ACM-SE
7. INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, ICFDG
8. ANNUAL CONFERENCE RESEARCH ON EQUITY AND SUSTAINED PARTICIPATION IN ENGINEERING, COMPUTING, AND TECHNOLOGY, RESPECT
9. INTERNATIONAL CONFERENCE ON COMPUTER SUPPORTED EDUCATION, CSEDU
10. INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING EDUCATION AND TRAINING, CSEE AND T
11. WESTERN CANADIAN CONFERENCE ON COMPUTING EDUCATION, WCCCE
12. INTERNATIONAL CONFERENCE ON LEARNING AND TEACHING IN COMPUTING AND ENGINEERING, LATICE
13. IEEE INTERNATIONAL CONFERENCE ON ADVANCED LEARNING TECHNOLOGIES, ICALT
14. IEEE SYMPOSIUM ON VISUAL LANGUAGES AND HUMAN-CENTRIC COMPUTING, VL/HCC
15. IEEE INTERNATIONAL CONFERENCE ON TEACHING, ASSESSMENT AND LEARNING FOR ENGINEERING, TALE
16. CONFERENCE ON TRI-ADA, TRI-ADA
17. INTERNATIONAL CONFERENCE OF THE LEARNING SCIENCES, ICLS
18. IEEE INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND EDUCATION, ICCSE
19. IEEE INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM, IPDPS
20. IFIP WORLD CONFERENCE ON COMPUTERS IN EDUCATION, WCCE
21. ACM SYMPOSIUM ON SOFTWARE VISUALIZATION, SOFTVIS
22. ANNUAL ACM CONFERENCE ON LEARNING AT SCALE, L@S
23. IEEE INTEGRATED STEM EDUCATION CONFERENCE, ISEC
24. INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND COMPUTATIONAL INTELLIGENCE, CSCI
25. INTERNATIONAL CONFERENCE ON INTERACTIVE COLLABORATIVE LEARNING, ICL
26. ENGINEERING AS A HUMAN ENDAEVOR: PARTNERING COMMUNITY, ACADEMIA, GOVERNMENT, AND INDUSTRY
27. INTERNATIONAL CONVENTION ON INFORMATION AND COMMUNICATION TECHNOLOGY, ELECTRONICS AND MICROELECTRONICS, MIPRO
28. INTELLIGENT NARRATIVE TECHNOLOGIES III WORKSHOP, INT3
29. ACM INTERACTION DESIGN AND CHILDREN CONFERENCE, IDC
30. ACM/IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING EDUCATION AND TRAINING, ICSE-SEET
31. LECTURE NOTES IN COMPUTER SCIENCE
32. INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE IN EDUCATION, AIED
33. INTERNATIONAL CONFERENCE ON COMPUTERS IN EDUCATION, ICCE
34. AAAI SPRING SYMPOSIUM
35. ACM ANNUAL CONFERENCE
36. EUROPEAN CONFERENCE ON GAME BASED LEARNING, ECGBL
37. IEEE ANNUAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, COMPSAC
38. INTERNATIONAL COMPUTER PROGRAMMING EDUCATION CONFERENCE, ICPEC
39. INTERNATIONAL CONFERENCE ON COMPUTATIONAL THINKING EDUCATION, CTE

40. INTERNATIONAL CONFERENCE ON COMPUTER-SUPPORTED COLLABORATIVE LEARNING, CSCL
41. INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY BASED HIGHER EDUCATION AND TRAINING, ITHET
42. INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY- NEW GENERATIONS, ITNG
43. INTERNATIONAL OLYMPIAD IN INFORMATICS, IOI
44. INFORMATION SYSTEMS EDUCATION CONFERENCE, ISECON
45. INTERNATIONAL CONFERENCE JOINT WITH THE INTERNATIONAL OLYMPIAD IN INFORMATICS, IOI
46. INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ICSE
47. INTERNATIONAL MULTI-CONFERENCE ON SOCIETY, CYBERNETICS AND INFORMATICS, IMSCI
48. AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, AAAI
49. ANNUAL ACM SIGPLAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, SYSTEMS, LANGUAGES, AND APPLICATIONS, OOPSLA
50. ANNUAL CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI
51. IASTED INTERNATIONAL CONFERENCE ON COMPUTERS AND ADVANCED TECHNOLOGY IN EDUCATION, CATE
52. IEEE BLOCKS AND BEYOND WORKSHOP, B AND B
53. IEEE WORLD ENGINEERING EDUCATION CONFERENCE, EDUNINE
54. INTERNATIONAL CONFERENCE ON TECHNOLOGY FOR EDUCATION, T4E
55. WORKSHOP ON PROCEDURAL CONTENT GENERATION IN GAMES, PC GAMES, CO-LOCATED WITH THE FOUNDATIONS OF DIGITAL GAMES CONFERENCE
56. INTERNATIONAL CONFERENCE ON EDUCATIONAL DATA MINING, EDM
57. INTERNATIONAL CONFERENCE ON INTERACTION DESIGN AND CHILDREN, IDC
58. IFIP ADVANCES IN INFORMATION AND COMMUNICATION TECHNOLOGY
59. ALICE SYMPOSIUM, ALICE
60. AMERICAS CONFERENCE ON INFORMATION SYSTEMS, AMCIS
61. EUROPEAN CONFERENCE ON TECHNOLOGY ENHANCED LEARNING, EC-TEL
62. INFORMATION SYSTEMS EDUCATORS CONFERENCE, ISECON AND CONFERENCE ON INFORMATION SYSTEMS APPLIED RESEARCH, CONISAR
63. LATIN AMERICAN COMPUTING CONFERENCE, CLEI
64. IEEE SOUTHEASTCON, SOUTHEASTCON
65. IFIP INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING
66. ACM CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI
67. ANNUAL CONFERENCE OF THE SOUTHERN AFRICAN COMPUTER LECTURERS ASSOCIATION, SACLA
68. ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, HICSS
69. INTERNATIONAL CONFERENCE ON INTELLIGENT TUTORING SYSTEMS, ITS
70. INTERNATIONAL SYMPOSIUM ON COMPUTERS IN EDUCATION, SIIE
71. SEI CONFERENCE ON SOFTWARE ENGINEERING EDUCATION,
72. ACM INTERNATIONAL CONFERENCE ON USER MODELING, ADAPTATION AND PERSONALIZATION, UMAP
73. ACM SIGPLAN SYMPOSIUM ON SPLASH-E, SPLASH-E, CO-LOCATED WITH SPLASH
74. ANNUAL ACM SIGADA INTERNATIONAL CONFERENCE ON ADA, SIGADA
75. BRAZILIAN SYMPOSIUM ON SOFTWARE ENGINEERING, SBES
76. CONFERENCE ON SOFTWARE ENGINEERING EDUCATION, CSEE
77. IEEE INTERNATIONAL CONFERENCE ON EMERGING ELEARNING TECHNOLOGIES AND APPLICATIONS, ICETA
78. INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE, ICCS
79. INTERNATIONAL CONFERENCE ON EDUCATIONAL AND INFORMATION TECHNOLOGY, ICEIT
80. INTERNATIONAL CONFERENCE ON GAME DEVELOPMENT IN COMPUTER SCIENCE EDUCATION, GDCSE
81. LECTURE NOTES IN EDUCATIONAL TECHNOLOGY
82. ACM WORKSHOP ON SECURITY AND PRIVACY ON ARTIFICIAL INTELLIGENT, SPAI, CO-LOCATED WITH ASIACCS
83. AUSTRALASIAN COMPUTER SCIENCE WEEK MULTICONFERENCE, ACSW
84. IEEE INTERNATIONAL CONFERENCE ON ELECTRO/INFORMATION TECHNOLOGY, EIT
85. IEEE INTERNATIONAL CONFERENCE ON ENGINEERING, TECHNOLOGY AND EDUCATION, TALE
86. IEEE/ACM WORKSHOP ON EDUCATION FOR HIGH-PERFORMANCE COMPUTING, EDUHPC
87. IFIP TC3 OPEN CONFERENCE ON COMPUTERS IN EDUCATION, OCCE
88. IIAI INTERNATIONAL CONGRESS ON ADVANCED APPLIED INFORMATICS, IIAI-AAI
89. INTERNATIONAL ACM SIGACCESS CONFERENCE ON COMPUTERS AND ACCESSIBILITY, ASSETS
90. INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND TECHNOLOGIES, COMPSYSTECH
91. INTERNATIONAL CONFERENCE ON E-LEARNING, ICEL
92. INTERNATIONAL CONFERENCE ON LEARNING ANALYTICS AND KNOWLEDGE, LAK
93. INTERNATIONAL CONFERENCE ON ROBOTICS IN EDUCATION, RIE
94. WORKSHOP ON EDUCATION FOR HIGH-PERFORMANCE COMPUTING, EDUHPC
95. WORLD CONGRESS ON ENGINEERING, WCE
96. WORLD MULTI-CONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS, WMSCI
97. ACM SIGPLAN NOTICES
98. INTERNATIONAL CONVENTION : COMPUTERS IN EDUCATION
99. LECTURE NOTES IN NETWORKS AND SYSTEMS
100. ACM INTERNATIONAL CONFERENCE COMPANION ON OBJECT ORIENTED PROGRAMMING SYSTEMS LANGUAGES AND APPLICATIONS COMPANION, SPLASH
101. CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, CHI
102. COMPUTING CONFERENCE
103. EUROPEAN CONFERENCE ON SOFTWARE ARCHITECTURE, ECSA
104. INFORMATION SECURITY CURRICULUM DEVELOPMENT CONFERENCE, INFOSECCD
105. INTERNATIONAL CONFERENCE FOR YOUNG COMPUTER SCIENTISTS, ICYCS
106. INTERNATIONAL CONFERENCE IN METHODOLOGIES AND INTELLIGENT SYSTEMS FOR TECHNOLOGY ENHANCED LEARNING, MIS4TEL
107. INTERNATIONAL CONFERENCE ON INFORMATION AND EDUCATION TECHNOLOGY, ICIET
108. INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY: RESEARCH AND EDUCATION, ITRE
109. INTERNATIONAL CONFERENCE ON SOCIETY AND INFORMATION TECHNOLOGIES, ICSIT
110. INTERNATIONAL FLORIDA ARTIFICIAL INTELLIGENCE RESEARCH SOCIETY CONFERENCE, FLAIRS
111. IST-AFRICA CONFERENCE, IST-AFRICA
112. PANHELLENIC CONFERENCE ON INFORMATICS, PCI
113. WORKSHOP ON EMBEDDED AND CYBER-PHYSICAL SYSTEMS EDUCATION, WESE

**NON-DEDICATED JOURNALS WITH 5 OR MORE ARTICLES**

1. IEEE TRANSACTIONS ON EDUCATION
2. COMMUNICATIONS OF THE ACM
3. COMPUTER
4. EDUCATION AND INFORMATION TECHNOLOGIES
5. INFORMATICS IN EDUCATION
6. COMPUTERS AND EDUCATION
7. COMPUTER APPLICATIONS IN ENGINEERING EDUCATION
8. INTERNATIONAL JOURNAL OF ENGINEERING EDUCATION
9. JOURNAL OF EDUCATIONAL COMPUTING RESEARCH
10. IEEE ACCESS
11. REVISTA IBEROAMERICANA DE TECNOLOGIAS DEL APRENDIZAJE
12. INTERNATIONAL JOURNAL OF PHYTOREMEDIATION
13. EDUCATION AND COMPUTING
14. IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES
15. JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING
16. INTERACTIVE LEARNING ENVIRONMENTS
17. INTERNATIONAL JOURNAL OF EMERGING TECHNOLOGIES IN LEARNING
18. COMPUTING IN SCIENCE AND ENGINEERING
19. IEEE COMPUTER GRAPHICS AND APPLICATIONS
20. IEEE SECURITY AND PRIVACY
21. IEEE DISTRIBUTED SYSTEMS ONLINE
22. EDUCATION SCIENCES
23. COMPUTERS IN EDUCATION JOURNAL
24. INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE IN EDUCATION
25. ACM SIGGRAPH EDUCATORS PROGRAM, SIGGRAPH
26. IEEE SOFTWARE
27. TECHTRENDS
28. INTERNATIONAL JOURNAL OF CHILD-COMPUTER INTERACTION
29. COMPUTERS IN HUMAN BEHAVIOR
30. EDUCATIONAL TECHNOLOGY AND SOCIETY
31. ELECTRONIC NOTES IN THEORETICAL COMPUTER SCIENCE
32. JOURNAL OF UNIVERSAL COMPUTER SCIENCE
33. EDUCATIONAL TECHNOLOGY RESEARCH AND DEVELOPMENT
34. INFORMATIK-SPEKTRUM
35. MICROPROCESSING AND MICROPROGRAMMING
36. BRITISH JOURNAL OF EDUCATIONAL TECHNOLOGY

37. INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY EDUCATION
38. JOURNAL OF RESEARCH ON COMPUTING IN EDUCATION
39. JOURNAL OF SCIENCE EDUCATION AND TECHNOLOGY
40. JOURNAL OF THEORETICAL AND APPLIED INFORMATION TECHNOLOGY
41. ACTA POLYTECHNICA HUNGARICA
42. ANNALS OF THE HISTORY OF COMPUTING
43. COMPUTERS IN THE SCHOOLS
44. IEEE PERVASIVE COMPUTING
45. SUSTAINABILITY
46. AEDS JOURNAL
47. JOURNAL OF COMPUTING IN HIGHER EDUCATION
48. JOURNAL OF INFORMATION SYSTEMS EDUCATION
49. JOURNAL OF RESEARCH ON TECHNOLOGY IN EDUCATION

## References


1. Aiken, R.M.: Editorial notes and observations. SICCSE Bulletin **1**(4), 2 (1969)
2. Apiola, M., Tedre, M., López-Pernas, S., Saqr, M., Daniels, M., Pears, A.: A scientometric journey through the FIE bookshelf: 1982–2020. In: 2021 IEEE Frontiers in Education Conference (FIE), pp. 1–9 (2021). DOI https://doi.org/.1109/FIE49875.2021.9637209
3. Apiola, M., Saqr, M., López-Pernas, S.: The Hands that Made Computing Education Research: Top Authors, Networks, Collaboration and Newcomers. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
4. Apiola, M., López-Pernas, S., Saqr, M., Pears, A., Daniels, M., Malmi, L., Tedre, M.: From a National Meeting to an International Conference: A Scientometric Case Study of a Finnish Computing Education Conference. IEEE Access **10**, 66576–66588 (2022). DOI https://doi.org/10.1109/ACCESS.2022.3184718
5. Apiola, M., Saqr, M., López-Pernas, S., Tedre, M.: Computing education research compiled: Keyword trends, building blocks, creators, and dissemination. IEEE Access **10**, 27041–27068 (2022). DOI https://doi.org/10.1109/ACCESS.2022.3157609
6. Becker, B.A., Quille, K.: 50 years of cs1 at sigcse: A review of the evolution of introductory programming education research. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, pp. 338–344. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3287324.3287432
7. Becker, B.A., Settle, A., Luxton-Reilly, A., Morrison, B.B., Laxer, C.: Expanding opportunities: Assessing and addressing geographic diversity at the sigcse technical symposium. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, pp. 281–287. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3408877.3432448
8. Berglund, A., Daniels, M., Pears, A.: Qualitative research projects in computing education research: An overview. In: Proceedings of the 8th Australasian Conference on Computing Education-Volume 52, pp. 25–33 (2006)
9. Berglund, A., Box, I., Eckerdal, A., Lister, R., Pears, A.: Learning educational research methods through collaborative research: the phicer initiative. In: Proceedings of the tenth conference on Australasian computing education-Volume 78, pp. 35–42 (2008)
10. Cassel, L.N., Fox, E.A.: Editorial: Introducing the acm journal on resources in computing. J. Educ. Resour. Comput. **1**(1es), 1–es (2001). https://doi.org/10.1145/376697.382399
11. Fincher, S., Petre, M.: Computer Science Education Research. Taylor & Francis (2004)
12. Fincher, S., Tenenberg, J.: Using theory to inform capacity-building: Bootstrapping communities of practice in computer science education research. Journal of Engineering Education **95**(4), 265–277 (2006). URL https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2168-9830.2006.tb00902.x
13. Fincher, S.A., Dorn, B., Hundhausen, C., McCartney, R., Murphy, L.: Computing Education Research Today. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, pp. 40–55. Cambridge University Press (2019). DOI https://doi.org/10.1017/9781108654555.001
14. Franceschini, F., Maisano, D., Mastrogiacomo, L.: Empirical analysis and classification of database errors in Scopus and Web of Science. Journal of Informetrics **10**(4), 933–953 (2016). DOI https://doi.org/10.1016/j.joi.2016.07.003.

15. Guzdial, M., du Boulay, B.: The history of computing education research. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, pp. 11–39. Cambridge University Press, Cambridge (2019). DOI https://doi.org/10.1017/9781108654555.002.
16. Jones, E.C.: A journey through the FIE bookshelf-1971-2000. In: 30th Annual Frontiers in Education Conference. Building on A Century of Progress in Engineering Education. Conference Proceedings, vol. 1, pp. T4G/1–T4G/6 vol.1 (2000). DOI https://doi.org/10.1109/FIE.2000.897669
17. Jones, E.C., Rowland, J.R.: Frontiers in Education—have we made a difference? If so, what? In: 2016 IEEE Frontiers in Education Conference (FIE), pp. 1–4 (2016). DOI https://doi.org/10.1109/FIE.2016.7757553
18. Lishinski, A., Good, J., Sands, P., Yadav, A.: Methodological Rigor and Theoretical Foundations of CS Education Research. In: Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16, pp. 161–169. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2960310.2960328
19. López-Pernas, S., Apiola, M., Saqr, M., Pears, A., Tedre, M.: A Scientometric Perspective on the Evolution of the SIGCSE Technical Symposium: 1970–2021. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research: A Global Perspective, pp. in–press. Springer (2023)
20. López-Pernas, S., Saqr, M., Apiola, M.: Scientometrics: A Concise Introduction and a Detailed Methodology for the Mapping of the Scientific Field of Computing Education Research. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
21. Loui, M.C., Borrego, M.: Engineering Education Research. In: S.A. Fincher, A. Sally, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, pp. 292–321. Cambridge University Press, Cambridge (2019). https://doi.org/10.1017/9781108654555.002
22. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Characterizing research in computing education: A preliminary analysis of the literature. In: Proceedings of the Sixth International Workshop on Computing Education Research, ICER '10, pp. 3–12. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1839594.1839597
23. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Kinnunen, P., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Theoretical underpinnings of computing education research: What is the evidence? In: Proceedings of the Tenth Annual Conference on International Computing Education Research, ICER '14, pp. 27–34. Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2632320.2632358
24. Malmi, L., Sheard, J., Kinnunen, P., Simon, Sinclair, J.: Computing education theories: What are they and how are they used? In: Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER '19, pp. 187–197. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3291279.3339409
25. Margulieux, L.E., Dorn, B., Searle, K.A.: Learning Sciences for Computing Education. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, chap. 8, pp. 208–230. Cambridge University Press, Cambridge (2019)
26. McCartney, R., Sanders, K.: ITiCSE working groups and collaboration in the computing education community. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2018, pp. 332–337. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3197091.3197143
27. McCartney, R., Sanders, K.: ITiCSE Working Groups As an Engine for Community-Building. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, pp. in–press. Springer (2022)
28. McCartney, R., Sanders, K.: ITiCSE Working Groups As an Engine for Community-Building. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research: A Global Perspective. Springer (2022)

29. Miró Julià, J., López, D., Alberich, R.: Education and research: Evidence of a dual life. In: Proceedings of the Ninth Annual International Conference on International Computing Education Research, ICER '12, pp. 17–22. Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2361276.2361281
30. Pears, A.N., Daniels, M.: Structuring csed research studies: Connecting the pieces. ACM SIGCSE Bulletin **35**(3), 149–153 (2003)
31. Pears, A., Daniels, M., Berglund, A.: Describing computer science education research: an academic process view. Simulation Series **34**(1), 99–104 (2002)
32. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., Paterson, J.: A Survey of Literature on the Teaching of Introductory Programming. SIGCSE Bulletin **39**, 204–223 (2007). http://doi.acm.org/10.1145/1345375.1345441
33. Randolph, J.J.: Computer science education research at the crossroads: A methodological review of the computer science education research: 2000–2005. Ph.D. thesis, Utah State University, Logan, UT, USA (2007)
34. Randolph, J.J., Bednarik, R., Myller, N.: A methodological review of the articles published in the proceedings of Koli Calling 2001–2004. In: Proceedings of the 5th Annual Finnish/ Baltic Sea Conference on Computer Science Education, pp. 103–109. Helsinki University of Technology Press, Finland (2005)
35. Richards, L.G.: Pursuing the frontiers: The history and future of the Frontiers in Education conference. In: 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, pp. 1–4 (2014). DOI https://doi.org/10.1109/FIE.2014.7044187
36. Robins, A., Rountree, J., Rountree, N.: Learning and Teaching Programming: A Review and Discussion. Computer Science Education **13**(2), 137–172 (2003)
37. Saarikivi, T., Saarikivi, J.: Turhan Tiedon Kirja (The Book of Useless Knowledge) . SKS Kirjat (2021)
38. Saqr, M., Ng, K., Oyelere, S.S., Tedre, M.: People, ideas, milestones: A scientometric study of computational thinking. ACM Trans. Comput. Educ. **21**(3) (2021). https://doi.org/10.1145/3445984
39. Settle, A., Becker, B.A., Duran, R., Kumar, V., Luxton-Reilly, A.: Improving Global Participation in the SIGCSE Technical Symposium: Panel. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20, pp. 483–484. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3328778.3366979
40. Simon: A Classification of Recent Australasian Computing Education Publications. Computer Science Education **17**(3), 155–169 (2007). https://doi.org/10.1080/08993400701538021
41. Simon: Koli Calling comes of age: an analysis. In: R. Lister, Simon (eds.) Proceedings of the Seventh Baltic Sea Conference on Computing Education Research (Koli Calling 2007), *CRPIT*, vol. 88, pp. 119–126. Australian Computer Society (2008)
42. Simon: Informatics in Education and Koli Calling: a Comparative Analysis. Informatics in Education **8**(1), 101–114 (2009)
43. Simon: Ten years of the australasian computing education conference. In: Proceedings of the Eleventh Australasian Conference on Computing Education - Volume 95, ACE '09, pp. 157–164. Australian Computer Society, Inc., AUS (2009)
44. Simon: Emergence of computing education as a research discipline. Ph.D. thesis, Aalto University School of Science (2015)
45. Simon: A Picture of the Growing ICER Community. In: Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16, pp. 153–159. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2960310.2960323
46. Simon: The Koli Calling Community. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli Calling '16, pp. 101–109. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2999541.2999562
47. Simon: Twenty-four years of ITiCSE authors. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, pp. 205–211. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3341525.3387387

48. Simon: Twenty-two years of ace. In: Proceedings of the Twenty-Second Australasian Computing Education Conference, ACE'20, pp. 203–210. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3373165.3373188
49. Simon, Sheard, J.: Twenty-Four Years of ITiCSE Papers. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, pp. 5–11. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3341525.3387407
50. Smith, B., Browne, C.A.: Tools and Weapons: The Promise and Peril of the Digital Age. Hodder & Stoughton (2021)
51. Szabo, C., Falkner, N., Petersen, A., Bort, H., Cunningham, K., Donaldson, P., Hellas, A., Robinson, J., Sheard, J.: Review and use of learning theories within computer science education research: Primer for researchers and practitioners. In: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE-WGR '19, pp. 89–109. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3344429.3372504
52. Tedre, M., Apiola, M.: Three Computing Traditions in School Computing Education. In: D.M. Kadijevich, C. Angeli, C. Schulte (eds.) Improving Computer Science Education, pp. 100–116. Routledge (2013)
53. Tedre, M., Simon, Malmi, L.: Changing aims of computing education: a historical survey. Computer Science Education **28**(2), 158–186 (2018). https://doi.org/10.1080/08993408.2018.1486624
54. Tenenberg, J., McCartney, R.: Introducing the ACM Transactions on Computing Education. ACM Trans. Comput. Educ. **9**(1) (2009). https://doi.org/10.1145/1513593.1513594
55. Tenenberg, J., McCartney, R.: Looking backward to look forward: Toce in transition. ACM Trans. Comput. Educ. **15**(3) (2015). https://doi.org/10.1145/2817209
56. Valentine, D.W.: Cs educational research: A meta-analysis of sigcse technical symposium proceedings. SIGCSE Bull. **36**(1), 255–259 (2004). https://doi.org/10.1145/1028174.971391
57. Walker, H.M., Dooley, J.F.: The history of the sigcse submission and review software: From paper to the cloud? In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, pp. 1074–1080. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3287324.3287427
58. Wegner, P.: Three computer cultures: Computer technology, computer mathematics, and computer science. pp. 7–78. Elsevier (1970). DOI https://doi.org/10.1016/S0065-2458(08)60431-3
59. Zhang, J., Luxton-Reilly, A., Denny, P., Whalley, J.: Scientific collaboration network analysis for computing education conferences. In: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE '21, pp. 582–588. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3430665.3456385

# The Evolving Themes of Computing Education Research: Trends, Topic Models, and Emerging Research


**Mikko Apiola, Mohammed Saqr, and Sonsoles López-Pernas**

## 1 Introduction

For building an understanding of any discipline of science, it is crucial to take a look at its research areas. Previous meta-research has covered trends in research topics, nature of publications, use of research methods and development of theoretical frameworks [25]. The findings show how experience reports have evolved into empirical research, a sustained focus on programming education, which is the all-time most popular topic of CER, and the decrease of research on some areas, such as that of which programming language to use [25]. Findings from meta-studies are well aligned with other recent analyses of CER, which have revealed, e.g., influential articles, theoretical backgrounds, institutions and communities, and characteristics of top publication outlets and related citation statistics, and e.g. the large proportion of research originating in high-income countries, especially North America [3]. Out of all the research areas, during all times of CER, teaching and learning of programming has always been the top and dominating area of research, and much continues to be so [3, 25].

Other top areas of research in CER have included those of K-12 computing education, and computational thinking (CT), research on a variety of tools, such as those on visualisation, automatic assessment or learning analytics, and research on software engineering education [3]. In this chapter, we add to the previous analyses of research areas and keyword trends in CER in the following way. First, in Sect. 2, we provide a brief historical look into the evolution of the research areas of CER, and results from an analysis of top keyword trends. Second, we present an analysis based on unsupervised classification of keyword metadata, resulting in a model of 29 topics, introduced in Sect. 3. Third, we present analysis of emerging common

M. Apiola (✉) · M. Saqr · S. López-Pernas
University of Eastern Finland, Joensuu, Finland
e-mail: mikko.apiola@uef.fi; mohammed.saqr@uef.fi; sonsoles.lopez@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_8

words in abstracts and titles of articles in Sect. 4. Finally, the results are presented in Sect. 5 and discussed in Sect. 6. The chapter is concluded in Sect. 7.

## 1.1 Research Approach and Data

The following three research questions were set:

1. *What are the top keyword trends of CER ($RQ_1$)*
2. *What are the main research areas of CER ($RQ_2$)*
3. *What are the most relevant words in titles and abstracts with the highest growth rates? ($RQ_3$)*

This research is based on a comprehensive scientometric dataset of computing education research (CER). The data was obtained from Scopus database on January 24th, 2022. The process of data searching, retrieval, screening, and cleaning, followed the principles of the PRISMA-S (Preferred Reporting Items for Systematic reviews and Meta-Analyses) literature search extension [20]. Data was retrieved from two sources. First, a full download from a set of publication venues that are exclusively dedicated in publishing CER was performed. Second, a keyword query search from other venues was performed, and these two sets were joined. After screening, cleaning, and pre-processing, the result set contained 16,863 articles, including a total of 14,044 unique keywords [20].

This chapter is based on three analyses. First, trends of the top 20 keywords were analysed. Second, we used Structural Topic Modelling (STM), which is a method for studying text data across several disciplines [36, 52]. It is an unsupervised method that works without a-priori coding, and enables discovery of hidden (latent) structures in data [34]. STM offers an additional perspective over traditional keyword analysis, which may underline important themes of research, while focusing on most frequent single keywords. For analysis, the R package *stm* was used. The topics were modeled using the articles' metadata (title, abstract, keywords) as input [8, 35]. Keyword cleaning was performed by OpenRefine tool [20, 50]. We evaluated various combinations of topic models, ranging from models with 5 to 60 topics, by examining the combinations by three experts (the authors of this chapter) who made a choice for the best number of topics, based on the following criteria: (1) the meaningfulness of the keywords inside the topic, (2) minimal possible overlap with other topics, and (3) no significant ambiguity of the representative words. The evaluation resulted in a consensus of a model with 29 topics. Third, in analysing emerging common words, we used TF-IDF (term frequency-inverse document frequency) to detect the common words, and then identified words with the highest growth rates. Finally, a network of topics was created to show connections between topics [20].

## 2 The Research Areas of CER: Keyword Trends

### *2.1 A Brief Historical Overview of Research Areas*

A historical overview of influential research areas over the past decades shows the evolution of the publication profile of CER [3]. The times before 1970s focused much on various curriculum guidelines [39], recommendations, surveys, and reviews for building the grounds to teach the then-new discipline of computer science [3]. Central concerns of 1970s included those of how to teach programming, which programming language to use, and whether to focus on mathematics versus practical, hands-on skills [3]. The new computing curriculum CC'78 [6] was also launched, presenting a diversification from a mathematical view of computer science into applications, hands-on work and programming [6]. Works on human-computer interaction (HCI), teamwork in programming, tools and educational technologies and pedagogical aids also started [48]. In 1980s, debates on how to teach programming continued, an increase in empirical research was seen, and HCI and usability continued to mature as their own research tracks [3]. Also, Seymour Papert's creative ideas contributed to increased attention put into pedagogies. Pedagogical considerations deepened further in 1990s, while programming education continued to be a research area of high interest. Passionate discussions and debates around preferences for programming languages and programming paradigms have taken place in all decades of CER before the turn of the millennium [48], as well as after the turn of the millennium [3]. One of such debates has been about how and when to introduce object-orientation to students, with advocates for approaches of "objects-first" [9], "objects-late", and advocates for using, e.g., Java, Scala or Python as the preferred programming language. The 2000s were marked by continuing attention in programming education, increase in empirical research, and growth of tools-research, as seen, e.g., in the increase of research on areas such as visualization [3]. An increase in research on areas such as K-12 computing education and computational thinking was also perceived [3]. The 2000s were also marked by publication of the influential book on CER by Fincher and Petre in 2004 [14]. In 2010s, CER continued to mature as an established field of research, with the launch of new publication venues, professorships and other appointments [2]. Programming education continued to be a research area of high interest, and the amount of research on K-12 and computational thinking continued to increase, together with research on tools, learning analytics, and software engineering education [3]. An important milestone of 2010s is the publication of the Cambridge Handbook of Computing Education Research [15] in 2019.

### *2.2 Research Areas Through Analysis of Keywords*

After the turn of the millennium, CER continued to mature as a respectable research specialisation of its own, including the launch of new venues of dissemination, and

more strict demands for methodological rigor and empirical research [2]. The use of keywords also became a consistent practice, enabling reliable analysis of top keywords and keyword trends [3]. Recent research has analysed the frequencies and ranks of top keywords between 2000 and 2020, as well as proportions of top keywords between 2000 and 2020, as well as between 2011 and 2020 [3], revealing a unique picture about the evolution of the top research trends since the turn of the millennium. In this section we present the evolution of top 20 keywords' frequencies, from 2000, when keywords became available, to 2021. The analyses of keyword trends add to the narrative of evolving CER by providing increased quantitative precision to previous analyses of research trends in single publication outlets. Our analysis reveals the following top trends of research (see Fig. 1).

First, while historical accounts reveal the prominence of programming education as a hot area of research in CER, including debates about which programming language to use, the analysis of keywords (Fig. 1) shows that, programming education is, without any doubt, the most researched area of CER, which is confirmed by the popularity of the top keywords **CS1/CS2** (which refers to introductory computer science courses, typically introductory programming), **novice programmers** and **programming education**. Also **Java**, and **object-oriented programming** are among the top 20 keywords, but their popularity has been decreasing after 2010s. One reason may be the rising popularity of Python as a language for CS1, as well as Scratch and other block-based programming languages in 2010s [12]. Also, previous meta-studies have found a diminishing interest in research on what programming languages should be used [25], which may explain the

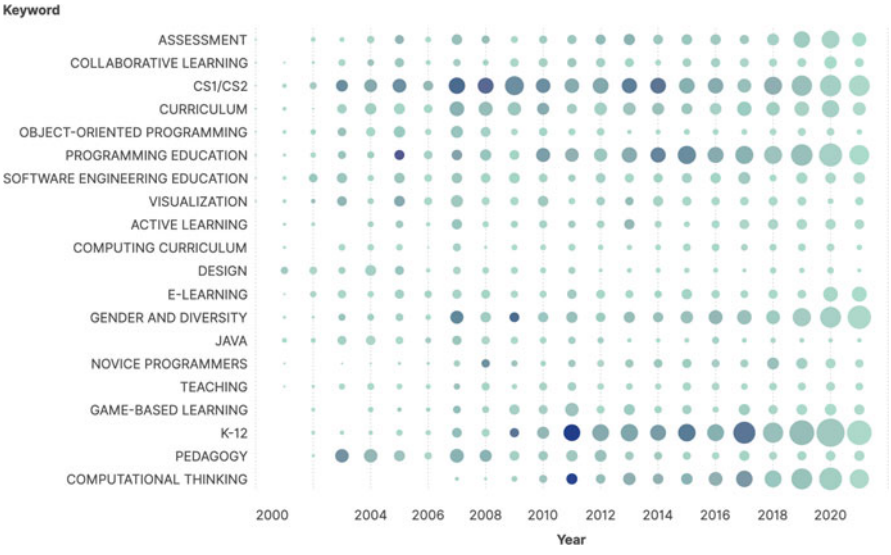


**Fig. 1** Top keywords' evolution in time (2000–2021). The circle size represents the frequency of articles, and the color represents number of citations (darker color means more citations)

sinking trend of these top keywords. Second, two keywords among the top 20 with an increasing trend are **K-12**, referring to computing education in school and **computational thinking** (CT). Third, research on pedagogical topics has its place among the top 20 keywords, with trends in the use of the following keywords: **active learning**, **assessment**, **collaborative learning**, **game-based learning**, **pedagogy** and **teaching**. Fourth, the presence of the tools-research is seen in the popularity of the keywords **visualisation**, referring to commonly used tools to assist in learning programming or data structures, and **e-learning** referring to a broad variety of learning tools used in CER. A probable explanation for the rise of e-learning is also the global COVID-19 pandemic and the related need to arrange education online. These findings complement recent analyses of top keywords [3], which shows the high popularity of tools research in **automatic assessment**, referring to tools used commonly, e.g., in introductory courses with large amounts of students, to ease off the workload of teachers and teaching assistants [23], and **LA-EDM** (learning analytics and educational data mining), referring to the application of a variety of tools and techniques of learning analytics in the context of computing education. Such tools may include, e.g., tools to predict high or low performing students, or detection of misconceptions in programming tasks. Fifth, other research areas within the top 20 keywords are: **curriculum** and **computing curriculum**, a common area in CER from the early times on, **software engineering education**, a research area that has always been a fundamental part of CER [6, 48], **gender and diversity**, a top trend of research in the recent decades, and **design**.

In all, recent analysis of keywords show that the mainstream tracks of research in CER since the turn of the millennium are: research on programming education; research on K-12 computing education, including computational thinking; research on pedagogical approaches in computing education, such as active, collaborative or game-based learning; tools research, including research on learning analytics tools, visualisation tools, and automatic assessment tools; research on software engineering education; and topics of gender and diversity. It is noteworthy that recent emerging trends may not become visible in an analysis that covers top keywords over two decades. We will look at emerging research in Sect. 4.

## 3 A Model of 29 Topics

The topic modeling resulted in a model of 29 topics, identified by unsupervised classification of keywords, and expert classification. The 29 topics can be grouped to several broad and partly overlapping themes. The 29 topics are summarised in Table 1. While topics and their keywords have connections with each other, there is not a single correct way to categorise them, as many topics are overlapping and could be categorised in several different ways. However, four (4) of the topics are, to a greater or lesser extent, related to ***introductory courses and programming***, including the topics *programming languages*, *programming*, *introductory courses*, and *OOP* (object-oriented programming). Together, the research under these topics

**Table 1** A topic model of 29 topics. Label is the name of the topic. Frequent keywords shows the top keywords in the topic

| Topic | Frequent keywords |
|---|---|
| Programming languages | Programming languages, programming, java programming language, computer programming languages, java, computer programming, high level languages, C (programming language) |
| Programming | Programming, computer programming, programming education, mathematical programming, novice, block-based programming, novice programmer, programming environment, scratch |
| Introductory courses | CS1/CS2, introductory, introductory programming, motivation, introductory computer science, engineering education, engineering research, courses, computer science courses |
| OOP | Object-oriented programming, object-oriented, abstracting, engineering research, innovation |
| Curriculum | Curriculum, information technology, computing, engineering education, computing curricula, societies and institutions, information systems, curriculum development, interdisciplinary, computer science curricula, cybersecurity, information |
| Classroom pedagogy | Environments, classroom, flipped, blended, engineering education, e-learning, distance learning |
| Pedagogy | Pedagogy, research, CS education, pedagogical approach, teaching assistants, engineering education, qualitative research |
| Educational psychology | Self-efficacy, attitudes, surveys, undergraduate students, behavioral research |
| Collaborative learning | Collaborative learning, collaborative, pair, pair programming, computer supported collaborative work, empirical studies |
| Assessment | Assessment, algorithms, data structures, automatic assessment, grading |
| Games | Games, game-based learning, computer games, game design/development, informatics education, interactive computer graphics |
| STEM | STEM, science, engineering, physical, computational, high school students |
| Tools | Educational tools, tool, interactive, interactive learning, systems, environments |
| Educational Technology | Educational technology, websites, technology, world wide web, multimedia systems, social networking (online), human computer interaction |
| Visualisation | Visualization, animation, data visualization, parallel programming |
| Online learning | E-learning, computer aided instruction, online, learning environments, distance education, online systems, intelligent tutoring systems |
| Robotics | Robotics, robot programming, educational robotics, databases |
| Software engineering | Software engineering, computer software, problem solving, software design, engineering education, project management, problem-solving, development |
| Projects | Projects, project-based learning, capstone, experiential learning, open source software |

(continued)

**Table 1** (continued)

| Topic | Frequent keywords |
|---|---|
| Design | Design, human factors, human engineering, instructional design, user experience |
| Software testing | Software testing, test-driven development, engineering research, engineering education, testing |
| Information systems | Information systems, engineering education, formal methods, information use |
| Data mining | Data mining, LA/EDM, codes (symbols), source codes |
| AI & ML | Learning systems, artificial intelligence, machine learning, engineering education, mathematical models, active learning |
| Computer architecture | Computer architecture, computer hardware, program compilers, simulation, hardware, embedded systems |
| Operating systems | Operating systems, security and privacy, cryptography, computer systems, computer operating systems |
| Computational thinking | K-12, computational thinking, teachers, training, high school, professional development, engineering education, primary school |
| Gender and diversity | Gender and diversity, women, broadening participation, diversity, engineering education, K-12 |
| Other | Computing, computation theory, computer applications, research questions, mobile, engineering research, teaching and learning |

and their keywords forms a massive part of CER, and is well aligned with the keyword analysis of top research trends in Sect. 2.2, as well as findings from previous research [3, 25, 29], showing the dominance of research on introductory courses and programming education in CER. Programming education, indeed, has deep roots in history, from the publication of the first textbooks about programming in 1951, the emerging software industries in 1960s, and within the influential ACM curriculum recommendations [6]. How to teach programming has been a central topic for debates during 1970s and 1980s [45, 48], and continued to be so in 1990s and 2000s as well, with a number of seminal papers published (e.g. [28, 31]). The strong foothold that programming education has in CER is well visible in numerous previous analyses of CER [3, 5, 42].

The topic of ***curriculum*** represents research on curriculum initiatives, a top trend of research through all times of CER. Several topics are associated with ***pedagogies***, such as those of *classroom pedagogy*, *pedagogy*, *educational psychology*, *collaborative learning*, and *assessment*. The topic of *games* includes research on game-based learning and education of game design and development, while *STEM* consists of research that deals with building bridges between learning science, technology, engineering, and mathematics, often, but not always, connected to computational thinking or the K-12 context. The topic of *online learning* is overlapping with ***pedagogies*** and ***tools research***, a common branch of CER. Pedagogical approaches reflected in the analysis of topics include keywords of common pedagogical setups, such as flipped or blended learning, role of teaching assistants, and distance learning. Topics on collaborative learning and online learning include keywords that

resemble research on pedagogies that are based on learning in pairs or small groups, and more broadly the research track of computer-supported collaborative learning (CSCL), through pedagogies, which combine both online and collaborative learning. The topic of assessment captures research under the keywords assessment and grading, as well as automatic assessment, which resembles research on automatic assessment tools, in the intersection of pedagogies and tools research. Research on ***educational tools*** is a mainstream branch in CER, and is represented through various topics, including those of *tools*, *educational technology*, *visualization*, with topics such as interactive learning systems, program visualization tools, and educational robotics. Tools-research is a unique branch of CER, and originates from the fact that many CER researchers have a background in computer science, which makes them capable in building new tools and technologies to address educational challenges, a unique capability specific to CER [26].

Several topics are highly associated with ***software engineering education***, and they represent a diverse range of topics including software design, project management, project-based learning, capstone courses, instructional design, software testing, and formal methods, all representing central research topics in software engineering education. Software engineering education has been a fundamental part of CER over the decades, ever since the birth of the software industry [3]. One topic of *information systems* represents this closely connected discipline and its associations with e.g. software engineering. Several topics are associated with ***artificial intelligence in education***, including research on *data mining* and *learning analytics and educational data mining*, representing the branch of research where educational data is analysed for the purposes of improving learning and teaching prospects [17]. The topic of *AI & ML* also covers approaches for teaching AI and ML, an emerging area of research in CER [51]. Two topics of *computer architecture* and *operating systems* represent CER on topics of computer architecture, computer hardware, operating systems, compilers, and embedded systems.

Research on the megatrending topic of ***computational thinking*** is well captured in our analysis by one topic, which includes keywords of computational thinking and many keywords that point to school context: K-12, high school, primary school, and e.g. to teacher training, one important subarea of research on K-12 and computational thinking [10]. Finally, ***gender and diversity*** consists of keywords related to broadening participation and issues of equality, while ***other*** keywords are related to education of computational theory, as well as computer applications. In all, the 29 topics represent the largest and mainstream topics and themes of CER. One must note that almost all the topics in CER are interconnected to a greater or lesser extent. For example, *engineering education* is known to be a neighboring discipline of CER [21], and its presence is seen in many of the topics. The diversity and overlap is also represented in the network of topics seen in Fig. 2, which shows the connections between the topics, and also reveals the hardship in categorising the topics under broader labels, as topics and underlying keywords form diverse connections with each other. It is also noteworthy that models have similar or overlapping keywords, which is a sign of many overlapping research areas, and perhaps in some cases, of inconsistent use of keywords.

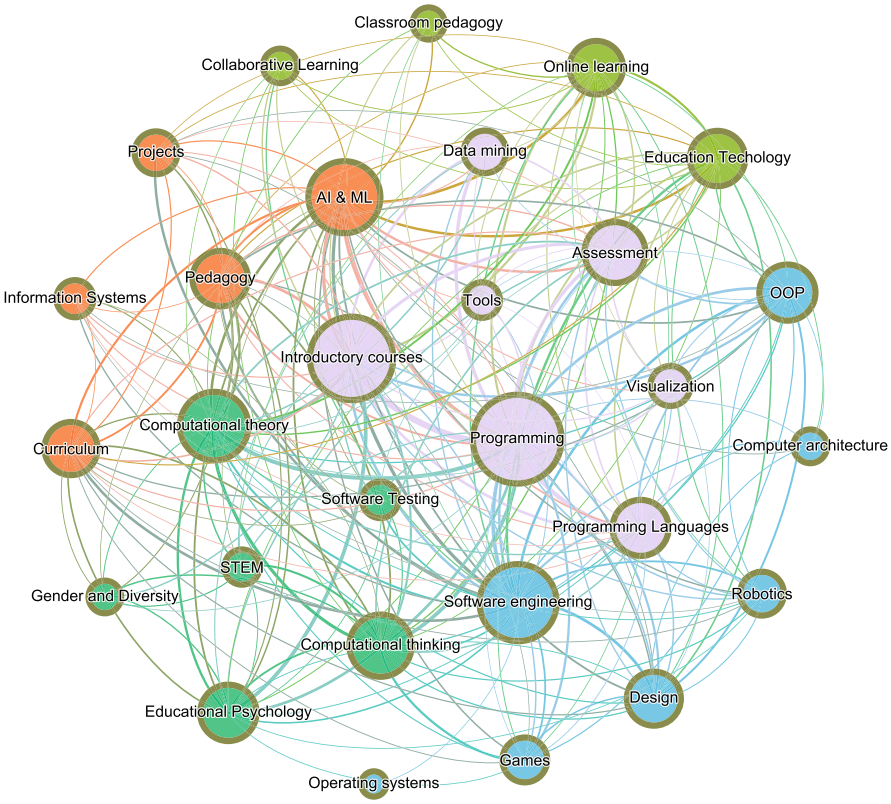


**Fig. 2** Network of Topics: connections between topics. If an article includes keywords from two topics, a connection between those topics is formed. The size of the circle is representative of the total count of keyword appearances for all keywords in the topic, while edge thickness denotes the frequency of co-occurrence. The colors indicate clusters of topics

Figure 3 visualises the evolution of CER within the model of 29 topics, capturing and representing the essence of the focus of research of CER, over the course of years, from 2001 onwards. The topics on **introductory courses**, **programming**, **computational thinking**, **other**, are the most popular models, followed by **AI & ML**, **educational psychology**, **curriculum**, and others, as observed by the frequencies in recent years. With regards to declining topics, **OOP** (object oriented programming) was peaking at around 2008, with a high yearly amount of citations, but has since declined. Similar observations have been found, e.g., with the decline of the keyword **Java** (Sect. 2.2). Analysis of meta-studies has also shown that research on which programming language to use has been on the decrease [25]. Research with steady popularity among the years include those of **operating systems**, and **visualization**.

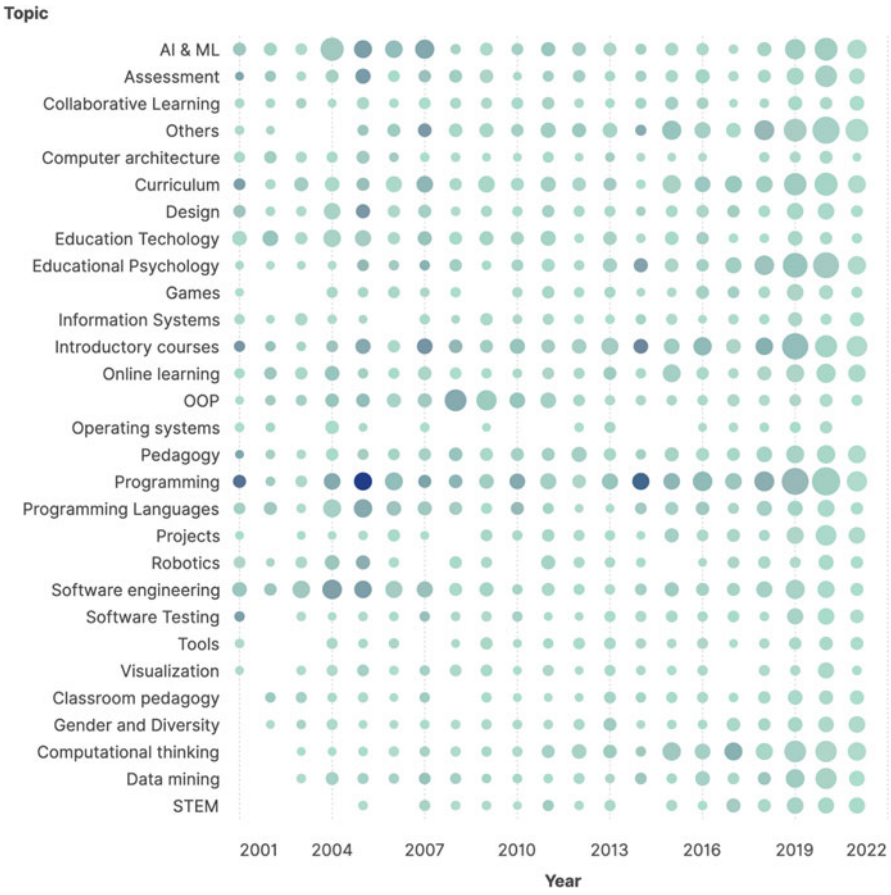


**Fig. 3** Evolution of Topics in Time. The size of the circle indicates frequency of published research, and the size of the circle indicates number of citations (darker color means more citations)

## 4 Emerging and Fast Growing Topics

We also analysed the titles and abstracts of all articles in the dataset for relevant words by using TF-IDF (term frequency-inverse document frequency). Of the relevant words, the ones with the highest trend of growth are shown in Fig. 4. The words *covid*, *pandemic*, *quantum*, *gamification* and *equity* are highly emerging in both 2020 and 2021. For 2020, emerging words include *tles* (two-line element sets), *adversarial*, referring to adversarial machine learning, *flipped* (flipped classroom pedagogy), *matchingref*, *dsdraw* for animations, *notebooks*, *worksheet*, and *DevOps*, referring to modern approaches in software engineering. In 2021, the emerging words refer to *esports* (electronic sports), *latinx* (a non-binary form Latino or

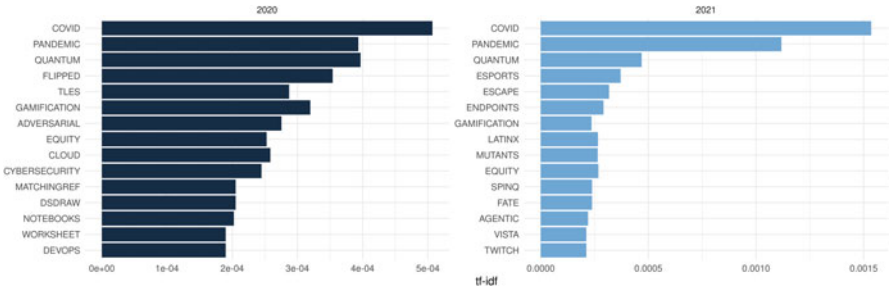


**Fig. 4** The emerging words identified in abstracts and titles by TF-IDF with highest yearly increment. The X-axis represents yearly increment

Latina), *SpinQ*, referring to a quantum computing platform, and *twitch*, referring to video game live streaming. These highest growing common words represent an assortment of emerging modern technologies, topics of computing, pedagogies and tools. Their analysis gives hints as to where CER may be headed at, after its heavy, decades long focus on areas of programming education and other classical topics.

## 5 Answers to Research Questions

In this chapter we have taken a peek into the research areas of CER. Previous analyses have shown how CER has, over time, matured from publishing mostly experience reports to exchange ideas of teaching practice, towards methodological and empirical rigor [25]. In addition, CER has broadened its research culture, integrated more with the learning sciences and engineering education research, started to appreciate learning theoretical constructs, and brought in new research designs such as those based on design research [4, 24, 27]. In this chapter, we have added to the previous analyses by introducing several new analyses, based on keyword trends and structural topic modelling (STM), an unsupervised classification of keyword metadata, which has identified a total of 29 topics. We have investigated the research areas of CER specifically from three perspectives; that of top keyword trends, topic modelling, and emerging research areas.

### *5.1 Top Keyword Trends*

Our first research question asked: *What are the top keyword trends of CER?* Our analysis of keywords complemented previous analyses of topics and keyword trends [3] in building a comprehensive picture about the topics of CER. Previous historical overviews have shown how CER has gradually evolved from early curriculum initiatives to bring computer science into colleges, rise of the software engineering

industry and related need to train software engineers, including heavy debates, e.g., between a need to focus on mathematics versus hands-on training, or on which programming language to use [48]. Previous analyses of keyword trends [3, 25] have shown top trending topics, such as research on introductory programming, computational thinking, and K-12 computing education [3]. Our analysis has complemented previous findings with the most recent data, and confirmed the prominence of programming education as a top topic, a rising trend of computational thinking and K-12, and prominence of certain pedagogical trends within CER, including those of collaborative learning, pair programming, flipped classroom, and game-based learning. Also, tools-research is clearly visible in the analysis, in addition to the highly popular trends of curriculum research, gender and diversity, and software engineering education.

## *5.2 Topic Modeling*

Our second research question asked: *What are the main research areas of CER?* The analysis of the identified topics through topic modelling adds to building a comprehensive picture about the mainstream research topics of CER, not only by inspecting individual and separate keyword trends, but by investigating groups of keywords that link together as topics. The method of unsupervised classification revealed a good fit of the data for a model with 29 topics, confirmed by expert classification by the authors of this chapter. The analysis of the 29 topics strengthened the understanding of top research areas by showing how keywords link with each other in the most common research tracks in CER. The strong presence of research on introductory courses and programming, curriculum, pedagogies, tools, software engineering education, and modern data analytics as research areas is confirmed, together with research on computational thinking, and diverse topics. Some topics in the generated model contain same keywords, which tells about the diversity and overlaps in research areas: research topics in CER do not form their isolated and independent silos, but research projects may span over different and diverse combinations of topics.

A network analysis of the 29 topics shows how these are connected with each other in clusters. One observation from the network analysis is how all research areas of CER are more or less connected with each other, while some areas, such as programming, introductory courses, and assessment, are more strongly connected. Analysis of the evolution of the topics in time shows that increasing or emerging topics include introductory courses, computational thinking, and other models, while declining topics include object-oriented programming and games. Top topics with a steady popularity include research on operating systems and visualization.

### *5.3 Emerging Research Areas*

Our third research question asked: *What are the most relevant words in titles and abstracts with the highest growth rates?* Analysis of relevant words in titles and abstracts of articles, as identified by TF-IDF, shows that highest growing emerging words include those related to covid-19 pandemic, and words related to adversarial machine learning, modern pedagogies such as flipped classroom, and modern approaches in software engineering, such as that of DevOps, as well as modern quantum computing platforms such as SpinQ. Reflections on the emerging topics and research areas of the future are provided in the following subsection.

## 6 Discussion

First, while previous analyses have shown that programming education is the all-time most researched area in CER, our analysis has complemented this finding by showing the dominance of programming among the top keywords, how programming is seen in topic modelling, and how topic modeling connects keywords that are related to introductory courses and programming. Novel findings as compared to previous analysis of keywords are the insight into programming education through a total of four topics, including a range of keywords. Such grouping to topics will make it possible to further analyse any set of topics and related keywords to perform a more in-depth investigation of any topical area, such as that of programming education. An alternative, scientometric view can complement the range of reviews and meta-studies already conducted on programming education [22, 37, 38]. Programming education has always received enormous attention within CER, from the introduction of the early programming textbooks in 1951, through the growth of the software industries, and as seen in various curriculum recommendations over the years [3, 48]. While empirical research on programming education started to be more common in 1980s [7], programming persisted as a central topic of research in 1990s, as well as 2000s and 2010s, too [3]. The dominance of programming education as a research topic is well visible in many previous analyses of publication trends in CER [5, 42–44, 49], and our analyses show that the amount of research on programming education is growing. While programming is quite a central topic in computing, some have started to question if such a heavy focus on programming education is justified, and if there are other important topics that would deserve more attention in the future. For example, many have stressed the importance of increasing research on education of AI and machine learning, as our world is becoming increasingly dependent on such systems, with hard-to-predict consequences [41, 46, 51]. Our analysis of emerging topics shows that the amount of research on machine learning and quantum computing have indeed experienced heavy growth in 2021 (Sect. 4). Other related discussions have been building, e.g., around the need to increase training in AI ethics education, as

massive populations of people are living among and being influenced by machine learning systems [13, 33].

Second, a strong presence of educational technology and tools is confirmed by the analysis. Tools-research has always been an important part of CER [42]. One reason is that many CER researchers have their basic training in computer science, which means that they possess the unique strength to design and develop new tools, as compared to, e.g., educational researchers [2]. In many cases, the topics of CER have their origins in concrete challenges of teaching praxis, and attempts to develop tailored tools or interventions to address those challenges [23]. Examples of such situations include massive numbers of students in introductory programming courses, and related efforts to develop tools on automatic assessment to ease the workload of teachers and teaching assistants in grading weekly assignments and final exams [23], learning challenges in the form of student misconceptions, and related efforts to automatically detect misconceptions and offer tailored assistance, and visualisation tools for help in comprehending the step-by-step execution of practice programs when learning to program. Other tools include, e.g., detection of students in risk of dropping out and offering related tailored interventions to assist such students [18], or even whole pedagogies that are heavily connected to an underlying educational technology [53]. Learning analytics tools are also visible in the analysis, and include e.g. the trending keyword learning analytics and educational data mining (LA/EDM). Modern approaches include e.g. detection of students' computational thinking in agent-based modelling activities, and related learning analytics of knowledge building pedagogies [1].

Third, the strong foothold of K-12 computing education and computational thinking in CER are well known [3], and is confirmed both through analysis of keywords, topics, and emerging topics. While the roots of CT are in 1950s and beyond, and in works of Knuth, Dijkstra, Papert, and many others [11], a seminal paper on CT was published in 2006 [54], after which, the amount of research on the topic increased significantly, with an annual growth rate of 61.2% [40]. While the large majority of CT approaches focus on rule-driven programming, CT 2.0 [47] brings a much needed update to CT by capturing additional and essential concepts, which are not well covered by the mainstream definition of CT, yet. Such additions are much relevant in the technological environment of today, and include e.g. neural networks, curating and training data, and reinforcement learning [47]. This is one trend of the future in CER, which has been understudied in the previous decade.

Fourth, software engineering has always had a strong foothold in CER, ever since the rise of the software industry starting from somewhere in 1960s. Along the years, subtopics of software engineering, such as usability and Human Computer Interaction (HCI) in 1980s, and user experience design (UX) have been brought along, together with topics such as software testing, and research on capstone projects, referring to courses where groups of students learn by participating in a software project. While all these typical approaches in software engineering are seen in the topic model analysis of this chapter, and software engineering education constantly appears as top keyword when analysing keyword trends, in recent times, many voices have started to stress the inadequacy of common methods

typical in software engineering, such as user requirement definition, and the need to bring understanding of communities, cultures, and habits into software development [4, 19, 32, 33], with increased attention on design skills. This approach, which one could label as SE 2.0, is one crucially important trend of the near future.

Fifth, other common topical areas in CER focus on pedagogies. One major contribution to pedagogies of CER is Seymour Papert's Mindstorm and constructionism [30]. Other common pedagogical contributions, visible in the topic modeling, include modern pedagogies of flipped classroom. Also, focus on collaborative learning and e.g. pair programming has a strong foothold in CER, together with pedagogies that focus on technology-mediated learning, such as computer aided instruction, distance education, and intelligent tutoring. Other common topical areas of CER, as shown by topic modelling, are: education of computer architectures and operating systems, and research on diverse topics, such as education of game-design and game-based learning, research focusing on gender and diversity, and computational theory. Analysis of topics with high growth rates include those on introductory courses and programming, computational thinking and computational theory, while declining topics include e.g. object-oriented programming. Steadily popular topics include games, operating systems, and visualization.

### *6.1 Limitations*

While Elsevier's Scopus has good accuracy as compared to some other databases, e.g. Web of Science [16], the data still contains numerous challenges. Some of the known problems with Scopus metadata include, but are not limited to: missing fields, inconsistent recording of keywords and references, mistakes in publication venues, missing data and even missing years in some venues. Many mistakes can be corrected by using manual and algorithmic methods. However, correcting all errors in the data is unfeasible. In addition, inconsistent use of keywords is a known issue among authors. With this data, the method of topic modelling could not fully separate overlapping research areas, which resulted in some keywords being among more than one category. All of these issues may have had an impact to the analyses and generated models. These limitations are fully explained in chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" of this book [20].

## 7 Conclusions

In all, the topics of CER are wide and rich. All the central topics and research areas are interconnected. The mainstream and dominating topics are programming education, computational thinking, K-12 computing education, software engineering education, and research on tools. A wide array of research is conducted under these broad themes. While this research has painted a macro-level view about the topics of

CER, future research could zoom in on more specific topics, such as, e.g. software engineering, or tools-research, and provide a more nuanced view of research in those areas. Findings from scientometric analyses must be accompanied with more in-depth meta-studies, which can more accurately cover nature of published papers, use of research methods, and building of theoretical frameworks, research methods, tools descriptions, and data collection [25]. Topic modeling has proved to be a valuable addition to complement keyword analysis by revealing nuances that can not be seen in analysis of plain keywords, revealing a broader landscape of how topics are grouped and connected with each other. All topics are more or less connected, and the boundaries between topics overlap with each other, while the boundaries evolve in time. One direction for the future is to take a closer inspection into the emerging trends of research with deep reflections about how CER needs to evolve, given the rapidly changing global technological milieu.

## References

1. Apiola, M., Lipponen, S., Seitamaa, A., Korhonen, T., Hakkarainen, K.: Learning Analytics for Knowledge Creation and Inventing in K-12: A Systematic Review. In: Lecture Notes in Networks and Systems (Proceedings of 2022 Computing Conference). Springer (2022)
2. Apiola, M., López-Pernas, S., Saqr, M.: The Venues that Shaped Computing Education Research: The Gatekeepers Under the Lens. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
3. Apiola, M., Saqr, M., López-Pernas, S., Tedre, M.: Computing education research compiled: Keyword trends, building blocks, creators, and dissemination. IEEE Access **10**, 27041–27068 (2022). https://doi.org/10.1109/ACCESS.2022.3157609
4. Apiola, M., Sutinen, E.: Design science research for learning software engineering and computational thinking: Four cases. Computer Applications in Engineering Education, pp. 1–19 (2020). https://doi.org/10.1002/cae.22291
5. Apiola, M., Tedre, M., López-Pernas, S., Saqr, M., Daniels, M., Pears, A.: A Scientometric Journey Through the FIE Bookshelf: 1982–2020. In: 2021 IEEE Frontiers in Education Conference (FIE), pp. 1–9 (2021). https://doi.org/10.1109/FIE49875.2021.9637209
6. Austing, R.H., Barnes, B.H., Bonnette, D.T., Engel, G.L., Stokes, G.: Curriculum '78: Recommendations for the undergraduate program in computer science– a report of the ACM curriculum committee on computer science. Communications of the ACM **22**(3), 147–166 (1979). DOI http://doi.acm.org/10.1145/359080.359083
7. Butcher, D.F., Muth, W.A.: Predicting performance in an introductory computer science course. Commun. ACM **28**(3), 263–268 (1985). URL https://doi.org/10.1145/3166.3167
8. Chen, X., Zou, D., Cheng, G., Xie, H.: Detecting latent topics and trends in educational technologies over four decades using structural topic modeling: A retrospective of all volumes of computers & education. Computers & Education **151**, 103855 (2020). URL https://doi.org/10.1016/j.compedu.2020.103855
9. Cooper, S., Dann, W., Pausch, R.: Teaching objects-first in introductory computer science. In: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '03, pp. 191–195. Association for Computing Machinery, New York, NY, USA (2003). URL https://doi.org/10.1145/611892.611966
10. Dagienė, V., Gulbahar, Y., Grugurina, N., López-Pernas, S., Saqr, M., Apiola, M., Stupurienė, G.: CER in Schools. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)

11. Denning, P.J., Tedre, M.: Computational Thinking. Essential Knowledge Series. The MIT Press (2019)
12. Fagerlund, J., Häkkinen, P., Vesisenaho, M., Viiri, J.: Computational thinking in programming with scratch in primary schools: A systematic review. Computer Applications in Engineering Education **29**(1), 12–28 (2021). DOI https://doi.org/10.1002/cae.22255
13. Fiesler, C., Garrett, N., Beard, N.: What do we teach when we teach tech ethics? a syllabi analysis. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20, pp. 289–295. Association for Computing Machinery, New York, NY, USA (2020). URL https://doi.org/10.1145/3328778.3366825
14. Fincher, S., Petre, M.: Computer Science Education Research. Taylor & Francis (2004)
15. Fincher, S.A., Robins, A.V. (eds.): The Cambridge Handbook of Computing Education Research. Cambridge University Press (2019). https://doi.org/10.1017/9781108654555.001
16. Franceschini, F., Maisano, D., Mastrogiacomo, L.: Empirical analysis and classification of database errors in scopus and web of science. Journal of Informetrics **10**(4), 933–953 (2016). DOI https://doi.org/10.1016/j.joi.2016.07.003
17. Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S.H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M.A., Sheard, J., Skupas, B., Spacco, J., Szabo, C., Toll, D.: Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. In: Proceedings of the 2015 ITiCSE on Working Group Reports, ITICSE-WGR '15, pp. 41–63. ACM, New York, NY, USA (2015). URL http://doi.acm.org/10.1145/2858796.2858798
18. Kaila, E.: Utilizing Educational Technology in Computer Science and Programming Courses. Ph.D. thesis, Turku Centre for Computer Science (University of Turku, Department of Future Technologies) (2018)
19. Kelly, K.: The Inevitable: Understanding the 12 Technological Forces That Will Shape Our Future. Penguin Books (2017)
20. López-Pernas, S., Saqr, M., Apiola, M.: Scientometrics: A Concise Introduction and a Detailed Methodology for the Mapping of the Scientific Field of Computing Education Research. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
21. Loui, M.C., Borrego, M.: Engineering Education Research. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research. Cambridge University Press, pp. 292–321 (2019)
22. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory programming: A systematic literature review. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2018 Companion, pp. 55–106. Association for Computing Machinery, New York, NY, USA (2018). URL https://doi.org/10.1145/3293881.3295779
23. Malmi, L., Hellas, A., Ihantola, P., Isomöttönen, V., Jormanainen, I., Kilamo, T., Knutas, A., Korhonen, A., Laakso, M.J., Poranen, T., Salakoski, T., Suhonen, J., and, S.L.P.: Computing Education Research in Finland. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
24. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Kinnunen, P., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Theoretical underpinnings of computing education research: What is the evidence? In: Proceedings of the Tenth Annual Conference on International Computing Education Research, ICER '14, pp. 27–34. Association for Computing Machinery, New York, NY, USA (2014). URL https://doi.org/10.1145/2632320.2632358
25. Malmi, L., Simon, Sheard, J., Kinnunen, P., Sinclair, J.: The Evolution of Computing Education Research: A Meta-Analytic Perspective. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
26. Malmi, L., Utting, I., Ko, A.J.: Tools and Environments, pp. 639–662. The Cambridge Handbook of Computing Education Research. Cambridge University Press (2019). https://doi.org/10.1017/9781108654555.022

27. Margulieux, L.E., Dorn, B., Searle, K.A.: Learning Sciences for Computing Education. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, chap. 8, pp. 208–230. Cambridge University Press, Cambridge (2019)
28. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., Laxer, C., Thomas, L., Utting, I., Wilusz, T.: A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students. In: Working group reports from ITiCSE on Innovation and technology in computer science education, ITiCSE-WGR '01, pp. 125–180. ACM, New York, NY, USA (2001). URL http://doi.acm.org/10.1145/572133.572137
29. Papamitsiou, Z., Giannakos, M., Simon, Luxton-Reilly, A.: Computing education research landscape through an analysis of keywords. In: Proceedings of the 2020 ACM Conference on International Computing Education Research, ICER '20, pp. 102–112. Association for Computing Machinery, New York, NY, USA (2020). URL https://doi.org/10.1145/3372782.3406276
30. Papert, S.: MINDSTORMS: Children, Computers, and Powerful Ideas. Basic Books (1980)
31. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., Paterson, J.: A Survey of Literature on the Teaching of Introductory Programming. SIGCSE Bulletin **39**, 204–223 (2007). URL http://doi.acm.org/10.1145/1345375.1345441
32. Pears, A., Tedre, M., Valtonen, T., Vartiainen, H.: What makes computational thinking so troublesome? In: To Appear in FIE'21 Frontiers in Education Conference (2021)
33. Raji, I.D., Scheuerman, M.K., Amironesei, R.: You Can't Sit With Us: Exclusionary Pedagogy in AI Ethics Education. In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21, pp. 515–525. Association for Computing Machinery, New York, NY, USA (2021). URL https://doi.org/10.1145/3442188.3445914
34. Roberts, M.E., Stewart, B.M., Airoldi, E.M.: A model of text for experimentation in the social sciences. Journal of the American Statistical Association **111**(515), 988–1003 (2016). URL https://doi.org/10.1080/01621459.2016.1141684
35. Roberts, M.E., Stewart, B.M., Tingley, D.: Stm: An r package for structural topic models. Journal of Statistical Software **91**(2) (2019). URL https://doi.org/10.18637/jss.v091.i02
36. Roberts, M.E., Stewart, B.M., Tingley, D., Lucas, C., Leder-Luis, J., Gadarian, S.K., Albertson, B., Rand, D.G.: Structural topic models for open-ended survey responses. American Journal of Political Science **58**(4), 1064–1082 (2014). URL https://doi.org/10.1111/ajps.12103
37. Robins, A., Rountree, J., Rountree, N.: Learning and Teaching Programming: A Review and Discussion. Computer Science Education **13**(2), 137–172 (2003)
38. Robins, A.V.: Novice programmers and introductory programming. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, pp. 327–376. Cambridge University Press (2019). https://doi.org/10.1017/9781108654555.001
39. Salton, G.: Information science in a ph.d. computer science program. Commun. ACM **12**(2), 111–117 (1969). URL https://doi.org/10.1145/362848.362871
40. Saqr, M., Ng, K., Oyelere, S.S., Tedre, M.: People, ideas, milestones: A scientometric study of computational thinking. ACM Trans. Comput. Educ. **21**(3) (2021). URL https://doi.org/10.1145/3445984
41. Shapiro, B., Fiebrink, R., Norvig, P.: How machine learning impacts the undergraduate computing curriculum. Communications of the ACM **61**(11), 27–29 (2018)
42. Simon: Emergence of computing education as a research discipline. Ph.D. thesis, Aalto University School of Science (2015)
43. Simon: Twenty-two years of ace. In: Proceedings of the Twenty-Second Australasian Computing Education Conference, ACE'20, pp. 203–210. Association for Computing Machinery, New York, NY, USA (2020). URL https://doi.org/10.1145/3373165.3373188
44. Simon, Sheard, J.: Twenty-Four Years of ITiCSE Papers. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, pp. 5–11. Association for Computing Machinery, New York, NY, USA (2020). URL https://doi.org/10.1145/3341525.3387407
45. Soloway, E.: Learning to Program = Learning to Construct Mechanisms and Explanations. Communications of the ACM **29**(9), 850–858 (1986). URL http://doi.acm.org/10.1145/6592.6594

46. Tedre, M., Denning, P., Toivonen, T.: Ct 2.0. In: 21st Koli Calling International Conference on Computing Education Research, Koli Calling '21. Association for Computing Machinery, New York, NY, USA (2021). URL https://doi.org/10.1145/3488042.3488053
47. Tedre, M., Denning, P., Toivonen, T.: Ct 2.0. In: 21st Koli Calling International Conference on Computing Education Research, Koli Calling '21. Association for Computing Machinery, New York, NY, USA (2021). URL https://doi.org/10.1145/3488042.3488053
48. Tedre, M., Simon, Malmi, L.: Changing aims of computing education: a historical survey. Computer Science Education **28**(2), 158–186 (2018). URL https://doi.org/10.1080/08993408.2018.1486624
49. Valentine, D.W.: Cs educational research: A meta-analysis of SIGCSE technical symposium proceedings. SIGCSE Bull. **36**(1), 255–259 (2004). URL https://doi.org/10.1145/1028174.971391
50. Valtonen, T., López-Pernas, S., Saqr, M., Vartiainen, H., Sointu, E.T., Tedre, M.: The nature and building blocks of educational technology research. Computers in Human Behavior **128**, 107123 (2022). URL https://doi.org/10.1016/j.chb.2021.107123
51. Vartiainen, H., Toivonen, T., Jormanainen, I., Kahila, J., Tedre, M., Valtonen, T.: Machine learning for middle schoolers: Learning through data-driven design. International Journal of Child-Computer Interaction **29**, 100281 (2021). DOI https://doi.org/10.1016/j.ijcci.2021.100281.
52. Vayansky, I., Kumar, S.A.: A review of topic modeling methods. Information Systems **94**, 101582 (2020). URL https://doi.org/10.1016/j.is.2020.101582
53. Vihavainen, A., Paksula, M., Luukkainen, M.: Extreme Apprenticeship Method in Teaching Programming for Beginners. In: Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education, SIGCSE '11, pp. 93–98. ACM, New York, NY, USA (2011). URL http://doi.acm.org/10.1145/1953163.1953196
54. Wing, J.M.: Computational thinking. Communications of the ACM **49**(3), 33–35 (2006)

# Capturing the Impact and the Chatter Around Computing Education Research Beyond Academia in Social Media, Patents, and Blogs


**Mohammed Saqr, Sonsoles López-Pernas, and Mikko Apiola**

## 1 Introduction

Altmetrics was conceived over a decade ago as *alternative metrics* to the commonly used sciento-*metrics* for the evaluation of research impact, e.g., citation counts and H-index [1–3]. The main driver behind the initial idea was to help researchers use the wisdom of the web and social media crowds as curators of relevant research, given the burgeoning number of research published every day [1]. Another reason was to capture the online scientific conversations as scientists and the public engage in discussions about academic scholarly work [1, 3]. Such conversations take place on a wide array of non-scholarly platforms which include Twitter, Facebook, blogs, LinkedIn, etc. Altmetrics also captures other sources such as mentions by Wikipedia**,** policy websites, syllabi, as well as Mendeley readers [4]. In doing so, Altmetrics captures the attention an article gets from the wider community, the public's reactions to it (likes, retweets, saves, page views, downloads, etc.) and the exposure (page views or hits) [4–6]. Taken together, Altmetrics offers a measure of dissemination of scholarly works, as well as an indication of their influence and impact across the Internet audience at large [7].

While Altmetrics have been around for over a decade, little—if any at all—is known about how Computing Education Research (CER) researchers have embraced the idea or how spreading the word about research helps attract the attention of other researchers or boost research impact. What is more, no previous study has captured the conversation on social media, blogs, and news about CER: a gap which this article aims to bridge. We take advantage of the latest advances in analytics and offer a comprehensive analysis of CER in social media, news, and patents. This article is structured as follows: First, we offer a background about

M. Saqr (✉) · S. López-Pernas · M. Apiola
University of Eastern Finland, Joensuu, Finland
e-mail: mohammed.saqr@uef.fi; mikko.apiola@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_9

Altmetrics. Then, we review the main sources of Altmetrics data and, later, we describe the motivation of the study. The background section is followed by the methods, the results and the discussion.

## 2 Background

Compared to traditional scholarly metrics such as citations, Altmetrics are faster to curate. In traditional scholarly journals, when a paper cites a given paper, it takes time until the citing paper gets published and the citations are recorded [3]. Conversely, web mentions—the subject of Altmetrics—are immediately available and therefore, they are easier to accumulate and monitor as soon as they are online [4]. Another advantage of Altmetrics is the diversification of how we measure research impact and influence, which allows us to understand—in real time—if and to which extent a paper has succeeded in garnering attention as well as by whom and when [3, 8]. Such advantages in diversity, speed and scale have made Altmetrics a valuable tool used by the public, policy makers, funders as well as the academics [5]. Nowadays, most publishers embed Altmetrics within their platforms or have alternative solutions that offer similar functionality, e.g., Plumx and ImpactStory [4]. Libraries—the traditional curators of knowledge—have started to capitalize on the potential of Altmetrics and the insights it offers into what could be of interest to curate or subscribe to.

As a quantitative tool, Altmetrics suffers from the same drawbacks of traditional citation counts and metrics [6, 8]. Quantifying research work—as numbers of impact—is an oversimplification of a complex reality that both Altmetrics and citation counts suffer from [9, 10]. Furthermore, Altmetrics—being collected from the wider web—are more subject to manipulation by, e.g., Twitter bots [11]. In the case of blogs, it is unclear how credible the mentions that Altmetrics collects are, e.g., is every blog site a legitimate site? If so, how can we weigh the evidence we get from such a blog given its nature as a non-peer reviewed site? [4, 7]. What is more, not all articles make it to the social web; many authors are not social media users, and only a fraction of journals has social media presence. Consequently, Altmetrics information is available about just a "slice" of the scholarly works [4]. It should be noted that absence of social media presence can be mistakenly—and should not be—considered as a sign of lack of impact. Researchers may share their articles using non-standard links (shortened links), or links to online repositories, e.g., the University version of the article or ResearchGate. These links are not collected or curated by Altmetrics.

Several studies have evaluated the concordance between Altmetrics and traditional metrics (citation counts and H-index) in the measurement of impact. For such a purpose, most researchers have performed correlations between the Altmetrics indicators (e.g., Mendeley reads, Twitter posts) and traditional indicators (citations count and H-index). While the results vary, there is an agreement that a correlation exists between the number of citations and Altmetrics indicators. Some studies have

found that blog count is the best predictor of citation count (e.g., [6]), while other studies have pointed to Mendeley reader count as the best predictor of citations and the H-index of the author [5]. Most of the studies, however, also agree that social media usage correlates weakly with citation count [5, 6, 12]. A meta-analysis of 40 metrics found an overall moderate to weak correlation between Altmetrics and citation counts. The authors of the meta-analysis concluded that results "do indeed measure a different kind of research impact, thus acting as a complement rather than a substitute to traditional metrics" [7].

Our review of the literature leads us to some important conclusions regarding Altmetrics that are relevant to the current study. Altmetrics offers a different approach to measure reach impact and reach over the social web at large. Altmetrics data is a complementary piece of the story, an enhancement to our knowledge of how research has been part of the public discourse. In the next section we offer an overview of the main sources of information in Altmetrics which we will cover in our research.

## *2.1 Twitter*

Twitter is a social media platform launched in 2006 as a microblogging website where users could use their phone text messaging to post brief blogs. Soon after its launch, Twitter usage grew to include all aspects of life including scholarly work [13]. Today, Twitter stands as the most studied metric in the Altmetrics literature [4]. Among researchers, Twitter usage ranges from 5% to 32% including personal and professional use [14]. Yet, it is also well-known that regular activity is low in Twitter and therefore, the actual activity of tweeting or interacting about research work could be far lower. Altmetrics coverage of articles on Twitter increased from 10–15% in 2012–2014 to about 40% in 2018 [4]. Several studies have confirmed the correlation between Twitter mentions and traditional citation count. Yet, correlation was mostly weak in general [7]. However, to what extent tweeting causes or leads to article citations is controversial, i.e., has tweeting caused the citations or is it the impact of the article that resulted in tweeting?

## *2.2 Mendeley*

Mendeley was launched in 2008 as an attempt to allow researchers to share their favorite scholarly work with their colleagues. Since then, Mendeley has grown in adoption, functionality, and importance. Mendeley offers a cross-platform application, a website, and a smartphone application which allow users to share their curated papers, annotate as well as to handle their references [15]. Mendeley is known to have the highest coverage by Altmetrics of articles among all social media platforms, reaching up to 90% of all published articles [14]. Yet, Mendeley is used

by just 5–10% of all researchers and is mostly dominated by early-stage researchers. Therefore, the number of Mendeley readers should be viewed as only an indication of readership representing a certain demographic, not the actual number of readers at large [4, 8]. Nonetheless, as discussed earlier, Mendeley readership is one of the strongest and most consistent predictors of article impact [7].

## *2.3 News*

Mentions by news outlets, as the name implies, capture the attention an article gets from news sources, e.g., mainstream media and press releases. Unlike Twitter and Mendeley, news mentions are based on multiple sources [4, 14]. The list of outlets and the scope of coverage are not clear and Altmetrics announces expansion to the list overtime. Thereupon, news can be viewed as a relative proxy indicator rather than a concrete measure of news attention. Still, attention by news is a good indication of how news outlets engage with research findings. Altmetrics' coverage of research articles remains low with a range from 0.1% in 2012 to 3.8% in 2017 [4], which can be attributed to the idea that not all articles receive news coverage or trigger the attention of news outlets.

## *2.4 Blogs*

Blogging started as early as 1990 before most other social networking sites and contributed—at least partially—to the rise and spread of social networking. Many scientists blog continuously about their research, or science in general and enjoy the rich conversations blogging brings. Blogging enables scientists to summarize their research, disseminate it to the public, comment on others' work or criticize research or academic life in general. Blogging can be performed on dedicated web sites or services, e.g., WordPress, Blogspot or LiveJournal or as part of University personal web pages or research networking sites such as ResearchGate. Such diversity and source fragmentation has led to the rise of blog aggregation, i.e., sites that help collect different blogs of interest. However, systemic aggregation of blogs is a difficult task and the list and extent of coverage by Altmetrics is unclear and probably underestimates the full breadth of academic blogging. Yet, blogs are a good proxy indicator of academic dialogue or an "alternative" platform for summarizing research findings. Altmetrics coverage of articles ranged from 0.6% in 2012 to 8.8% in 2018. The coverage varies by discipline and demographic, i.e., socio demographics affect blogging and engagement with blogs [4, 14].

### 2.5 Other Sources

Altmetrics also monitors a wide range of public policy documents that mentions public research, Wikipedia mentions (or references) of a public article, patents, data from the Open Syllabus Project regarding usage of the published research in syllabi as well as peer-reviewed reporting sources, e.g., Publons. In addition, Altmetrics monitors other social media sites, e.g., Facebook pages, LinkedIn, YouTube, Reddit and the popular questions and answers site Stack overflow [4].

### 2.6 The Motivation for this Study

Research impact goes beyond academia and exists in the multiplicity of digital platforms that we use to read, share, and discuss knowledge. CER is no exception: although research is created in academia—the typical research institutions—, it is talked about widely on social media, blogs, and news websites. The literature review presented earlier shows that Altmetrics have become a mainstream platform for measuring research attention and dissemination on such platforms. While not perfect, it tells an important part of the story. Therefore, the aim of this study is to have a comprehensive look at how research in CER has been received, talked about in social media, discussed on blogs, and penetrated to the news and other traditional media.

## 3 Methods

The data for this chapter were obtained from the Altmetrics website using two methods. First, Altmetrics data of all Digital Object Identifiers (DOI) of the 16,383 articles from the CER dataset described in chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" of this book [16] were retrieved from Altmetrics.com using its API. Second, a search for titles, keywords, and venues (dedicated journals and conferences of CER) was performed to retrieve articles that do not have a DOI or may have been recorded by title or without their DOI. The first method retrieved 1712 articles, and the second resulted in 1360 articles. The two datasets were combined, and duplicates were removed, resulting in a final dataset containing 2336 articles (13.9% of the CER articles) with at least one Altmetrics data field (e.g., Twitter mentions). The data were analyzed and visualized using the R programming language [17]. In addition to common analysis of trends of growth, we analyze trends of usage of social media and quantitative analysis of platforms, articles, and venues. The analysis also includes which articles in which subfields had a wide impact, and for whom (i.e., which platforms had more impact). Furthermore,

such analysis includes the themes of research that have garnered more attention from social media users. These themes were extracted using structural topic modeling as described in detail in [18]. Such results are discussed with an in-depth qualitative analysis that reflects on the value and importance of the findings.

## 4 Results

Although Altmetrics was launched in 2011, some 728 of the papers published before this year had Altmetrics data (representing 31.2% of all the papers with Altmetrics data, which makes around 8.7% of the papers published to that date). Of all Altmetrics sources, Mendeley had the highest coverage. The average number of tweets per article in the pre-Altmetrics era (before 2011) was 0.39 compared to 3.98 in the post-Altmetrics era; such a large difference was statistically significant $t(1715.89) = 11.81$, $p < .001$). The average number of Mendeley readers for the pre-Altmetrics era was 37.7, compared to 45.6 in the post-Altmetrics era and the difference was statistically insignificant. The descriptive statistics for papers with Altmetrics data are presented in Table 1. As the table shows, except for Mendeley and Twitter mentions, the numbers were very low. The average number of Twitter mentions was 2.87 on average (SD = 10.11, Median = 1.00). News mentions were low (mean = 0.07, SD = 0.64, Median = 0.00) and so were blog mentions (mean = 0.06, SD = 0.29, Median = 0.00). The mean number of Mendeley readers was 43.17 (SD = 91.64, Median = 21). Wikipedia mentions (mean = 0.09, SD = 0.42, Median = 0.00), and Facebook mentions (mean = 0.03, SD = 0.21, Median = 0.00) were also very low.

The trend of growth of Altmetrics data in Fig. 1 shows that only Twitter mentions are up-trending while all other trends are irregular or trending down. It is noteworthy to mention that each service has different time dynamics. For instance, in Mendeley, it is conceivable that the older a paper is, the more likely it is to be read by more

**Table 1** Mention statistics per source (n = 2336)

| | Median | Mean | Std. deviation | Minimum | Maximum |
|---|---|---|---|---|---|
| Twitter mentions | 1.00 | 2.87 | 10.11 | 0 | 313 |
| Number of Mendeley readers | 21.00 | 43.17 | 91.64 | 0 | 1836 |
| Facebook mentions | 0.00 | 0.03 | 0.21 | 0 | 4 |
| Blog mentions | 0.00 | 0.06 | 0.29 | 0 | 6 |
| News mentions | 0.00 | 0.07 | 0.64 | 0 | 16 |
| Patent mentions | 0.00 | 0.19 | 2.62 | 0 | 118 |
| Policy mentions | 0.00 | 0.03 | 0.18 | 0 | 2 |
| Q&A mentions | 0.00 | 0.01 | 0.08 | 0 | 2 |
| Video mentions | 0.000 | 0.004 | 0.062 | 0.000 | 1.000 |
| Wikipedia mentions | 0.000 | 0.086 | 0.422 | 0.000 | 6.000 |

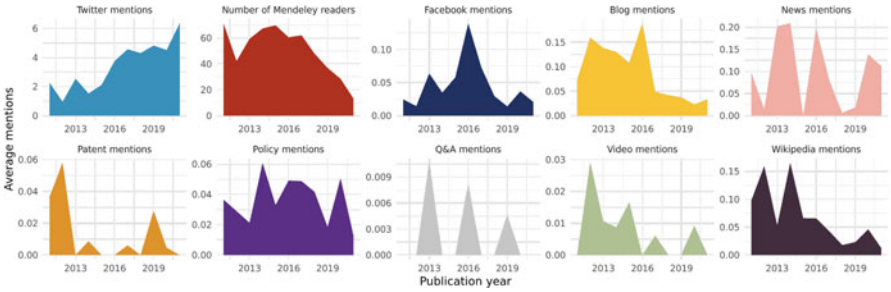


**Fig. 1** Evolution of mention statistics per source

readers and therefore, the trend should not be interpreted as a decreasing *number of readers*, but rather that older papers have been read more times overall. Similar to Mendeley, the patents, policy and Wikipedia mentions are expected to cite highly regarded or well-established papers. On the other hand, in news and blogs, recent papers are expected to make it to the news or be blogged about by authors. Similarly, Twitter mentions are expected to grow with time as researchers turn to Twitter to discuss the emerging research. Yet, it is hard to infer any future trends from other services, as the number of paper mentions is small and current trends are irregular.

## *4.1 Twitter Mentions*

The oldest paper that has Altmetrics data in our dataset was published in 1968 [19]. The paper was tweeted by the account of Teaching NLP Workshop "How long have folks been thinking about #TeachingNLP? Here's a paper from more than 50 years ago by Susumu Kuno and Anthony G. Oettinger (CACM 1968)." [20]. Articles with Twitter mentions were 1391 (59.5% of all articles with Altmetrics data, 15.3% of all articles in the post-Altmetrics era and 8.3% of all the articles in the dataset). The average number of Twitter mentions for any article in the whole dataset (16,838 articles) was 0.39, while the average number of tweets for the articles was 4.81 (SD = 12.75, Median = 2, range: [1, 313]). The average age of articles that have received Twitter mentions was 5.8 years, compared to 15.8 in the non-mentioned articles. Such difference was statistically significant, and large (difference = 9.97, 95% CI [9.61, 10.33], $t(2915.89) = 54.74$, $p < .001$; Cohen's d = 2.03, 95% CI [1.94, 2.12]). Similarly, the tweeted articles received an average of 2.9 citations/article/year compared to 0.7 in the non-tweeted articles, the difference was statistically significant, and large (difference = −2.19, 95% CI [−2.45, −1.93], $t(1410.09) = -16.56$, $p < .001$; Cohen's d = −0.88, 95% CI [−1.12, −0.77]). Articles with Twitter mentions were more likely to have more Mendeley readers (mean = 48.57) compared to 35.22 for the articles with no Twitter

**Table 2** Correlation between Twitter mentions and other social media

| Source | *r* | p |
|---|---|---|
| Number of Mendeley readers | 0.163 | <0.001 |
| Policy mentions | 0.106 | <0.001 |
| Q&A mentions | 0.052 | 0.055 |
| Wikipedia mentions | 0.069 | 0.010 |
| Patent mentions | −0.048 | 0.071 |
| Facebook mentions | 0.099 | <0.001 |
| Total citations | 0.033 | 0.216 |
| Total citations per year | 0.167 | <0.001 |
| Age of publication | −0.252 | <0.001 |

coverage. This difference was statistically significant and the effect size was small $t(1779.40) = -3.35$, $p < .001$; Cohen's $d = -0.16$, 95% CI [−0.25, −0.07]).

The correlation between the number of Twitter mentions and total citations was statistically insignificant, while the correlation between Twitter mentions and number of citations per article per year was weak $r = 0.17$, $p < 0.001$. Furthermore, the correlation between Twitter mentions and number of Mendeley readers, policy mentions, Wikipedia mentions, patent mentions, and Facebook mentions were either trivial or statistically insignificant.

In summary, articles on Twitter tended to be more recent, with slightly more citations per article per year, as well as more Mendeley readers. It is important here to emphasize that we make no assumptions of any causal relationship, i.e., we do not imply that Twitter mentions increased the citations or readership. In fact, it is possible that the mechanism that made the article receive more citations (e.g., interesting, or novel findings) caused both Twitter mentions and citations. In all cases, such differences were very small (Table 2).

The most mentioned topic on Twitter was *computational thinking* which garnered 2384 mentions (9.7% of all Twitter mentions), followed by *computational theory* (1943 mentions, 8%), *programming* (1861, 7.6%), *introductory courses* (1443, 5.9%) and *pedagogy* (1416, 5.7%) and *education psychology (*1347, 5.5%). The order of the most mentioned topics and the timeline of tweets per year of publication is shown in Fig. 2. We see that topic of pedagogy, assessment, and introductory courses as well as games were early mentioned on Twitter. As the graph shows, there is no certain pattern that we can discern from the graph, and the timeline looks rather irregular. We also see that the year 2022 had witnessed a large increase in Twitter mentions for the first five topics (*computational thinking, computational theory*, *introductory courses, pedagogy, and education psychology).*

The top articles mentioned on Twitter in Table 3 come from different themes, e.g., ethics, programming education, introductory courses as well as computational thinking and inclusion. The top cited article in the list discusses the state of ethics education in computer science education. The authors claim that the field has an "ethics crisis" that needs to be addressed to avoid what they call "exclusionary pedagogy" where there is lack of interdisciplinarity and collaboration with other fields to improve the ethics curricula [21]. Five other papers addressed programming


**Fig. 2** Evolution of Twitter mentions by topic. (Circle size reflect the number of papers, color intensity reflects number of citations)

education discussing diverse topics. McGowan et al. [22] reported a positive correlation between seating in the front row during programming classes and performance. Stefik and Siebert [23] investigated the intuitiveness of the syntax of different programming languages. Drake and Sung [24] used board games to introduce computer science topics to university students. Salac and Franklin [25] found a weak correlation between performance and quality indicators calculated from school children's Scratch artifacts. In the same token, Chen et al. [26] found a positive correlation between prior programming experience and attitudes towards programming as well as academic achievement, and concluded that it is more effective to teach young students using a graphical language than a text-based one. Two articles among the top mentioned articles discussed political aspects of computer science education [27, 28]. The last article in our list [29] discusses female participation and attainment in CS; where the findings indicate that females score

**Table 3** Top mentioned articles in Twitter

| Title | Authors | Year | Twitter mentions | Citations |
|---|---|---|---|---|
| "You can't sit with us": Exclusionary pedagogy in AI ethics education | Raji, Scheuerman and Amironesei | 2021 | 313 | 3 |
| Learning to program – choose your lecture seat carefully! | McGowan, Hanna, Greer and Busch | 2017 | 139 | 3 |
| An empirical investigation into programming language syntax | Stefik and Siebert | 2013 | 123 | 139 |
| Teaching introductory programming with popular board games | Drake and Sung | 2011 | 115 | 20 |
| The organization and content of informatics doctoral dissertations | Shortliffe | 2016 | 104 | 1 |
| Infrastructures of abstraction: How computer science education produces anti-political subjects | Malazita and Resetar | 2019 | 103 | 4 |
| Political computational thinking: Policy networks, digital governance and 'learning to code' | Williamson | 2016 | 67 | 31 |
| If they build it, will they understand it? Exploring the relationship between student code and performance | Salac and Franklin | 2020 | 67 | 4 |
| The effects of first programming language on college students' computing attitude and achievement: A comparison of graphical and textual languages | Chen, Haduong, Brennan, Sonnert and Sadler | 2019 | 65 | 23 |
| Female performance and participation in computer science: A national picture | Kemp, Wong and Berry | 2019 | 61 | 4 |

higher than their male peers but lower than their average score in other courses. The article also argues that the introduction of CS into the national curriculum might "decrease the number of girls choosing further computing qualifications or pursuing computing as a career".

It is worth noting that six of the top Twitter mentioned articles have received less than 5 citations, emphasizing the discord between Twitter publicity and academic interest (as measured by citation count). Nonetheless, it is not difficult to discern where there has been a conversation about these articles on Twitter. For instance, two articles' titles have chosen thought provoking titles "You can't sit with us" and "choose your lecture seat carefully". One article addresses the programming language war, and two articles address policy and politics, and an article cautions against introducing CS into female education. Lastly, the remaining of these articles

addresses graduate students' dissertations [30] which would be expected to be shared among doctoral students who are social media users.

A total of 3044 authors had at least a paper mentioned on Twitter: 75.6% of them had a single paper and around 97.2% of the authors had five papers or less. The median year of the first publication of the authors with Twitter mentions was 2016 (compared to 2011) which reflects the recency of the Altmetrics and Twitter. There was a weak correlation between the mean number of tweets an author has and the mean number of citations their article gets; Spearman's rank correlation was statistically significant and medium ($r = 0.21$, $p < 0.001$). Yet, while the correlation is weak, it should not be interpreted as causation.

The top authors with Twitter mentions were not the most cited or the most productive authors. Nonetheless, they were mostly among the top 50 authors. On top of the list was Brett A. Becker, an assistant professor at the School of Computer Science at University College Dublin who had 30 of his papers mentioned on Twitter, each receiving an average of four mentions. Aman Yadav, a professor of educational psychology & educational technology at Michigan State University, had 24 of his papers discussed on Twitter, with a total of 202 mentions and an average of 8.4 mentions per paper. Amy J. Ko, professor of informatics at University of Washington, Seattle, and the Editor-in-Chief of TOCE had 22 of her articles discussed on Twitter, with an average of 6.2 mentions per paper. Table 4 has the full list of authors with most discussed papers on Twitter.

**Table 4** Top mentioned authors on Twitter

| Author | Oldest | # of papers | Proportion | Rank | Mean mentions | Mean citations |
|---|---|---|---|---|---|---|
| Becker BA | 2016 | 30 | 0.682 | 36 | 4.1 | 17.2 |
| Yadav A | 2011 | 24 | 0.667 | 64 | 8.4 | 26.75 |
| Ko AI | 2009 | 22 | 0.611 | 61 | 6.2 | 22.36 |
| Petersen A | 2011 | 20 | 0.4 | 23 | 3.5 | 19.95 |
| Porter I | 2010 | 19 | 0.333 | 14 | 2.2 | 20 |
| Franklin D | 2011 | 18 | 0.409 | 38 | 9.9 | 11.67 |
| Cutts Q | 2007 | 16 | 0.432 | 55 | 11.6 | 9.62 |
| Hellas A | 2016 | 16 | 0.356 | 35 | 4.7 | 10.19 |
| Sentance S | 2011 | 16 | 0.364 | 39 | 11.6 | 14 |
| Falkner K | 2009 | 15 | 0.375 | 47 | 5.4 | 13.93 |
| Guzdial M | 1994 | 15 | 0.172 | 1 | 1.5 | 23.6 |
| Simon | 1997 | 15 | 0.174 | 2 | 3 | 23.4 |
| Luxton-Reilly A | 2005 | 14 | 0.2 | 6 | 3.4 | 27.36 |
| Kafai YB | 2008 | 13 | 0.325 | 48 | 3.3 | 23.23 |
| McGill MM | 2009 | 13 | 0.277 | 32 | 4.6 | 11 |

### 4.2 Mendeley

Some 2268 articles had Mendeley data, which is 97% of all articles with Altmetrics data and 13.46% of all the articles in the dataset. The mean number of mentions was 43.17 (SD = 91.64, Median = 21), and the mean age of publication (time since published) was 9.21 (SD = 8.37, Median = 7.00) which is older than the mean age on Twitter. The presence of Mendeley data and the number of readers per article are well known to correlate with the number of citations across several studies (e.g., [7]), which was the case in our study. Articles with Mendeley data were more likely to be cited with a mean of 19.51 citations compared to 6.00 in articles without, the difference was statistically significant and medium (difference = −13.51, 95% CI [−15.39, −11.63], t(2331.86) = −14.08, p < 0.001; Cohen's d = −0.58, 95% CI [−0.76, −0.50]). Within articles with Mendeley data, correlation between number of readers per article and citation count was statistically significant, positive and very large (r = 0.64, 95% CI [0.61, 0.66], t(2266) = 39.14, p < 0.001). There was also a weak correlation between the number of Mendeley readers and policy, news, blogs, Facebook, or Wikipedia mentions.

Regarding authors, the number of articles with Mendeley readers was correlated with citation count, which was statistically significant, and effect size was very large (r = 0.68, 95% CI [0.66, 0.69], t(4327) = 60.86, p < 0.001). Similarly, the number of Mendeley readers was highly correlated with citation counts which was statistically significant with a very large effect size (r = 0.78, 95% CI [0.77, 0.80], t(4327) = 83.35, p < 0.001). In summary, there is a very strong association between Mendeley mentions and citation counts for either papers or authors, which reflects the paper importance or impact rather than what has caused the citation.

The top CSE papers with the highest number of readers (Table 5) are expected to also reflect highly cited papers since we have established that correlation was high. Our top papers include seven papers that address issues related to computational thinking of CSE in schools. The other three papers address game-based learning, an instructional computer laboratory and computer curriculum. Most of the top read Mendeley papers are highly cited with six papers having over 100 citations. The top read paper about game-based learning [31] is also the top cited paper in our complete dataset; the fifth paper about bringing computational thinking to K-12 [32] is the second most cited paper, and the seventh top read paper about Scratch is the third most cited paper [33].

The list of authors with a high number of articles in Mendeley (Table 6) show interesting findings that are different from those of Twitter. Most of the authors in the list are among the top publishing authors in the general dataset. The top authors were also well-established authors who started their careers during the last century or in the early 2000s, the mean years in publishing about CSE was 19.93 [6, 32]. Each of the top authors had an average of 53.41 Mendeley readers [18.83, 118.72]; each of their papers received a mean number of citations per paper of 27.26 [11.47, 48.92]. Such numbers were not much different from other authors who are not on the top list.

**Table 5** Top read papers in Mendeley

| Title | Authors | Year | Mendeley readers | Citations |
|---|---|---|---|---|
| Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation | Papastergiou | 2009 | 1836 | 984 |
| Progress report: Brown University Instructional Computing Laboratory [34] | Brown and Sedgewick | 1984 | 1763 | 16 |
| Computational thinking [35] | Henderson, Cortina and Wing | 2007 | 1509 | 63 |
| Design patterns: An essential component of CS curricula [36] | Astrachan, Mitchener, Berry and Cox | 1998 | 1466 | 42 |
| Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? [32] | Barr and Stephenson | 2011 | 817 | 658 |
| Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test [37] | Román-González, Pérez-González and Jiménez-Fernández | 2017 | 786 | 215 |
| The scratch programming language and environment | Maloney, Resnick, Rusk, Silverman, Eastmond and Evelyn | 2010 | 727 | 640 |
| Computational thinking in elementary and secondary teacher education [38] | Yadav, Mayfield, Zhou, Hambrusch and Korb | 2014 | 613 | 191 |
| Computational thinking for all: Pedagogical approaches to embedding twenty-first century problem solving in K-12 classrooms | Yadav, Hong and Stephenson. | 2016 | 587 | 133 |
| Constructivism in computer science education [39] | Ben-Ari | 1998 | 584 | 140 |

## *4.3 News and Blogs*

CER has appeared rarely in the news where only 79 (0.46%) articles were mentioned across the whole dataset, with a total of 136 mentions in total. The most mentioned articles by the news (Table 7) seem to address diversity and gender issues, which made more than half of the news mentions. The top article in the list was discussed by, e.g., Scientific American, Los Angeles Times, Christian Science Monitor, Houston Chronicle and SF Gate. Christian Science Monitor presented the article and concluded "Children need to be engaged in STEM before they start to lose interest. The image of STEM as solitary and isolating is strong in our culture. If

**Table 6** Top read authors on Mendeley

| Author | Oldest | # of papers | Proportion | Rank | Mean mentions | Mean citations |
|---|---|---|---|---|---|---|
| Guzdial M | 1994 | 49 | 0.56 | 1 | 38.06 | 38.37 |
| Rodger SH | 1993 | 42 | 0.78 | 16 | 18.83 | 18.48 |
| Astrachan O | 1990 | 34 | 0.65 | 17 | 55.74 | 11.47 |
| Becker BA | 2016 | 33 | 0.75 | 36 | 37.36 | 17.12 |
| Simon | 1997 | 33 | 0.38 | 2 | 36.09 | 17.30 |
| Ben-Ari M | 1996 | 28 | 0.57 | 26 | 76.36 | 38.18 |
| Ko AI | 2009 | 27 | 0.75 | 61 | 55.04 | 23.00 |
| Lister R | 2000 | 25 | 0.37 | 9 | 53.88 | 48.92 |
| Sheard J | 1997 | 25 | 0.33 | 4 | 52.88 | 27.16 |
| Yadav A | 2011 | 25 | 0.69 | 64 | 118.72 | 30.48 |
| Edwards S | 1998 | 24 | 0.35 | 8 | 47.71 | 36.04 |
| Porter I | 2010 | 24 | 0.42 | 14 | 32.08 | 20.50 |
| Luxton-Reilly A | 2005 | 23 | 0.33 | 6 | 55.04 | 29.65 |
| Petersen A | 2011 | 23 | 0.46 | 23 | 51.22 | 20.91 |
| Armoni M | 2004 | 22 | 0.49 | 34 | 72.14 | 31.36 |

we make STEM social, we can help inspire more students to discover their interest in STEM" [40]. The second article with a significant number of news mentions has also discussed gender diversity and was mentioned by Business Insider, World Economic Forum, and The National Interest. For instance, Business Insider titled their story "Women are just as capable as men in computing skills—but they're not as confident. Here's how that's contributing to the gender gap in tech" [41]. The authors of the paper concluded that "many have made the case that companies need better participation of women in the STEM workforce for greater innovation and productivity. These efforts have had some success, but other avenues are needed to promote STEM careers to women and help them to believe in their abilities." The World Economic Forum presented a similar story with the title "Computing has a gender problem – and isn't about talent."

Some 134 papers (0.8%) had blog mentions with 150 blog appearances in total. The highest blogged about paper (six times) was also the paper that received the top news mentions and addressed the stereotypes about girls interest in computer science [51]. The blog named "Scienceblog" published a blog post titled "To Get Girls More Interested In Computer Science, Make Classrooms Less 'Geeky'". All other blog mentions were two mentions or less and therefore, will not be discussed in detail here.

## *4.4 Patents*

A total of 131 (0.78%) articles received patent mentions and received a total of 434 mentions combined. Table 8 shows the articles with the most mentions.

**Table 7** Top mentioned papers in the news

| Title | Authors | Year | Citations | News mentions |
|---|---|---|---|---|
| Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science | Master, Cheryan and Meltzoff | 2016 | 173 | 16 |
| Fostering gender diversity in computing [42] | Prey and Weaver | 2013 | 13 | 16 |
| Multiple case study of nerd identity in a CS class [43] | Davis, Yuen and Berland | 2014 | 5 | 9 |
| They can't find us: The search for informal CS education [44] | DiSalvo, Reid and Roshan | 2014 | 22 | 9 |
| Gender neutrality improved completion rate for all [45] | Svedin and Bälter | 2016 | 1 | 6 |
| What is AI literacy? Competencies and design considerations [46] | Long and Magerko | 2020 | 51 | 6 |
| Computer science trends and trade-offs in California high schools [47] | Bruno and Lewis | 2021 | 1 | 6 |
| History of logo [48] | Solomon, Harvey, Kahn, Lieberman, miller, Minsky, Papert and Silverman | 2020 | 9 | 5 |
| A growth mind-set intervention improves interest but not academic performance in the field of computer science [49] | Burnette, Hoyt, Russell, Lawson, Dweck and Finkel | 2020 | 23 | 5 |
| Collaborative strategic board games as a site for distributed computational thinking [50] | Berland and Lee | 2011 | 99 | 4 |

The article that received almost one third of all patent mentions describes an online computerized testing system called "QUIZIT" which supports adaptive and standard testing, automatic grading, and storage of results [52]. The paper was mentioned by several patents across a wide range of applications that include systems and methods for automatic scheduling of a workforce, discovering customer center information, recording audio as well as by a web service for student information and course management systems. The next article on the list discusses the development of a programming project where Java applets can be dynamically updated in an undergraduate programming course [53]. The paper was mentioned by several patents (30) mostly covering access to database and software design. The remaining papers with patent mentions revolve around the same themes, i.e., either enhancement to an online teaching system or teaching programming.

**Table 8** Top papers mentioned by patents

| Title | Authors | Year | Citations | Patent mentions |
|---|---|---|---|---|
| Online evaluation in WWW-based courseware | Tinoco, Barnette and Fox | 1997 | 19 | 118 |
| Developing integrated web and database applications using Java Applets and JDBC drivers [53] | Yang, Linn and Quadrato | 1998 | 5 | 30 |
| A reusable graphical user interface for manipulating object-oriented databases using Java and XML [54] | Dietrich, Suceava, Cherukuri and Urban | 2001 | 6 | 11 |
| A constructivist approach to object-oriented design and programming [55] | Hadjerrouit | 1999 | 30 | 11 |
| The KScalar simulator [56] | Moure, Rexachs and Luque | 2002 | 9 | 11 |
| Interactive hypermedia courseware for the World Wide Web [57] | Marshall and Hurley | 1996 | 6 | 9 |
| On-line programming examinations using WebToTeach [58] | Arnow and Barshay | 1999 | 9 | 9 |
| Teaching web development technologies in CS/IS curricula [59] | Lim | 1998 | 8 | 8 |
| Using a model railroad to teach digital process control [60] | McCormick | 1988 | 9 | 8 |
| Using Java to teach networking concepts with a programmable network sniffer [61] | Jipping, Bugaj, Mihalkova, Porter and Donald | 2003 | 10 | 8 |

### *4.5 Other Altmetrics Sources*

Other services had very few mentions. Only two articles had six mentions by Wikipedia [62, 63], where the first discussed computer science education in French Universities and the second discussed visual simulation. Facebook mentions were also very scarce: the highest mentioned article received only four mentions and discussed computational thinking [64]. On the questions and answers website Stack Exchange, the mentions were even fewer, with only a single article mentioned two times [65]. The article was mentioned as a reply to the question "Which math classes should be included in an undergraduate computer science program?"

## 5 Discussion and Conclusions

Altmetrics' vision was to establish an alternative way of evaluating science, capturing the conversation across social media and the web at large, and presenting an immediate record of the attention a scholarly work gets. We offer a discussion of this vision in light of the results we had and the review of CER within Altmetrics.

The results of this study showed that correlation between citation counts and the social media mention of articles, or other sources of mentions e.g., news, blogs or Wikipedia was weak in the former or negligible in the latter. The case was also true for the authors where a direct correlation was not possible to establish. Therefore, social media cannot be viewed as a measure of scientific impact in the traditional scholarly way. These conclusions are further supported by the finding that articles that received the most mentions on Twitter were not highly cited, also, authors with the most mentions on Twitter were not the highest cited. While some differences exist between articles or authors who received mentions and those that do not, the differences were small. Furthermore, a causal relationship between mentions and citation count is impossible to establish given the weak associations and the fact that the mechanism that led an article to receive Altmetrics mentions (i.e., important findings) could be the same reason for receiving citations. These findings might be disappointing to researchers who may resort to social media to increase attention, attract readership, or disseminate their research. Nonetheless, the results are indeed a reflection of a different impact that is not essentially correlated to citations but possibly complementary to the traditional scientometrics and leads to engagement of a relatively different audience.

The case was different for Mendeley, where the correlation between the number of Mendeley readers and citation counts was strong for papers and authors. Yet, Mendeley is not a social media platform—despite the fact that it was designed to be so—as it currently stands. Mendeley's focus is to offer citation management and archival of papers, rather than any true social dialogue about the scholarly work. Therefore, Mendeley numbers should be viewed as predictors of paper impact on the scholarly work in the same way citation counts are, rather than being "alternative metrics".

The second vision that Altmetrics sought was to capture the web dialogue about scholarly work. Altmetrics has proven to collect a wide array of web sources. Given the small numbers on most platforms including social media, it seems that CER has not yet endorsed social media to its full potential. The social media uptake within the CER community has just started to take off in the last year. Yet, it is too early to conclude that the uptake will continue to grow. Furthermore, it is early to reach firm conclusions given the small numbers. Despite the fact that the speed of Altmetrics curation has proven to capture the immediate reactions and mentions of scholarly work as they are published online, they have not been used to their full potential within the CER community.

The analysis has shown that the chatter about CER on social media is more focused on topics that are essentially of public interest rather than the academic or pedagogical priorities. Topics that received more mentions on Twitter were computational thinking and programming in schools, ethics, diversity and gender issues (e.g., [21, 25, 29]). Topics of news' interest were particularly focused on gender issues, inclusivity and how to narrow the gender gap. Papers that addressed stereotypes about girls and computer science were more likely to resonate across the news and blogsphere (e.g., [51]). In the case of Mendeley, the topics of interest tended to be of scholarly interest and therefore, we can say that Mendeley offers

just another way to measure—or possibly predict the future of—academic impact and interest. Patents were rather few and, thus, solid conclusions cannot be drawn, although there seems to be a trend towards learning tools or educational technology.

In summary, Altmetrics reflect a different chatter, a dialogue that may be dominated by academics but not about academics, or their interests but rather geared to public concerns at large. Thereupon, Altmetrics can be viewed as a melting pot of dialogue that could help two seemingly distant communities—academics and the public—recognize each other's aspirations and perhaps awes too. Reading the Altmetrics with the lens of citation counts is simplistic, reductionist and defeats the purpose as Alternative metrics.

## References

1. Priem J, Taraborelli D, Groth P, Neylon C (2011) Altmetrics: A manifesto
2. Priem J, Groth P, Taraborelli D (2012) The altmetrics collection. PLoS One 7:e48753
3. Piwowar H (2013) Introduction altmetrics: What, why and where? Bull Am Soc Inf Sci 39:8–9
4. Ortega J-L (2020) Altmetrics data providers: A metaanalysis review of the coverage of metrics and publication. El profesional de la información (EPI) 29:
5. Nuzzolese AG, Ciancarini P, Gangemi A, et al (2019) Do altmetrics work for assessing research quality? Scientometrics 118:539–562
6. Hassan S-U, Imran M, Gillani U, et al (2017) Measuring social media activity of scientific literature: an exhaustive comparison of scopus and novel altmetrics big data. Scientometrics 113:1037–1057
7. Erdt M, Nagarajan A, Sin S-CJ, Theng Y-L (2016) Altmetrics: an analysis of the state-of-the-art in measuring research impact on social media. Scientometrics 109:1117–1166
8. Thelwall M (2020) The pros and cons of the use of altmetrics in research assessment. Scholarly Assessment Reports 2. https://doi.org/10.29024/sar.10
9. Hicks D, Wouters P, Waltman L, et al (2015) Bibliometrics: The Leiden Manifesto for research metrics. Nature 520:429–431
10. Schöbel S, Saqr M, Janson A (2021) Two decades of game concepts in digital learning environments – A bibliometric study and research agenda. Comput Educ 173:104296
11. Haustein S, Bowman TD, Holmberg K, et al (2016) Tweets as impact indicators: Examining the implications of automated "bot" accounts on Twitter. J Assoc Inf Sci Technol 67:232–238
12. Hassan S-U, Aljohani NR, Idrees N, et al (2020) Predicting literature's early impact with sentiment analysis in Twitter. Knowledge-Based Systems 192:105383
13. Kwak H, Lee C, Park H, Moon S (2010) What is Twitter, a social network or a news media? In: Proceedings of the 19th international conference on World wide web - WWW '10. ACM Press, New York, New York, USA
14. Sugimoto CR, Work S, Larivière V, Haustein S (2017) Scholarly use of social media and altmetrics: A review of the literature. J Assoc Inf Sci Technol 68:2037–2062
15. Elsevier (2013) Victor Henning's brief guide to Mendeley. In: Elsevier Connect. https://www.elsevier.com/connect/archive/victor-hennings-brief-guide-to-mendeley. Accessed 2 May 2022
16. López-Pernas, Saqr M, Apiola M (2023) Scientometrics: a concise introduction and a detailed methodology for the mapping of the scientific field of computing education. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer, p in–press
17. R Core Team (2018) R: A Language and Environment for Statistical Computing

18. Apiola M, López-Pernas S, Saqr M (2023) The evolving themes of computing education research: Trends, topic models, and emerging research. In: Apiola M, López-Pernas S, Saqr M (eds) Past, Present and Future of Computing Education Research. Springer
19. Kuno S, Oettinger AG (1968) Computational linguistics in a Ph.D. computer science program. Commun ACM 11:831–836
20. Teaching NLP Workshop @NAACL2021 (2021) How long have folks been thinking about #TeachingNLP? Here's a paper from more than 50 years ago by Susumu Kuno and Anthony G. Oettinger (CACM 1968). Computational Linguistics in a Ph.D. Computer Science program. https://t.co/kyLfSz46uD. In: Twitter. https://twitter.com/TeachingNLP/status/1349884735011610628. Accessed 23 Apr 2022
21. Raji ID, Scheuerman MK, Amironesei R (2021) You Can't Sit With Us: Exclusionary Pedagogy in AI Ethics Education. In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. Association for Computing Machinery, New York, NY, USA, pp 515–525
22. McGowan A, Hanna P, Greer D, Busch J (2017) Learning to Program: Choose Your Lecture Seat Carefully! In: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education. Association for Computing Machinery, New York, NY, USA, pp 4–9
23. Stefik A, Siebert S (2013) An Empirical Investigation into Programming Language Syntax. ACM Trans Comput Educ 13:1–40
24. Drake P, Sung K (2011) Teaching introductory programming with popular board games. In: Proceedings of the 42nd ACM technical symposium on Computer science education. Association for Computing Machinery, New York, NY, USA, pp 619–624
25. Salac J, Franklin D (2020) If They Build It, Will They Understand It? Exploring the Relationship between Student Code and Performance. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education. Association for Computing Machinery, New York, NY, USA, pp 473–479
26. Chen, Chen, Haduong P, et al (2019) The effects of first programming language on college students' computing attitude and achievement: a comparison of graphical and textual languages. Computer Science Education 29:23–48
27. Williamson B (2016) Political computational thinking: policy networks, digital governance and 'learning to code.' Critical Policy Studies 10:39–58
28. Malazita JW, Resetar K (2019) Infrastructures of abstraction: how computer science education produces anti-political subjects. Digital Creativity 30:300–312
29. Kemp PEJ, Wong B, Berry MG (2020) Female performance and participation in computer science. ACM trans comput educ 20:1–28
30. Shortliffe EH (2016) The organization and content of informatics doctoral dissertations. J Am Med Inform Assoc 23:840–843
31. Papastergiou M (2009) Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation. Comput Educ 52:1–12
32. Barr V, Stephenson C (2011) Bringing computational thinking to K-12. ACM Inroads 2:48–54
33. Maloney J, Resnick M, Rusk N, et al (2010) The Scratch programming language and environment. ACM trans comput educ 10:1–15
34. Brown MH, Sedgewick R (1984) Progress report. SIGCSE bull 16:91–101
35. Henderson PB, Cortina TJ, Wing JM (2007) Computational thinking. SIGCSE bull 39:195–196
36. Astrachan O, Mitchener G, Berry G, Cox L (1998) Design patterns. In: Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education – SIGCSE '98. ACM Press, New York, New York, USA
37. Román-González M, Pérez-González J-C, Jiménez-Fernández C (2017) Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Comput Human Behav 72:678–691

38. Yadav A, Mayfield C, Zhou N, et al (2014) Computational thinking in elementary and secondary teacher education. ACM trans comput educ 14:1–16
39. Ben-Ari M (1998) Constructivism in computer science education. SIGCSE bull 30:257–261
40. Master A (2016) Is making STEM social one way to get more children interested? In: The Christian Science Monitor. https://www.csmonitor.com/World/Making-a-difference/Change-Agent/2016/1007/Is-making-STEM-social-one-way-to-get-more-children-interested. Accessed 3 May 2022
41. The Conversation (2021) Women are just as capable as men in computing skills — but they're not as confident. Here's how that's contributing to the gender gap in tech. In: Insider. https://www.businessinsider.com/lack-of-confidence-among-women-gender-gap-in-stem-tech-2020-10?r=UK&IR=T. Accessed 3 May 2022
42. Prey JC, Weaver AC (2013) Fostering gender diversity in computing. Computer (Long Beach Calif) 46:22–23
43. Davis D, Yuen T, Berland M (2014) Multiple case study of nerd identity in a CS1 class. In: Proceedings of the 45th ACM technical symposium on Computer science education. ACM, New York, NY, USA
44. DiSalvo B, Reid C, Roshan PK (2014) They can't find us. In: Proceedings of the 45th ACM technical symposium on Computer science education - SIGCSE '14. ACM Press, New York, New York, USA
45. Svedin M, Bälter O (2016) Gender neutrality improved completion rate for all. Comput Sci Educ 26:192–207
46. Long D, Magerko B (2020) What is AI Literacy? Competencies and Design Considerations. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. ACM, New York, NY, USA
47. Bruno P, Lewis CM (2021) Computer science trends and trade-offs in California high schools. Educ Adm Q 0013161X2110548
48. Solomon C, Harvey B, Kahn K, et al (2020) History of Logo. Proc ACM Program Lang 4:1–66
49. Burnette JL, Hoyt CL, Russell VM, et al (2020) A growth mind-set intervention improves interest but not academic performance in the field of computer science. Soc Psychol Personal Sci 11:107–116
50. Berland M, Lee VR (2011) Collaborative strategic board games as a site for distributed computational thinking. Int j game-based learn 1:65–81
51. Master A, Cheryan S, Meltzoff AN (2016) Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. J Educ Psychol 108:424–437
52. Tinoco LC, Barnette ND, Fox EA (1997) Online evaluation in WWW-based courseware. SIGCSE bull 29:194–198
53. Yang A, Linn J, Quadrato D (1998) Developing integrated Web and database applications using JAVA applets and JDBC drivers. SIGCSE bull 30:302–306
54. Dietrich SW, Suceava D, Cherukuri C, Urban SD (2001) A reusable graphical user interface for manipulating object-oriented databases using Java and XML. SIGCSE bull 33:362–366
55. Hadjerrouit S (1999) A constructivist approach to object-oriented design and programming. SIGCSE bull 31:171–174
56. Moure JC, Rexachs DI, Luque E (2002) The KScalar simulator. ACM J Educ Resour Comput 2:73–116
57. Marshall AD, Hurley S (1996) Interactive hypermedia courseware for the World Wide Web. SIGCSE bull 28:1–5
58. Arnow D, Barshay O (1999) On-line programming examinations using Web to teach. SIGCSE bull 31:21–24
59. Lim BBL (1998) Teaching Web development technologies in CS/IS curricula. SIGCSE bull 30:107–111
60. McCormick JW (1988) Using a model railroad to teach digital process control. SIGCSE bull 20:304–308

61. Jipping MJ, Bugaj A, Mihalkova L, Porter DE (2003) Using Java to teach networking concepts with a programmable network sniffer. SIGCSE bull 35:120–124
62. Mounier-Kuhn P (2012) Computer science in french universities: Early entrants and latecomers. Inf cult 47:414–456
63. Osborne H, Yurcik W (2003) The educational range of visual simulations of the Little Man Computer architecture paradigm. In: 32nd Annual Frontiers in Education. IEEE
64. Yadav A, Hong H, Stephenson C (2016) Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. TechTrends 60:565–568
65. Liberal Arts Computer Science Conso (2007) A 2007 model curriculum for a liberal arts degree in computer science. ACM J Educ Resour Comput 7:2

# A Scientometric Perspective on the Evolution of the SIGCSE Technical Symposium: 1970–2021


**Sonsoles López-Pernas, Mikko Apiola, Mohammed Saqr, Arnold Pears, and Matti Tedre**

## 1 Introduction

ACM's SIGCSE is one of the first organizations focused on computing education [21]. In 1970, SIGCSE launched its Technical Symposium, which was initially focused as a forum for teachers of computing to share best practice with each other, exchange opinions, experiences, and course descriptions [21, 47]. Initial discussion topics, e.g., on programming paradigms and software engineering were enhanced by the presentation of teaching tools and educational technology towards the end of the first decade of SIGCSE [47]. Over the years, computing education research (CER) has significantly matured as an academic discipline, experience reports have been joined with more rigorous research, and more attention has been put to methodology, empirical evidence, and theory use [30]. In SIGCSE Technical Symposium, the scale of submissions has evolved from 18 accepted papers in the first year [50] to 161 accepted papers in 2018 and 297 papers in 2021. In recent years, SIGCSE Technical Symposium has attracted over 1000 attendees annually [21], and its publication profile and community have shifted remarkably from its inception. Out of all the central publication outlets that publish CER, SIGCSE Technical Symposium has, by far, published the largest share of articles [3], making it one of the most influential publication outlets of CER.

S. López-Pernas (✉) · M. Apiola · M. Saqr · M. Tedre
University of Eastern Finland, Joensuu, Finland
e-mail: sonsoles.lopez@uef.fi; mikko.apiola@uef.fi; mohammed.saqr@uef.fi; matti.tedre@acm.org

A. Pears
KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: pears@kth.se



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_10

As the SIGCSE Technical Symposium has expanded its publication profile and diversified its community, it is relevant to investigate that scholarly community and the community's publication and citation practices. In this chapter, we present a scientometric angle to the development of SIGCSE Technical Symposium in order to explore the collaboration networks, shifts in research focus, and citation practices. Scientometrics provides the possibility to go beyond simple counts to offer more mature and nuanced overviews of the temporal evolution of science. In this paper, state-of-the-art scientometric methods are used to offer an in-depth perspective on the evolution of the SIGCSE Technical Symposium. Our three research questions are:

1. How have authors and author networks shaped SIGCSE Technical Symposium and its community over time?
2. How has the publication profile of SIGCSE Technical Symposium evolved in terms of most-cited papers, keyword trends and keyword clusters?
3. How has SIGCSE Technical Symposium evolved from the viewpoint of international collaboration?

## 2 The Birth of SIGCSE

The need for large scale computing education efforts was born with the mass-production of fully-electronic, programmable computers just before the mid-1950s [47]. Around that time, in June 1954, *the First Conference on Training Personnel for the Computing Machine Field* was convened at Wayne University in Michigan, bringing together more than 150 interested people [26]. The early computing workforce was primarily trained by the burgeoning computer industry, but pressure was building up at universities to catch up with the computing education efforts [14]. At the turn of 1960s, already 150 universities offered some training in computing [15], and competition had started between major organizations in the computing field over their curriculum development efforts [47]. ACM started an education committee in 1960, but competitors were quick to establish their programs: In 1962, DPMA offered a professional examination in data processing and IFIP started panels on information processing [20, 47].

ACM formed a curriculum committee in 1962, but its first draft curriculum took 3 years and the final ACM Computing Curriculum (CC'68) took another three. However, once it was out, it quickly established an authoritative role in higher education: Just 4 years later, 78% of US computing education programs in universities reported that they considered ACM's curriculum valuable for their computing education efforts [51].

ACM's CC'68 had been a major undertaking involving many computing pioneers in a joint education effort, and the same year it was published, a number of pioneers signed a petition, at 1968 Fall Joint Computer Conference in Las Vegas, to establish ACM Special Interest Committee for Computer Science Education (SICCSE) [47]. Starting from 1969, the committee's newsletter, *SICCSE Bulletin*, was devoted

to computing education practice: "descriptions of new courses, novel approaches to established courses, problems and solutions, comments on the development of computer science education …" [22]. In 1970, the committee became an ACM special interest group (SIG), was rebranded as SIGCSE, and launched its first Technical Symposium a day prior to 1970 Fall Joint Computer Conference [1].

By 1970, SIGCSE membership had grown to over 600 members [32]. The early SIGCSE membership was primarily North American: in a member listing in December 1970, of SIGCSE Bulletin, out of 600+ SIGCSE members 80 were from outside the US, of whom 26 were from Canada. Accordingly, the early SIGCSE symposia gathered visitors primarily from North America (Fig. 3). For almost 30 years, reviewers were all North American (although it was shown that has no effect on review scores) [50]. Noticeable changes in the international profile of the symposium started to be seen in 1990s, which will be discussed more in Sect. 6 (Fig. 3). The same diversification was also noted by the SIGCSE Bulletin, which for the first time in 1997 received over 50% of its submissions from outside the US [34]. Two shifts can be discerned in the author composition of SIGCSE papers: roughly since the mid-1990s multiple-author papers have dominated over single-author papers, and international collaboration papers started to frequently appear in 2000s (Fig. 1). Papers have been overwhelmingly from the US, with more than 92% of attendees in recent years having an U.S. affiliation [42].

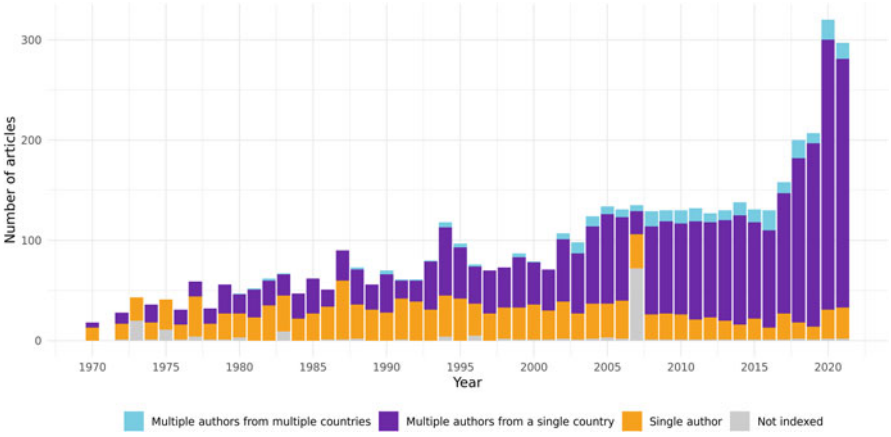


**Fig. 1** Number of papers presented and published in SIGCSE conferences, divided into papers with a single author, papers with multiple authors from a single country, and papers with multiple authors from multiple countries (authors affiliated with institutions from different countries). Gray color indicates unavailability of country metadata

### 2.1 Related Work

This is not the first paper to analyse the publications of the SIGCSE Technical Symposium. One of the earliest efforts to analyse the SIGCSE Technical Symposium is that of Valentine [49], and the classifications *Marco Polo, Tools, Experimental, Nifty, Philosophy and John Henry*. In Valentine's classification, *Marco Polo* ("I went there and I saw this") refers to an experience report, typically of trying a new curriculum or teaching method, *Tools* refers to research on educational tools, *Nifty* including research on innovative assignments, *John Henry*, describing papers that push the boundaries of pedagogy, *Experimental* referring to research with an experimental setup, and *Philosophy* debating issues on philosophical grounds [49]. Valentine's analysis sparked an interest in the idea of CER as a research field or discipline, and in turn led to a number of other efforts, both nearly immediately [38], and in the years that followed [16, 18, 43]. In addition there has been increasing focus on what the field is about, on how to understand what research has been done, and what to prioritize in the future [37, 47].

Some relevant finger-posts in the debate on CER, and what conferences should be about include a panel debate at ITiCSE 2004 [18], where three views of CER were presented and analysed, namely, *classroom practice reports, observing phenomena, and subject based learning research in collaboration with educational researchers*, with a fourth perspective, *reporting*, which focused on the need for research ethics and honest reporting, arguing that all the other three forms were valid contributions, but needed to be honestly reported.

Fincher's book on CS Education Research [16] also emerged the same year (2004), fueling an attempt to define a Core Literature for CER [38]. Since then there have been several attempts to develop research taxonomies [30, 43, 45] and analyse the research methods commonly employed [8, 12]. These endeavours provide the background to the current attempt to understand the evolution and significance of the SIGCSE Technical Symposium and its contributor communities. In particular the data presented in this chapter extends and contrasts with prior work on classification of publications and taxonomising the SIGCSE publication corpus [6, 49]. The main innovation in comparison with prior attempts is that our approach provides access to solid scientometrics and author network analysis, empirical data visualisations, and through these analyses a new longitudinal perspective on the development of the conference community, the shifts in themes and topics and significance of publications that appear in the conference proceedings.

## 3 Methodology

The metadata for all (1970–2021) SIGCSE Technical Symposium papers were retrieved from Scopus, as Scopus had the best quality metadata of SIGCSE

Technical Symposium papers.[1] In the early years, proceedings were published as a special issue of the ACM SIGCSE Bulletin. Articles in each special issue of the bulletin that were not part of the conference proceedings were manually removed, resulting in a total of 4982 articles.

Further cleaning was applied to fix database inconsistencies, which were manually fixed and verified against information on the ACM website. Author keywords were cleaned and equivalent keywords were combined: for instance, "CS1"/"CS 1", "K-12"/"K12", and "OSS"/"open source software" were grouped. The data were analyzed using the Bibliometrix R package [4]. Scopus metadata were used to construct a temporal timeline for the evolution of keyword use. A similar visualization was created for country contributions, to plot the trend of country participation over the years. The overall frequency of country contributions was further plotted on a world map.

The author analysis included frequency, earliest and latest contributions, as well as the number of total citations. A network of co-authorships was constructed using fractional counting, which has become the preferred method for co-authorship link weighting over traditional counting [39]. In traditional counting, each co-authored paper counts as one link between each pair of co-authors. In fractional counting, this link is weighted by the number of authors in the paper. Thus, the more co-authors a paper has, the weaker the link among them. Louvain community detection algorithm [13] was applied to highlight frequently collaborating authors and significant SIGCSE co-authorship communities.

For clarity, the analysis included only the top 100 authors, ranked by the number of distinct collaborators. Another network was constructed for author keywords based on keyword co-occurrence. Louvain community detection algorithm was applied to highlight important themes within SIGCSE publications. For more details on the analysis methodology, refer to chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" of this book [28].

## 4 Authors

Since the first SIGCSE Technical Symposium in 1970, 7349 unique author names have appeared in SIGCSE proceedings. Among the authors of SIGCSE papers, 5197 (70.8%) appeared just once, and 1040 (14.2%) twice, with a mean of 1.9 papers per author. For those authors who appeared more than once in SIGCSE, the mean

[1] Although the ACM Library also covers all the SIGCSE Technical Symposium proceedings, the Scopus metadata identify authors by their author IDs, allowing to disambiguate authors that use multiple names and affiliations and, thus, yielding more accurate results. Moreover, the citation count by Scopus is more representative of an article's impact since Scopus coverage is much larger than that of ACM library.

**Table 1** Twenty most productive authors in SIGCSE proceedings 1970–2021

| Author | First | Recent | Cites | Articles |
|---|---|---|---|---|
| D.D. Garcia | 2002 | 2021 | 195 | 47 |
| O. Astrachan | 1990 | 2019 | 401 | 39 |
| M. Guzdial | 1994 | 2020 | 725 | 37 |
| S.H. Rodger | 1993 | 2021 | 336 | 34 |
| J.C. Adams | 1993 | 2020 | 292 | 30 |
| H.M. Walker | 1990 | 2020 | 85 | 30 |
| K.E. Boyer | 2007 | 2021 | 232 | 28 |
| R.A. Mccauley | 1994 | 2019 | 329 | 28 |
| S.H. Edwards | 2004 | 2020 | 386 | 27 |
| L.N. Cassel | 1983 | 2021 | 154 | 26 |
| S. Cooper | 2003 | 2021 | 701 | 26 |
| B. Simon | 2004 | 2021 | 879 | 26 |
| D. Franklin | 2011 | 2021 | 327 | 24 |
| B.L. Kurtz | 1980 | 2014 | 154 | 24 |
| D. Baldwin | 1990 | 2018 | 152 | 23 |
| C.M. Lewis | 2010 | 2021 | 173 | 23 |
| N. Parlante | 1997 | 2021 | 48 | 23 |
| L. Porter | 2013 | 2021 | 363 | 23 |
| T. Barnes | 2006 | 2021 | 249 | 22 |
| O. Hazzan | 2001 | 2021 | 278 | 22 |

First = First year of appearance, Recent = Most recent appearance, Cites = Cites to the author's SIGCSE papers in Scopus, Articles = Number of papers

number of publications is 3.9. Several authors stand out for a large number of contributions to the SIGCSE Technical Symposium series. All authors on the list of 20 most productive authors (Table 1) have authored or co-authored 22 or more papers in SIGCSE. The top positions on the list of most productive authors features well known computing educators. Daniel D. Garcia from University of California, USA, was involved in 47 papers within a timespan from 2002 to 2021, with a total of 195 citations for his SIGCSE publications in the whole Scopus database, earning the top position on the list of most productive authors in SIGCSE, followed by Owen Astrachan with 39 papers and 401 cites in Scopus within a timespan from 1990 to 2019, and Mark Guzdial with 37 papers and 726 cites in Scopus, within a timespan from 1994 to 2020.

## *4.1 Collaboration*

Figure 2 presents co-authorships in the papers published in SIGCSE Technical Symposium. The nodes represent those authors who have most co-authors (more than five unique collaborators). The edges that connect the nodes represent co-

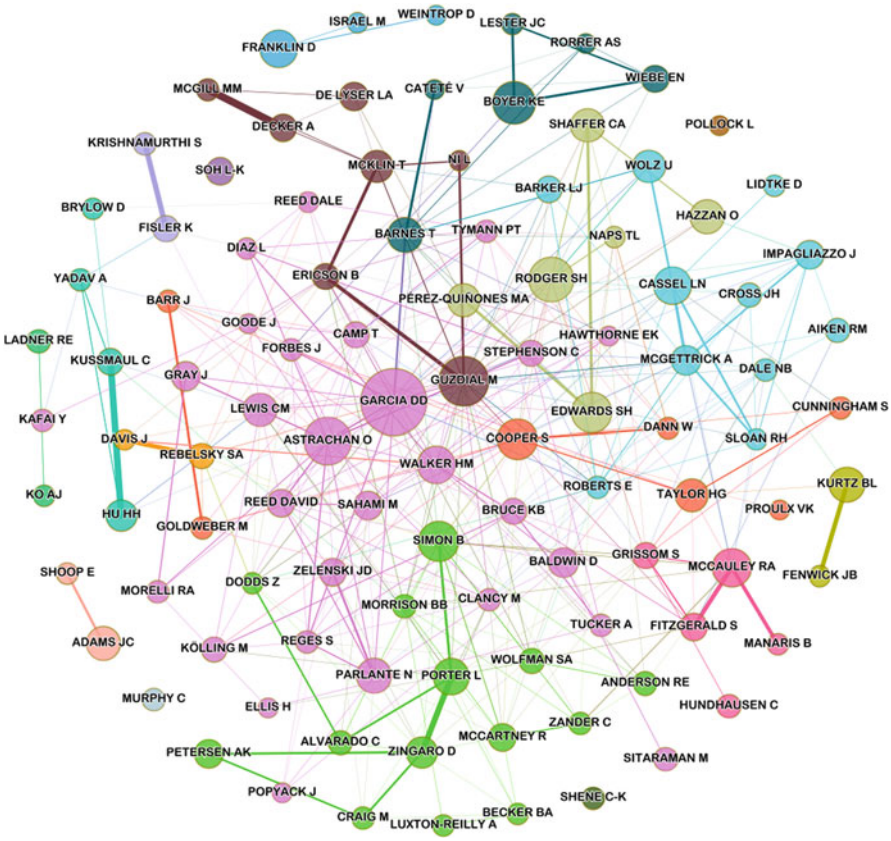


**Fig. 2** Co-author network of SIGCSE authors with most collaborators. Node size indicates the number of unique co-authors, edge thickness indicates number of co-authorships, and colors indicate communities of researchers who frequently collaborate together (using Louvain modularity algorithm)

authorships between the authors. Unconnected nodes are active collaborators, but whose co-authors do not belong to the group of most active collaborators in Fig. 2.

Some clearly identifiable clusters have formed around the authors of SIGCSE. The pink cluster at the center is formed around known SIGCSE contributors, editors, and award winners. These include Daniel D. Garcia (University of California), Owen Astrachan (Duke University, Durham, NC), and Henry M. Walker (Grinnell College, Grinnell, IA), many-times chair and bulletin editor in SIGCSE, as well as receiver of the Lifetime Service to Computer Science Education award. The pink cluster also features David Reed (Dickinson College), the leader of the first New Educators Workshop in SIGCSE, as well as Julie Zelenski and Nick Parlante (Stanford University, Stanford, CA, USA) and Michael Kölling (King's College, London, England), known for his work on programming education tools, and

receiver of the SIGCSE Outstanding Contribution to Computer Science Education Award in 2013, and SIGCSE Test of Time award in 2020.

The green cluster at six o'clock centers around Beth Simon (University of California at San Diego, USA; an active member of the SIGCSE community, chairing numerous editions of SIGCSE Technical Symposium), Leo Porter and Christine Alvarado (University of California San Diego), also active members of SIGCSE, having served, for instance, as editor of SIGCSE Bulletin among the years. The cluster also features Daniel Zingaro, Andrew Petersen, Michelle Craig (University of Toronto, Ontario, Canada), and CER researchers Briana B. Morrison (University of Nebraska Omaha, Omaha, NE, USA), and Steven A. Wolfman (University of British Columbia, Vancouver, BC, Canada), with connections to Brett A. Becker (University College Dublin, Belfield, Ireland) and Andrew Luxton-Reilly (University of Auckland, Auckland, New Zealand).

The brown cluster features Mark Guzdial (Georgia Tech, Atlanta, Georgia), a long-time member and Outstanding Contribution Award winner in 2019, and Barbara Ericson (Georgia Tech, Atlanta, Georgia), among others, while the Dark Green cluster at one o'clock features Kristy Elizabeth Boyer and Tiffany Barnes (North Carolina State University). The gray green cluster at two o'clock is centered around Susan Rodger (Duke University), a long-time member and chair in SIGCSE, and Stephen Edwards (Virginia Tech, Blacksburg, VA, USA), Outstanding Contribution to Computing Education Award winner in 2021 and prominent member of the SIGCSE community.

The dark pink cluster at five o'clock features long time active members and chairs in SIGCSE, Renée McCauley (College of Charleston), and Sue Fitzgerald (Metropolitan State University, St. Paul, MN, USA), while the light blue cluster at two o'clock is centered around Lillian Cassell (Villanova University, Philadelphia, PA, USA) and John Impagliazzo (Hofstra University, Hempstead, NY, USA) who is a 2007 lifetime service to computer science education award winner of SIGCSE. Other remarkable computing educators and SIGCSE community members are featured in other, smaller clusters shown in Fig. 2. The clusters in SIGCSE are dominated by US-centered networks with few connections to other countries, and feature members, chairs, award winners, and extraordinary computer science educators and researchers, some of which are also part of the most productive authors in SIGCSE (see Table 1).

Interestingly, some other highly active and awarded members of the SIGCSE community from outside the USA are not present, including Judy Sheard (Monash University, Australia) and Alison Clear (EIT, New Zealand), Lauri Malmi (Aalto University, Finland)—winner of the Outstanding Contribution to Computer Science Education Award—, Mats Daniels (Uppsala University, Sweden)—SIGCSE Outstanding Service Award and Lifetime Service to Computer Science Education Award winner—, and Raymond Lister (UTS, Australia).

## 5 Papers

Learning how to program is, of course, a topic of persisting interest for computing education research. The most cited paper in SIGCSE [31] is about using Scratch in K-12 computing education. With 273 citations in Scopus, the paper has become a popular reference in K-12 computing education. The second most cited paper [11] is about object-orientation in CS1 and has significantly contributed to the debates of programming languages and paradigm choices in CS1. The next two most cited papers are about pair-programming in CS1 [33], and success factors in CS [53]. Both of these papers [33, 53] have become popular references in CS education with 247 citations in Scopus each. Other top papers include a broader selection of papers on varying topics; automatic grading, gender differences, game-based approaches, learning styles, and aptitude in computer science.

Not surprisingly nearly half of the top papers (Table 2) focus on teaching programming or introductory computer science: [11] is about object-orientation in CS1, while [33] and [35] centered around pair programming in CS1, and the topic of [27] was a gamified approach in CS1. Further on, success factors in programming were investigated by Bergin and Reilly [7], while other topics that centered around CS1 were learning styles [48], modeling of learning [40], factors of persistence [5], and errors in Java programming in CS1 [23]. Other common topics included K-12, Scratch, and computational thinking (CT) [29, 31, 41, 52]. One paper in the top cited list deals with gamification, an investigation of Iosup and Epema [24]. A number of tools papers are also represented among the most highly cited SIGCSE Technical Symposium publications, including: detecting plagiarism [54], program visualisation in Python [19], and tools for grading [25]. Other highly cited topics included success factors in CS [53], choice of major [10], and gender differences [9].

The impact of a publication on the community can be considered from a range of perspectives. Pears et al. [38] proposed a classification system which incorporated both sustained cumulative citation and scholarly estimation of impact, proposing a system that divided papers into qualitative categories *seminal, influential* and *synthesis*. If we consider this perspective a majority of the papers in the list are relatively old (were published prior to 2010), and can be considered established and influential. Interestingly relatively few are authored by the productive and highly collaborative, "central", members of the community depicted in Fig. 2; the exception being Barker. One can wonder why publications by the central and most collaborative members of the SIGCSE Technical Symposium community are not more highly represented in the list of highly cited works.

There are four more recent papers (less than 10 years old) with high citations that might be considered both influential and seminal, as they seem to be popular works upon which to ground further efforts in the areas of *program visualisation, modelling learning of programming, computational thinking* and *gamification*. These are all areas of strong interest in the current CER research agenda internationally,

**Table 2** Twenty SIGCSE papers with most citations in scopus

| Title | Author(s) | Year | Cit. |
|---|---|---|---|
| Programming by Choice: Urban Youth Learning Programming with Scratch [31] | J. Maloney, K. Peppier, Y.B. Kafai, M. Resnick, N. Rusk | 2008 | 273 |
| Teaching Objects-First in Introductory Computer Science [11] | S. Cooper, W. Dann, R. Pausch | 2003 | 254 |
| The Effects of Pair-Programming on Performance in an Introductory Programming Course [33] | C. Mcdowell, L. Werner, H. Bullock, J. Fernald | 2002 | 247 |
| Contributing to Success in an Introductory Computer Science Course: A Study of Twelve Factors [53] | B.C. Wilson, S. Shrock | 2001 | 247 |
| Improving the Cs1 Experience with Pair Programming [35] | N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, S. Balik | 2003 | 227 |
| Why Students with an Apparent Aptitude for Computer Science Don't Choose to Major in CS [10] | L. Carter | 2006 | 215 |
| Yap3: Improved Detection of Similarities in Computer Program and Other Texts [54] | M.J. Wise | 1996 | 206 |
| Scratch for Budding Computer Scientists [29] | D.J. Malan, H.H. Leitner | 2007 | 189 |
| Online Python Tutor: Embeddable Web-Based Program Visualization for Cs Education [19] | P.J. Guo | 2013 | 188 |
| Grading Student Programs Using Assyst [25] | D. Jackson, M. Usher | 1997 | 174 |
| Gender Differences in Computer Science Students [9] | S. Beyer, K. Rynes, J. Perrault, K. Hay, S. Haller | 2003 | 173 |
| The Fairy Performance Assessment: Measuring Computational Thinking in Middle School [52] | L. Werner, J. Denner, S. Campe, D.C. Kawamoto | 2012 | 172 |
| Scalable Game Design and Development of a Checklist for Getting Computational Thinking ... [41] | A. Repenning, D. Webb, A. Ioannidou | 2010 | 143 |
| A Games First Approach to Teaching Introductory Programming [27] | S. Leutenegger, J. Edgington | 2007 | 143 |
| Programming: Factors That Influence Success [7] | S. Bergin, R. Reilly | 2005 | 141 |
| An Experience Report on Using Gamification in Technical Higher Education [24] | A. Iosup, D. Epema | 2014 | 136 |
| Learning Styles and Performance in the Introductory Programming Sequence [48] | L. Thomas, M. Ratcliffe, J. Woodbury, E. Jarman | 2002 | 125 |
| Exploring Factors That Influence Computer Science Introductory Course Students to Persist... [5] | L.J. Barker, C. Mcdowell, K. Kalahar | 2009 | 124 |
| Modeling How Students Learn to Program [40] | C. Piech, M. Sahami, D. Koller, S. Cooper, P. Blikstein | 2012 | 122 |
| Identifying and Correcting Java Programming Errors for Introductory Computer Science ... [23] | M. Hristova, A. Misra, M. Rutter, R. Mercuri | 2003 | 119 |

C/A = Country of the first author's affiliation, Cit. = Cites in Scopus

and can probably be rightfully claimed as some of the Symposium's more recent seminal works.

## 6 International Collaboration

SIGCSE Technical Symposium was launched in the United States, and has always been strongly US-oriented. Figure 3 shows frequencies of contributions per country in each year of SIGCSE, as determined by the first author affiliation. Prior to 1990, only few papers originated from outside of North America (US or Canada). The countries with most contributors are United States (11,003 author appearances), Canada (429 author appearances), United Kingdom (245 author appearances), and Australia (169 author appearances). The number of contributions from other countries has increased over the years, and nowadays an increasing number of papers in SIGCSE originate from European, South American, African, Oceanian, and Asian countries. But although the symposium's papers today originate from many corners of the globe (Fig. 4), contributors from Africa are rare, and Asia and South America are greatly underrepresented. It seems plausible to argue that continuity of publication in the Symposium is linked to strong SIGCSE chapters and activities. This hypothesis is supported by the observation that Finland, Sweden, Australia, New Zealand, UK and Israel demonstrate continuity in participation. The aforementioned countries also are characterised by having active CER groups and PhD programmes, as well as, in most cases, access to local SIGCSE conferences such as ITiCSE (mostly in Europe), ICER (UK, Australasia, US), CompEd (held outside of North America and Europe), as well as Koli Calling (Finland, Sweden) and ACE (Australasia/Oceania).

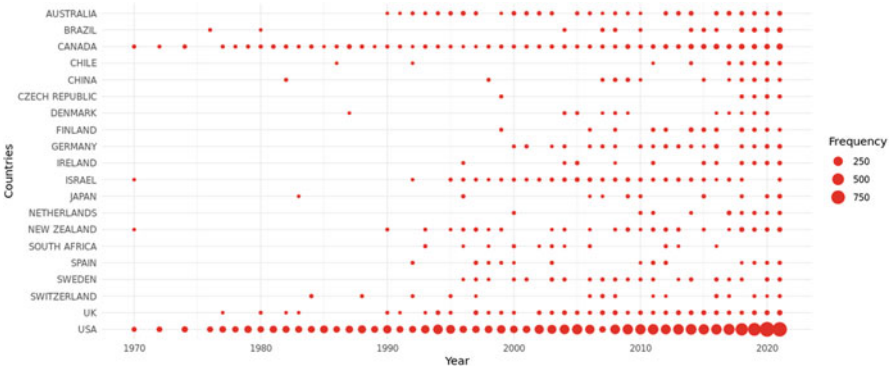


**Fig. 3** The 20 most active countries in SIGCSE proceedings by the number of papers published as determined by the affiliation of the first authors. The size of the circle indicates the number of first authors from a given country each year. In most early years, only the US and Canada exceed the threshold for visualization

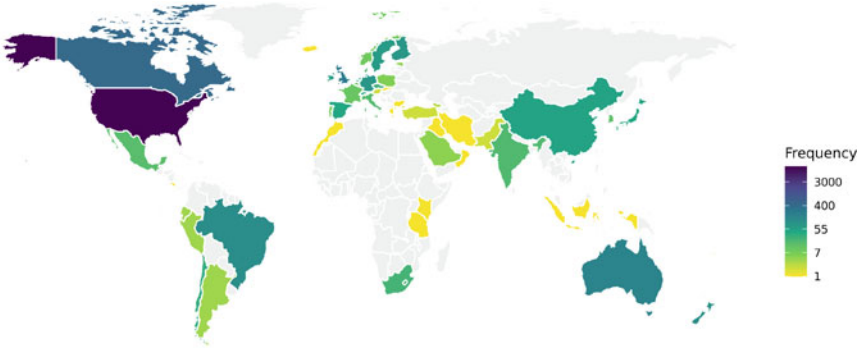


**Fig. 4** Distribution of SIGCSE papers across the globe

A typical paper in SIGCSE Technical Symposium does not seem to be amenable to international collaboration. Until 2000s, papers with authors from more than one country were few and far between (Fig. 1), and only some 5.4% of papers ever published in SIGCSE contain authors from multiple countries. A large portion (28.7%) of papers in SIGCSE have been authored by a single person, while out of all papers with multiple authors, some 92% were authored by authors from a single country, with only 8% including authors from multiple countries. Out of all papers, 6% could not be indexed with regards to author country. A minor increase in multi-country papers can be observed in the most recent two decades of SIGCSE Technical Symposium (Fig. 1).

## 7 Keywords and Themes

Keyword analysis reveals emergence, evolution, rise and fall of topics and trends during the history of SIGCSE conference. For almost 30 years in the beginning of SIGCSE, postal mail was the method of paper and review submission [50]. Electronic submission systems started to evolve from 2000, initiated by then Program Chair Henry Walker [50]. Prior to 2003, keywords were not used at all, or they were not used consistently [36]. Thus, analysis of keywords is only possible from 2003 onward. Figure 5 shows yearly occurrences for all 20 keywords that have appeared in top five keywords during one or more years of SIGCSE. Figure 5 shows popularity and emergence of keywords, such as the steady popularity of *CS1*, slight decline of *pedagogy*, and increase of *K-12*, *computational thinking*, and *gender and diversity*. The frequency of keywords *object-oriented programming* and *Java* clearly attenuates soon after 2015 and appear to be dying out.

This is reflected in the keywords' evolution, too, and while also the processing of papers was manual and done via postal mail prior to 2003 [50], the top keywords overall were not present in those years.

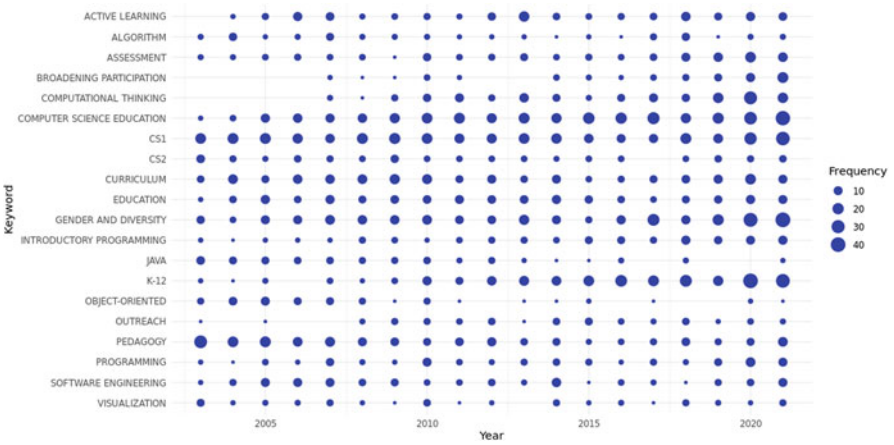


**Fig. 5** Changes in popular keywords (2003–2021)

An understanding of the interrelationships between CER topic areas can be obtained by conducting a network analysis of keyword clusters. The network in Fig. 6 shows clusters of keywords, linking together keywords that are commonly found together in the keywords block of the publication. The *light blue cluster* centers around computational thinking (CT) linking the area to contexts such as, K-12 education, using Scratch and robotics in computing education, with a large twist in gender diversity. The *orange cluster* centers around introductory programming (CS1, CS2) and preliminary programming course (CS0), showing a strong interest in the research community in related pedagogies, such as active learning and collaborative learning, and known debates around teaching CS1: object orientation, choices of programming language, assessment, and motivation. The *mint green cluster* centers around algorithms and data structures and their visualisation, while the *dark green cluster* centers around collaboration and pair-programming. The *pink cluster* shows that there are strong interests in connecting and exploring combinations of topics of relevance to software engineering, simulation, interdisciplinary education, curriculum, cybersecurity, and accessibility in computing education.

## 8 Discussion

Our first research question asked: *How have authors and author networks shaped SIGCSE Technical Symposium and its community over time?* Here our co-authorship analysis combined with our data on frequency of publication per country clearly demonstrate that the SIGCSE community remains extremely US centric with most collaborations being within a number of strong personal networks based around prominent researchers and their close colleagues and former students. Some isolated links to other networks can be detected and have been discussed earlier. Overall the

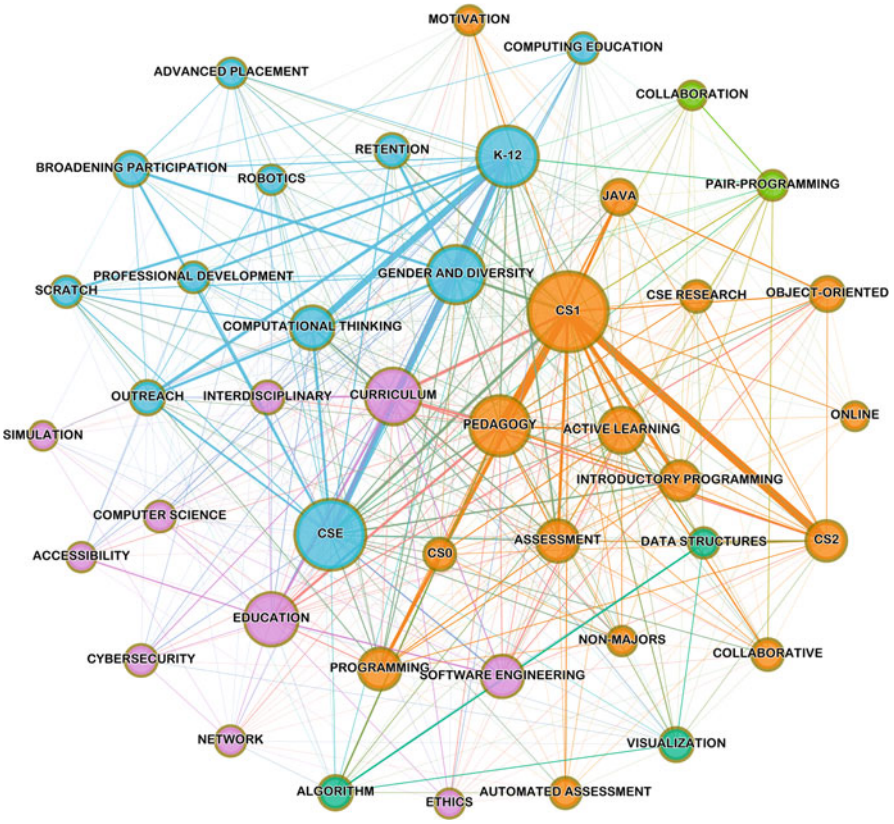


**Fig. 6** Keyword co-occurrence

SIGCSE Technical Symposium community is characterised by longstanding US-based collaborative groups. We also observe that some research communities, for instance Africa and Asia are particularly under-represented.

Our second research question asked: *How has the publication profile of SIGCSE Technical Symposium evolved in terms of most-cited papers, keyword trends and keyword clusters?* The analysis of keywords show that learning how to program is a topic of persisting interest in SIGCSE, which is well known [6]. Themes such as program visualisation, computational thinking and gamification are areas with strong interest within the CER community and can be considered as more recent seminal work in SIGCSE. Changes in topics post-2003 show increases in K-12 computing education, computational thinking and gender diversity, and dying out of object-oriented programming and Java. Clusters of keywords have centered e.g. around introductory programming and computational thinking and K-12.

A significant part of top cited papers in SIGCSE are about *introductory programming*. A 2004 meta-analysis of SIGCSE papers published between 1984 and

2003 showed that first year CS instruction, including CS1 and CS2, formed a significant portion of SIGCSE papers already in 1984–2003 [49], while recent research systematically categorised some 481 CS1 papers published during the first 50 years of SIGCSE from 1970 to 2018 with the following categories: teaching (105 papers), students (78 papers), first languages & paradigms (45 papers), tools (38 papers), CS1 content (38 papers), collaborative approaches (36 papers), CS1 design & structure (60 papers), learning & assessment (81 papers) [6]. Introductory programming is a top topic in SIGCSE, as it is in other publication forums of computing education, too [46].

Our final research question concerns *How has SIGCSE Technical Symposium evolved from the viewpoint of international collaboration?* Attendance to SIGCSE has greatly diversified when compared to the symposium's early decades, when the participants were predominantly from the US and Canada. Yet even in 2021, countries and whole continents are greatly underrepresented on the map of contributions to SIGCSE. For example, voices from Africa, a home to 1.2 billion people, are seldom heard, even when the virtual format would enable online presentations. In 1970, when the symposium started, the world looked different: a number of industrialized countries were far ahead of others in the computerization race and in computing education activities, many non-English speaking countries had their own forums of the SIGCSE kind, the academic world was much more segregated than it is in 2022, and publishing in national languages was more often the norm in earlier times. But in 2022, there is a need for ACM's prime (flagship) international computing education conference to discuss what the symposium could do to better serve the needs of computing education for all, in all countries, and not only in a few high-income countries.

The SIGCSE symposium was initially a forum for teachers of computing to share their best practices [21]. While research papers were also part of SIGCSE, the focus was initially on supporting practitioners [21]. Computing education research became more common post-2005 as the field started to evolve into its "mature era" [21]. The year 2005 also marks a turning point in computing education research where the discipline started to expand from mostly experience reports only to more rigorous computing education research [44].

## *8.1 Limitations*

This research has several limitations. First, while Elsevier's Scopus is accurate and maintained well [17], it is not flawless. The data has multiple issues with missing fields, inconsistencies in keywords, citation counts and references not being perfectly recorded. Manual checkups were needed in cases, but even with extensive cleaning, we can claim to have reached a representative, but not a comprehensive, sample of research. It is well known that citation counts differ between engines such as Scopus, Web of Science, Google Scholar, or ACM Digital Library. Comparing the reliability between the citation counts in different engines is beyond this work.

Lastly, another limitation of our work is that we used first authors' affiliation to analyze countries' productivity, to avoid overrepresenting countries in which there are many authors per paper. However, this approach fails to capture all countries' contributions.

## 9 Conclusion

We observe a continuing US-centric publication focus, and an urgent need to enable broader participation from the developing world. While a strong European participation starts to appear along with emerging Asian countries, e.g., China and India, there is almost near absence of half of the world. This is of course made worse by visa issues, subscription fees, and costs of travel that many scholars in the global south can not afford. In line with ACM's initiatives for increasing global participation—founding the Global Computing Education Conference (CompEd) in 2019, held outside of North America and Europe—, SIGCSE can do more to bridge the global divide of knowledge, e.g., invest more in travel support, issue fee waivers for certain countries, endorse young scholars from the global south, or even celebrating the conference outside of the US. In addition, our thematic analysis indicates an over-emphasis on programming and the learning of programming in the historical publication data. These are worrying trends, and should be addressed as the Technical Symposium proceeds into the next decade. Our analysis also shows the weak appearance of learning theories, a fluctuating trend for pedagogy as well as learning methods e.g., active learning. As computer science education grows, more alignment with learning theories would help improve our teaching and learning. In fact, we believe that SIGCSE may be an important venue for discussions of innovative pedagogies and theories that are germane to twenty-first century computing education. Novel educational trends like educational data mining and learning analytics [2, 3], have not made it to the top 20 keywords in SIGCSE, which raises questions about how the computer science education symposium aligns with novel trends that are pioneered by computer scientists. A positive trend that continues to be strong is gender and diversity which gives hope that our research can inform practice into a more equitable and diverse future. Our analysis has kept us wondering, has SIGCSE been a driver of innovation of computing education research? or just a mirror of the community interests?

## References

1. Aiken, R.M.: Editorial notes and observations. SICCSE Bulletin **1**(4), 2 (1969)
2. Apiola, M., López-Pernas, S., Saqr, M.: The evolving themes of computing education research: Trends, topic models, and emerging research. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research. Springer (2023)

3. Apiola, M., Saqr, M., López-Pernas, S., Tedre, M.: Computing education research compiled: Keyword trends, building blocks, creators, and dissemination. IEEE Access **10**, 27041–27068 (2022). DOI https://doi.org/10.1109/ACCESS.2022.3157609
4. Aria, M., Cuccurullo, C.: Bibliometrix: An R-tool for comprehensive science mapping analysis. Journal of Informetrics **11**(4), 959–975 (2017). DOI https://doi.org/10.1016/j.joi.2017.08.007
5. Barker, L.J., McDowell, C., Kalahar, K.: Exploring factors that influence computer science introductory course students to persist in the major. In: Proceedings of the 40th ACM Technical Symposium on Computer Science Education, SIGCSE '09, pp. 153–157. Association for Computing Machinery, New York, NY, USA (2009). URL https://doi.org/10.1145/1508865.1508923
6. Becker, B.A., Quille, K.: 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, pp. 338–344. Association for Computing Machinery, New York, NY, USA (2019). URL https://doi.org/10.1145/3287324.3287432
7. Bergin, S., Reilly, R.: Programming: Factors that influence success. In: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '05, pp. 411–415. Association for Computing Machinery, New York, NY, USA (2005). URL https://doi.org/10.1145/1047344.1047480
8. Berglund, A., Daniels, M., Pears, A.: Qualitative Research Projects in Computing Education Research: An Overview. Australian Computer Science Communications **28**(5), 25–34 (2006)
9. Beyer, S., Rynes, K., Perrault, J., Hay, K., Haller, S.: Gender differences in computer science students. In: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '03, pp. 49–53. Association for Computing Machinery, New York, NY, USA (2003). URL https://doi.org/10.1145/611892.611930
10. Carter, L.: Why students with an apparent aptitude for computer science don't choose to major in computer science. In: Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '06, pp. 27–31. Association for Computing Machinery, New York, NY, USA (2006). URL https://doi.org/10.1145/1121341.1121352
11. Cooper, S., Dann, W., Pausch, R.: Teaching objects-first in introductory computer science. In: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '03, pp. 191–195. Association for Computing Machinery, New York, NY, USA (2003). URL https://doi.org/10.1145/611892.611966
12. Daniels, M., Pears, A.: Models and methods for computing education research. Australian Computer Science Communications **34**(2), 95–102 (2012)
13. De Meo, P., Ferrara, E., Fiumara, G., Provetti, A.: Generalized louvain method for community detection in large networks. In: 2011 11th International Conference on Intelligent Systems Design and Applications, pp. 88–93 (2011). DOI https://doi.org/10.1109/ISDA.2011.6121636
14. Ensmenger, N.L.: The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise. The MIT Press, Cambridge, MA, USA (2010)
15. Fein, L.: The role of the university in computers, data processing, and related fields. Communications of the ACM **2**(9), 7–14 (1959)
16. Fincher, S., Petre, M.: Computer Science Education Research. Routledge Falmer (2004). URL http://www.cs.kent.ac.uk/pubs/2004/1819
17. Franceschini, F., Maisano, D., Mastrogiacomo, L.: Empirical analysis and classification of database errors in scopus and web of science. Journal of Informetrics **10**(4), 933–953 (2016). DOI https://doi.org/10.1016/j.joi.2016.07.003.
18. Goldweber, M., Clark, M., Fincher, S., Pears, A.: The relationship between CS education research and the SIGCSE community. In: ITiCSE '04: Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, pp. 228–229. ACM Press, Leeds, United Kingdom (2004). DOI http://doi.acm.org/10.1145/1007996.1008057
19. Guo, P.J.: Online python tutor: Embeddable web-based program visualization for cs education. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education,

SIGCSE '13, pp. 579–584. Association for Computing Machinery, New York, NY, USA (2013). URL https://doi.org/10.1145/2445196.2445368
20. Gupta, K., Sleezer, C.M., Russ-Eft, D.F.: A Practical Guide to Needs Assessment, 2nd edn. Pfeiffer Publishing, San Francisco, CA, USA (2007)
21. Guzdial, M., du Boulay, B.: The history of computing education research. In: S.A. Fincher, A.V. Robins (eds.) The Cambridge Handbook of Computing Education Research, pp. 11–39. Cambridge University Press, Cambridge (2019). DOI https://doi.org/10.1017/9781108654555.002.
22. Hildebrandt, T.W.: Editor's message. SICCSE Bulletin **1**(1), 1 (1969)
23. Hristova, M., Misra, A., Rutter, M., Mercuri, R.: Identifying and correcting java programming errors for introductory computer science students. In: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '03, pp. 153–156. Association for Computing Machinery, New York, NY, USA (2003). URL https://doi.org/10.1145/611892.611956
24. Iosup, A., Epema, D.: An experience report on using gamification in technical higher education. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education, SIGCSE '14, pp. 27–32. Association for Computing Machinery, New York, NY, USA (2014). URL https://doi.org/10.1145/2538862.2538899
25. Jackson, D., Usher, M.: Grading student programs using assyst. In: Proceedings of the Twenty-Eighth SIGCSE Technical Symposium on Computer Science Education, SIGCSE '97, pp. 335–339. Association for Computing Machinery, New York, NY, USA (1997). URL https://doi.org/10.1145/268084.268210
26. Jacobson, A.W. (ed.): Proceedings of the First Conference on Training Personnel for the Computing Machine Field. Wayne University Press, Detroit, MI, USA (1955)
27. Leutenegger, S., Edgington, J.: A games first approach to teaching introductory programming. In: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '07, pp. 115–118. Association for Computing Machinery, New York, NY, USA (2007). URL https://doi.org/10.1145/1227310.1227352
28. López-Pernas, S., Saqr, M., Apiola, M.: Scientometrics: a concise introduction and a detailed methodology for mapping the scientific field of computing education research. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research: A Global Perspective, pp. XX–XX. Springer (2023). https://doi.org/10.1007/978-3-031-25336-2
29. Malan, D.J., Leitner, H.H.: Scratch for budding computer scientists. In: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '07, pp. 223–227. Association for Computing Machinery, New York, NY, USA (2007). URL https://doi.org/10.1145/1227310.1227388
30. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Characterizing research in computing education: A preliminary analysis of the literature. In: Proceedings of the Sixth International Workshop on Computing Education Research, ICER '10, pp. 3–12. Association for Computing Machinery, New York, NY, USA (2010). URL https://doi.org/10.1145/1839594.1839597
31. Maloney, J.H., Peppler, K., Kafai, Y., Resnick, M., Rusk, N.: Programming by choice: Urban youth learning programming with Scratch. In: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '08, pp. 367–371. Association for Computing Machinery, New York, NY, USA (2008). URL https://doi.org/10.1145/1352135.1352260
32. Matula, D.: Who is in SIGCSE? SIGCSE Bulletin **2**(5), 57–67 (1970)
33. McDowell, C., Werner, L., Bullock, H., Fernald, J.: The effects of pair-programming on performance in an introductory programming course. In: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, SIGCSE '02, pp. 38–42. Association for Computing Machinery, New York, NY, USA (2002). URL https://doi.org/10.1145/563340.563353
34. Miller, J.E.: Editor's comments. SIGCSE Bulletin **29**(2), 1 (1997)

35. Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., Balik, S.: Improving the CS1 experience with pair programming. In: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '03, pp. 359–362. Association for Computing Machinery, New York, NY, USA (2003). URL https://doi.org/10.1145/611892.612006
36. Papamitsiou, Z., Giannakos, M., Simon, Luxton-Reilly, A.: Computing education research landscape through an analysis of keywords. In: Proceedings of the 2020 ACM Conference on International Computing Education Research, ICER '20, p. 102–112. Association for Computing Machinery, New York, NY, USA (2020). DOI https://doi.org/10.1145/3372782.3406276
37. Pears, A., Malmi, L.: Values and Objectives in Computing Education Research. ACM Transactions on Computing Education **9**(3) (2009)
38. Pears, A., Seidman, S., Eney, C., Kinnunen, P., Malmi, L.: Constructing a core literature for computing education research. SIGCSE Bull. **37**(4), 152–161 (2005). DOI https://doi.org/10.1145/1113847.1113893
39. Perianes-Rodriguez, A., Waltman, L., van Eck, N.J.: Constructing bibliometric networks: A comparison between full and fractional counting. Journal of Informetrics **10**(4), 1178–1195 (2016). DOI https://doi.org/10.1016/j.joi.2016.10.006.
40. Piech, C., Sahami, M., Koller, D., Cooper, S., Blikstein, P.: Modeling how students learn to program. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12, pp. 153–160. Association for Computing Machinery, New York, NY, USA (2012). URL https://doi.org/10.1145/2157136.2157182
41. Repenning, A., Webb, D., Ioannidou, A.: Scalable game design and the development of a checklist for getting computational thinking into public schools. In: Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE '10, pp. 265–269. Association for Computing Machinery, New York, NY, USA (2010). URL https://doi.org/10.1145/1734263.1734357
42. Settle, A., Becker, B.A., Duran, R., Kumar, V., Luxton-Reilly, A.: Improving Global Participation in the SIGCSE Technical Symposium: Panel. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20, pp. 483–484. Association for Computing Machinery, New York, NY, USA (2020). URL https://doi.org/10.1145/3328778.3366979
43. Simon: A Classification of Recent Australasian Computing Education Publications. Computer Science Education **17**(3), 155–169 (2007). URL http://www.informaworld.com/10.1080/08993400701538021
44. Simon: Emergence of computing education as a research discipline. Ph.D. thesis, Aalto University School of Science (2015)
45. Simon, Carbone, A., Raadt, M.d., Lister, R., Hamilton, M., Sheard, J.: Classifying Computing Education Papers: Process and Results. In: R. Lister, M. Caspersen, M. Clancy (eds.) Fourth International Computing Education Research Workshop (ICER 2008). ACM Press, Sydney, Australia (2008)
46. Simon, Sheard, J.: Twenty-Four Years of ITiCSE Papers. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, pp. 5–11. Association for Computing Machinery, New York, NY, USA (2020). URL https://doi.org/10.1145/3341525.3387407
47. Tedre, M., Simon, Malmi, L.: Changing aims of computing education: a historical survey. Computer Science Education **28**(2), 158–186 (2018). URL https://doi.org/10.1080/08993408.2018.1486624
48. Thomas, L., Ratcliffe, M., Woodbury, J., Jarman, E.: Learning styles and performance in the introductory programming sequence. In: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, SIGCSE '02, pp. 33–37. Association for Computing Machinery, New York, NY, USA (2002). URL https://doi.org/10.1145/563340.563352

49. Valentine, D.W.: Cs educational research: A meta-analysis of SIGCSE technical symposium proceedings. SIGCSE Bull. **36**(1), 255–259 (2004). URL https://doi.org/10.1145/1028174.971391
50. Walker, H.M., Dooley, J.F.: The history of the SIGCSE submission and review software: From paper to the cloud? In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, pp. 1074–1080. Association for Computing Machinery, New York, NY, USA (2019). URL https://doi.org/10.1145/3287324.3287427
51. Walker, T.M.: Computer science curricula survey. SIGCSE Bulletin **5**(4), 19–28 (1973)
52. Werner, L., Denner, J., Campe, S., Kawamoto, D.C.: The fairy performance assessment: Measuring computational thinking in middle school. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE '12, pp. 215–220. Association for Computing Machinery, New York, NY, USA (2012). URL https://doi.org/10.1145/2157136.2157200
53. Wilson, B.C., Shrock, S.: Contributing to success in an introductory computer science course: A study of twelve factors. In: Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education, SIGCSE '01, pp. 184–188. Association for Computing Machinery, New York, NY, USA (2001). URL https://doi.org/10.1145/364447.364581
54. Wise, M.J.: Yap3: Improved detection of similarities in computer program and other texts. In: Proceedings of the Twenty-Seventh SIGCSE Technical Symposium on Computer Science Education, SIGCSE '96, pp. 130–134. Association for Computing Machinery, New York, NY, USA (1996). URL https://doi.org/10.1145/236452.236525

# ITiCSE Working Groups as an Engine for Community-Building


**Robert McCartney and Kate Sanders**

## 1 Introduction

The ITiCSE working groups provide a remarkable opportunity for collaboration. Working groups are small groups, typically 10–12 people, who work together on a specific computing-education-related project. Any ITiCSE attendee can submit a proposal for a working group; any attendee can apply to join one of the accepted working groups. Groups generally communicate electronically and do some work before the conference, then meet face-to-face for 3 or 4 days at the conference site.[1] Finally, during the month or two following the conference, working-group members complete and submit a written report on their work.

From the first ITiCSE in 1996 through the latest completely in-person ITiCSE in 2019, 129 working-group reports were accepted for publication. Topics have ranged from the internet to the internet of things, from specific courses such as introductory programming, computer architecture, discrete mathematics, and research methods to broader themes such as social and ethical issues, computing for the common good, and sustainability. Several of the working-group reports have been widely cited, for example McCracken et al. [16], Lister et al. [10] and Hamer et al. [9].

[1] The exact number of days and whether they are before, after, or partially during the conference has varied slightly from time to time.

R. McCartney (✉)
University of Connecticut, Storrs, CT, USA
e-mail: robert@engr.uconn.edu

K. Sanders
Rhode Island College, Providence, RI, USA
e-mail: ksanders@ric.edu



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_11

Moreover, these collaborations involve groups with no single institutional—and rarely even a single national—connection. All 129 in-person working groups have been multi-institutional, all but one have been multi-national, and the large majority (after the first year) have included both researchers who have previous working-group experience and those who do not. Over 600 researchers have participated.

In this chapter, we present a quantitative analysis of the working-group collaborations. Specifically, we investigate the following research questions:

1. How effectively do the working groups connect newcomers to experienced working-group participants?
2. How effectively do the working groups connect participants with computing education venues other than the working groups, specifically ICER, ITiCSE, and SIGCSE?
3. What is the pattern of collaborations between working-group authors after the first working group in which they collaborate?
4. How effectively do the working groups connect participants with computing-education researchers from other countries?

We limit our investigation to 1996–2019 working groups because, since the start of the pandemic, working groups have lacked the intense face-to-face contact of the pre-2020 working groups.

## 2 Background and Related Work

Much of the quantitative analysis of computing-education collaborations has been based on collaboration networks. In Sect. 2.1, we provide some background on collaboration networks. In the remaining subsections, we situate this chapter within earlier studies related to collaboration in computing education and connections between newcomers and old-timers, between venues, and between countries.

### *2.1 Background: Collaboration Networks*

For our purposes, a *collaboration* can be thought of as a relationship between two individuals—one created when they are both authors on the same paper. The first step in analyzing the collaborations at a venue is to collect basic descriptive data: how many distinct authors there are, how many papers, the average number of authors on a paper, the average number of distinct collaborators per author, and the average number of collaborations per author.

*Collaboration networks* support a more extensive analysis. They represent a set of papers as a graph in which each node stands for an individual author, and there is an edge between two authors if those authors have collaborated on one of the papers in the set.

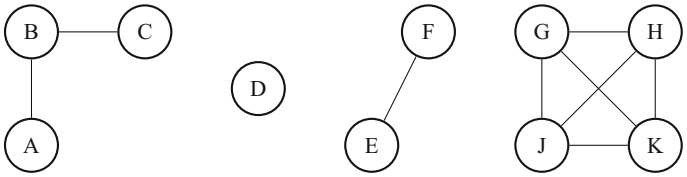


**Fig. 1** A collaboration network representing a simplified venue. The venue has ten authors (represented by nodes A, B, C, D, E, F, G, H, J, K, and L), nine collaboration relationships (represented by edges AC, AB, EF, GH, GJ, GK, HJ, HK, and JK), and four subgroups (corresponding to connected components ABCD, D, EF, and GHJK)

For example, the collaboration network shown in Fig. 1 represents a venue with only ten authors, A, B, C, D, E, F, G, H, J, and K. Each node corresponds to one of the authors. Author B has collaborated with both A and C, but Authors A and C have not collaborated directly. Authors E and F have collaborated with each other, but not with any of the rest, and Author D has not collaborated with any of the others. Authors G, H, J, and K have all collaborated with each other, but none of them has collaborated with any of the other authors.

It is common for collaboration networks to include a number of connected components, as this one does. These connected components correspond to subgroups of the larger group of authors. More precisely, two separate connected components correspond to two groups of authors where no one in either group has co-authored a paper with anyone in the other group.

The number of connected components in a collaboration network can be anywhere from 1 (all N nodes connected in a single component) to N (N separate nodes with no edges). A graph with a single component would represent a group of authors where each of them has a collaboration relationship with each of the others, either direct (like Authors A and B in Fig. 1) or indirect (like Authors A and C). At the opposite extreme, a collaboration network with N components would represent a group of authors none of whom has collaborated with any of the others, where all papers would be single-author papers. In real-world examples, both extremes are unlikely.

Although single-component collaboration networks are rare, it is characteristic of collaboration networks that model real-world examples that "a large percentage of the individuals are connected into a single very large cluster" [18, p. 19]. Size is measured as a percentage of the authors at a venue, and some examples are as high as 83% [18]. This cluster is called the *giant component*. Because it represents a large percentage of the authors, the giant component can be thought of as representing the "essence of the community" [18, p. 19].

The authors in the giant component are all connected—but how tightly are they connected? One measure is the *diameter* of the giant component. The diameter of any component is defined as the minimum number of edges that need to be crossed to reach from one randomly chosen node in the component to another. For example, in Fig. 1, the diameter of the giant component, component GHJK, is 1, because

it is possible to reach any node from any of the others by crossing at most one edge. By contrast, component ABC has a diameter of 2, because any node can be reached from any other by crossing at most 2 edges, and for at least one pair of nodes (Nodes A and C), 1 edge is not sufficient.

For any component containing more than one node (which it is safe to assume the giant component does), 1 is the minimum diameter. A giant component with diameter 1, like component GHJK, would represent a group of authors each of whom has published at least one paper with each of the others. The maximum diameter is $N - 1$ (the number of edges needed to connect all the nodes in single file, like Nodes A, B, and C in Fig. 1). Thus, a venue with 602 authors (the number of working-group authors from 1996 to 2019) could theoretically have a diameter anywhere from 1 to 601. As with the size of the giant component, both extremes of the diameter are unlikely.

The diameter suggests something about both social connections and the spread of ideas within a community. A diameter greater than 2 might seem like a lot—how likely are you to meet your collaborator's collaborator's collaborator? But you might still be influenced by their ideas. If Author M tells Author N that their paper must include a threats to validity section, or suggests a useful tool for jointly editing papers, Author N might pass that on to Author P, and then Author P to Author Q.

All these characteristics of collaboration networks—the number of connected components, the size of the giant component, and the diameter of the giant component—can change over time. Whenever an author publishes in a venue for the first time, a new node is added to the collaboration network. When two authors publish a paper together who had not previously collaborated at that venue, a new edge is added between the nodes that represent those authors.

For example, consider Fig. 2. In the leftmost box, the collaboration network contains a single connected component, representing three authors, one of whom (Author S) has collaborated with each of the other two. This component has a diameter of two.

In the center box, the collaboration network now contains two separate connected components, because a new node (node U) has been added. Node U represents a new author who has not collaborated with any of the previous authors at this venue. The RST component is the giant component. It contains the same three nodes as before, but its diameter has changed. A new edge has been added between R and T, indicating that R and T have published a paper together for the first time at this venue, with the result that the diameter of the RST component is now 1.

Finally, in the right-hand box, the number of connected components in the collaboration network is back down to 1. The giant component now contains four nodes, 100% of the author nodes. The new edge from node R to node U indicates that Authors R and U have now collaborated together, so Author U is now part of the larger group. Author U has only collaborated directly with Author R, however, and that fact is reflected in the diameter of the giant component, which has now increased from 1 to 2.

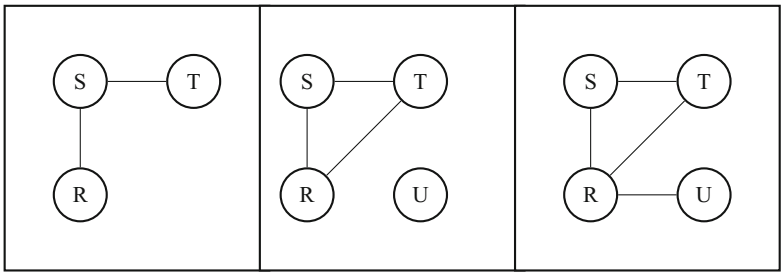


**Fig. 2** A sequence of collaboration networks, illustrating change over time

To analyze how newcomers join a collaboration network, Miró Julià et al. introduced a technique called "MEIN analysis" [18]. MEIN analysis classifies each new paper (i.e., each new collaboration) into one of four types:

- *(N)ewcomer*: a paper where all authors are newcomers, that is, they had not previously published at the conference;
- *(I)nternal*: a paper where all authors are old-timers and all were previously part of the same component;
- *(E)xtend*: a paper where there are both old-timers and newcomers, and all of the old-timers were previously part of the same component; and
- *(M)erge*: a paper where there are old-timers who were *not* previously part of the same component. These papers may or may not include newcomers.

Extend papers join newcomers to an existing connected component, and Merge papers join any newcomers that they include to two or more existing connected components (which are themselves being joined). Internal and Newcomer papers, on the other hand, make no connections between old-timers and newcomers.

Note that the collaboration networks shown in Figs. 1 and 2 do not capture the number of papers an author has in the dataset or the number of collaborations between a given pair of authors. For example, Author D could have a single publication, two, or more; Authors E and F could have collaborated on one paper or dozens. Component GHJK may correspond to a single four-author paper, six two-author papers, 20 four-author papers, or many other possible combinations. Weights could be added to the edges to record this information, but for this part of our analysis, capturing the existence (or not) of a direct collaboration relationship is sufficient.

In sum, collaboration networks allow us to characterize venues in a number of ways. For example:

- Are there subgroups within the larger community?
- If so, how many are there, and how large are they?
- What percentage of the authors belong to the largest subgroup?
- How does the size of the subgroups change over time?
- How many new authors join the community?

- Is the largest subgroup a fixed "in-group", or do other authors join it?
- What percentage of new authors join the largest subgroup (if any), and how long does it take?
- How tightly is the largest subgroup connected, and how does that change over time?

We discuss these questions further in relation to the working groups throughout the rest of the chapter.

## *2.2 Connections Between Newcomers and Old-Timers*

In a 2012 paper [18], Miró Julià et al. published a thought-provoking claim: that computing-education conferences were more "introverted" than computer-science conferences (or alternatively, that computer-science conferences were more extroverted). Specifically, they argued that the authors at computing-education conferences had fewer collaborators, there were fewer authors per paper, and there were fewer collaborations between newcomers and old-timers.

They based this interpretation on an analysis of 13 conferences: six computing-education conferences, six computer-science conferences, and ICER, which they considered separately because it had aspects of both. All the conferences in their dataset had at least 45% newcomers, but the authors at computing-education conferences had fewer collaborators, on average, and computing-education papers had fewer authors per paper. In addition, fewer of the authors at computing-education conferences were part of the giant component: from 14% to 46% of the total authors. The research conferences' giant components ranged from 43% to 83% (with one outlier at 18%). Finally, based on an MEIN analysis, the computing-education conferences had more papers where old-timers and newcomers do not collaborate (that is, the Internal and Newcomer papers, plus possibly some of the Merge papers).

ICER was something of a hybrid, resembling the research venues in the size of its giant component, authors per paper, and collaborations per author, but resembling the computing-education venues in its proportion of Internal plus Newcomer papers. ICER had a larger giant component than the other computing-education conferences and more collaborators per author, but even there, old-timers were more likely to publish with authors they had collaborated with before and less likely to collaborate with newcomers to the venue than authors at the computer-science conferences.

Most of the education conferences started around 1996, so Miró Julià et al. chose to include in their dataset papers from instances of these conferences held between January 1996 and September 2011 (the most recent data available when the paper was written). For ICER, they included papers from the start of the conference in 2005–2011.

Notably, however, the ITiCSE Working Groups were not included in this dataset. They were omitted deliberately, since none of the other conferences had anything

**Table 1** Connections between newcomers and old-timers at the ITiCSE working groups (WG) compared to other venues (1996–2011). The newcomer percentages do not include the first year. Source: [13]

| | ITiCSE | SIGCSE | ICER | WG |
|---|---|---|---|---|
| Newcomers | 69% | 67% | 55% | 58% |
| Giant component size | 14% | 29% | 49% | 95% |
| Authors per paper (avg.) | 2.29 | 2.42 | 3.00 | 8.00 |
| Collaborations per author (avg.) | 1.74 | 1.84 | 3.31 | 7.02 |

**Table 2** MEIN analysis of ITiCSE working groups compared to other venues (1996–2011). Sources: [18] (for the six computing-education conferences) and [13] (for the working groups). SIGCSE and ITiCSE were two of the six CS Education conferences, but not reported individually in [18]

| | Six CS Education conferences | ICER | WG |
|---|---|---|---|
| Merge papers | 4.0–8.5% | 8.4% | 9.6% |
| Extend papers | 22–39% | 35% | 84.9% |
| Internal papers | 18–24% | 23% | 1.4% |
| Newcomer papers | 30–54% | 34% | 4.1% |

comparable. But still, due to the highly collaborative, open nature of the working groups, this omission raised a question: would including the working groups in the analysis make a difference?

The answer was yes. In 2018, we added an analysis of the working groups to Miró Julià et al.'s work [13]. Following their methodology, we computed a collaboration network for all the working groups from the same time period they had used (1996–2011) and compared the working-group results with Miró Julià et al.'s. Because they had made their data publicly available [17], we were able to check our computations against theirs. In short, the working groups were not only more extroverted than the computing-education conferences Miró Julià et al. analyzed, they were more extroverted than all the conferences in their dataset.

The working-group results are given in the right-hand column of Table 1, alongside results for SIGCSE, ITiCSE, and ICER. As shown, the average number of authors per working-group paper was more than twice as high as any of these conferences. Similarly, the average number of collaborations per author was more than twice as high, and the working groups' giant component included nearly all of the authors: 95%, higher than any of the other conferences.

Moreover, the results of the MEIN analysis (Table 2)[2] showed that working-group newcomers collaborated with old-timers on the vast majority of the working-group papers. To start with, 84.9% of the working-group papers were Extend papers. Not only that, but all the working groups' Merge papers (a combination of old-

[2] The maximum percentage of Merge papers for the six computing-education conferences, 8.5% (shown in Table 2), was reported in [18] as an outlier and inadvertently omitted in [13]. Instead, the second-highest value, 6.5%, was reported as the maximum. We correct that omission here.

**Table 3** How closely working-group authors are connected, in relation to ITiCSE, SIGCSE, and ICER (as of 2011). Source: [13]

| | ITiCSE | SIGCSE | ICER | WG |
|---|---|---|---|---|
| Total authors | 2322 | 2272 | 178 | 368 |
| Giant component size | 14% | 29% | 49% | 95% |
| Giant component diameter | 22 | 15 | 6 | 6 |
| Number of components | 817 | 577 | 35 | 3 |

timers from different groups) also included newcomers (which is not necessarily the case for Merge papers). Thus, at 94.5% of the working groups—all the Extend and Merge groups—newcomers were working alongside old-timers. By comparison, the percentage at the six CS education conferences and ICER was at most 47.5%. If, as is possible, some of the Merge papers at the other venues did *not* include newcomers (unlike the working groups' Merge papers), the gap between the working groups and the other venues would be even greater.

Finally, the authors in the working groups' giant component were tightly connected. Recall that because there were 368 total authors through 2011, the giant component's diameter could theoretically be as low as 1 or as high as 367. Given the unusually collaborative nature of the working groups, we might have expected that the diameter would be in the low part of the range. Still, the results are surprising. As shown in Table 3, the working-groups' diameter is 6, tied with ICER's, and much smaller than SIGCSE's (at 15) or ITiCSE's (at 22). This cannot be explained just by the absolute number of authors in the giant component. The working groups' giant component includes roughly the same number of authors as ITiCSE's, and yet the diameter of the working groups' giant component is less than a third of the size of ITiCSE's. Moreover, although the working groups and ICER had the same diameter, the working groups connected both a much larger percentage of their authors (95% vs. 49%) and a larger absolute number of authors, compared to ICER.

## *2.3 Connections to the Larger Computing-Education Community*

Simon has published a series of papers focused on bibliometric analysis of computing-education venues [See., e.g., 20–23]. We discuss here those aspects related to ITiCSE working-group collaborations. In particular, Simon recently published an analysis of ITiCSE papers from 1996 through 2019 in which he considered conference papers and working-group reports together [22]. Thus, his analysis is based in part on the same dataset we examine here.

Unlike this chapter, however, Simon [22] included the ITiCSE conference papers and working-group reports in the same dataset. Working groups are held at ITiCSE, working-group members are required to register for the conference, and—while

working-group members do not join more than one working group in a year—they can and do publish conference papers as well as working-group reports. This perspective sheds light on how the working groups affect the larger ITiCSE community.

One example of the influence of the working groups is their effect on the size of the combined ITiCSE giant component. Taken separately (through 2011), ITiCSE's conference papers had a giant component that included 14% of its authors, and the working groups, 95%, as shown in Table 3. For his combined dataset, Simon reports a giant component of 48% [22], above that for SIGCSE or for ITiCSE's conference papers alone.

## *2.4 Connections Between Countries*

Another way to look at collaborations is to examine the geographical location of the collaborators. Working groups often provide their members with the opportunity to work with authors from other countries: as Simon found, 124 of the 129 working groups in our dataset (96.1%) were multi-national, compared with 115 of 1295 of ITiCSE's conference papers (8.9%) [22].

Lunn et al. [11] analyzed the geographical locations of authors at ICER, ITiCSE, and TOCE from 2015 to 2020, supplemented by the affiliations of the students in the Doctoral Consortia, in order to determine where students interested in graduate study in computing education might find programs. They did include the working groups, but did not separate out their results by venue, and did not discuss collaborations.

Later that year, Zhang et al. published a detailed analysis of the geographical location of the authors at SIGCSE, ICER, and ITiCSE [24]. It did not include the ITiCSE working groups, but did, remarkably, include all available papers from the lifetime of each conference, extending back as far as SIGCSE's first proceedings in 1970. Most relevant to this chapter, it included a graph of the average number of countries per connected component, for each year of each conference. ICER fluctuates between 1 and about 1.6; ITiCSE (after its first year) and SIGCSE are both very close to 1 each year. Since 1 is the minimum possible number of countries, and the maximum is substantially larger, as the paper concludes, "the majority of authors still opt to collaborate with people from the same country" [24, p. 586].

Apiola et al. [1] published a bibliometric analysis of computing education research from a variety of angles, including topics, keywords, geography, and more. Based on the analysis of author locations and collaborations, the paper expresses concern that there may exist a "dominant" group of institutions that collaborate primarily with each other, and that certain areas of the world, in particular Africa, are under-represented in the computing education literature.

### 2.5 Summary

Working groups had more authors per paper, had more collaborations per author, and were far more likely to make connections between newcomers and old-timers than the other venues. We found no work addressing follow-up collaborations after the working groups, but newcomers quickly became connected to the giant component, and the authors in the giant component—the vast majority of the authors—were closely connected to each other. Furthermore, the connections formed by the working groups also served to connect members of the larger ITiCSE community more closely together and to connect working-group authors to researchers from other countries.

## 3 Methodology

To perform our analysis, it was first necessary to get clean data concerning papers and their authors, then to analyze the data to address our research questions. We look at these tasks in turn.

### 3.1 Collecting and Cleaning Author and Paper Data

In order to extract and clean up the author and paper data, we began by assigning a sequence number to each paper within each venue and year, so any paper could be identified as a venue-year-sequence number triple, and we chose a canonical name (a string of the form surname rest-of-name) and assigned a unique integer IDs to each author.

The source data we used for these authors and papers were bib files downloaded from the ACM Digital Library: for the years 1996–2019 for the working group reports, and for the years 2012–2019 for SIGCSE, ITICSE, and ICER. These required some cleaning to remove entries for things like session chairs (included some years) and panels. The other data were included in bib files collected for ITiCSE, SIGCSE, and ICER by Miró Julià et al., made available at their website [17].

Once we had clean versions of the bib files for a venue, having a program extract the information was straightforward: for each bib-entry in the file,

1. Extract the author and year fields
2. Assign each paper an integer sequence number for its year,
3. Extract the list of authors from the author field,
4. Write an entry to a file for each author of the form Venue Year Sequence-no Author-name

This produces a text file with a line for each author on each paper. This file is used to produce another file, which lists each author (in alphabetical order) and a unique integer ID.

Extracting the list of authors was straightforward in the ACM bibs, as they are given as "surname, rest-of-name" separated by "and", e.g. "Danielsiek, Holger and Paul, Wolfgang and Vahrenhold, Jan", and generally put the comma in the correct place, e.g. "Huff Jr., James W.". The Miró Julià et al. bibs were in a slightly different format, so a slightly modified version of the program was used to extract the paper-author entries from these files.

The problem with this approach is that it requires any author in the data to have the same form of his or her name wherever it appears in the bib files. As others have noted [18, 22, 24], authors use different versions of their names on different papers, sometimes have their names spelled wrong, and sometimes change their names over time. Moreover, because we wanted to trace collaborations that working-group authors might have had in the other three venues, we needed to check authors across, as well as within, venues.

To address this problem, we started with the complete list of authors for all four venues, then tried to determine whether similar names (e.g., having the same surname and first initial) belonged to the same person. Sometimes this was easy (someone publishing with the same co-authors), but often it meant checking their publications for their affiliation. Once we determined that two names referred to the same author, we edited the names in the .bib files and re-ran the above code to produce more accurate authors and paper data. We were fairly conservative, not changing a name unless we had good evidence that the change was warranted.

The ultimate result of this cleaning was information about 5535 papers and 7809 distinct authors: which authors were on each paper, and where and when each paper was published.

### *3.2 Analysis: Connections Between Newcomers and Old-Timers*

The various analyses were done by processing these paper-author data. We built separate collaboration networks for each of ICER, ITiCSE, SIGCSE, and the working groups, with each venue's collaboration network based on its own author and paper data. Based on these collaboration networks, we were able to compute the general network characteristics for each individual venue—number of authors, number of papers, and so forth—as well as information about the connected components. Given the connected components, we could count them, find the largest one (i.e., the giant component), report its size, and compute its diameter.

We also examined the change in these networks over time. Recall that the MEIN analysis requires information about paper authors and their connected components from previous years. To calculate the MEIN values for year $n$ ($n > 1$), we

built a combined collaboration graph with its connected components for all of the years 1 to $(n - 1)$, from which we could determine, for each author and paper in year $n$, whether each author was a newcomer or old-timer, and—if an old-timer—which connected component they were in the years 1 to $(n - 1)$ graph, sufficient information to determine whether a paper was Merge, Extend, Internal, or Newcomer.

The collaboration networks for the various venues enabled us to look at within-venue collaborations. For each author in the graph, we calculated the number of papers they wrote, the number of collaborations they have been in, and the distinct authors they have collaborated with.

### *3.3 Analysis: Connections Between Working Groups and Other Venues*

We also built a combined collaboration network including all four venues. Based on these collaboration networks, we were able to compute the general network characteristics for the overall dataset.

### *3.4 Analysis: Follow-Up Collaborations*

The combined collaboration network also allowed us to examine an individual's publications and collaborators across all four venues. We wrote a program that extracted all of the collaborations across the four venues and associated a *sequence of collaborations* with each collaborating pair. Each collaboration in the sequence corresponds to a paper, and the elements of the sequence are ordered as follows: if two collaborations have different years, the one with the earlier year is earlier. If two collaborations have the same year, we order by venues: SIGCSE < ITiCSE < ICER < WG; the rationale is that the submission deadlines for SIGCSE, ITiCSE, and ICER reflect this order, and the working groups meet at ITiCSE, which is after the ICER submission date. This information is used to build summaries of temporal patterns of collaborations inside and outside of working groups by all of the authors. How these summaries are used is explained more in Sect. 4.

### *3.5 Analysis: Connections Between Different Countries*

To investigate the extent of multi-national collaborations at the working groups, we built on the analysis of the collaboration networks. Given the output of the previous computations, spreadsheets were sufficient for this part of the analysis.

The first step was to create a spreadsheet for the authors. It listed all the working groups in chronological order, with a separate line for each member of each working group. Each line contained the following previously computed information:

- Author name
- Year
- Working-group number
- MEIN classification of that working group

The working-group numbers are those used at the conference, which are re-used from year to year. The file is sorted by year, and within year, by working-group number. The author names had been previously cleaned as described above.

To determine which authors were newcomers, one additional item was computed and added to each row:

- The number of times the author in row N had appeared in rows 1 to N − 1 (that is, how many previous working groups the author had participated in).

Where the computed value was 0, the author was a newcomer in that group; otherwise, the author was identified as an old-timer.

Next, a working-group spreadsheet was created, with one row for each working group. The title pages of the 129 working group reports were examined manually to identify the countries with which authors were affiliated, and the results recorded in the spreadsheet.

Perhaps because the working groups are so multi-national, the issues that others have reported with identifying affiliations [23] almost never arose. There was only one decision: to change Macedonia to North Macedonia, since there were working-group members from before and after the year when that country changed its name. Since this decision affected only one individual, it had only a minor effect on the analysis.

The next step was to determine what component each author belonged to by the end of that year's working groups. The first year's working groups were considered Components 1, 2, 3, 4, and 5, one per working group (since the working groups have disjoint sets of authors). Component 3, with 10 members, was the largest and consequently the giant component.

For the second year (1997), we filled in a column with the component that each author belonged to at the end of 1997 working groups, as follows:

- If a working group was classified as *Extend*, we checked to see which group the old-timers in that group belonged to (the "old component"). The newcomer members of the group were then marked as belonging to the old component. The set of countries associated with the new extended component was the union of the set of countries previously associated with the old component and those associated with the newcomers.
- If a working group was classified as *Merge*, we checked to see which groups were being merged—that is, which groups the old-timers belonged to. As it turned out, there were always exactly two such groups, and one of them was

always the giant component (not surprising, because the giant component rapidly became much larger than the others). All the old-timers in the smaller component were reclassified as belonging to the giant component. If the Merge group also included newcomers (as in practice, Merge groups always did), these were also added to the giant component. The set of countries associated with the new even-more-giant component was the union of the set of countries previously associated with the giant component, those associated with the smaller component, and those associated with the newcomers.

- There was only one group of type *Internal* (consisting entirely of old-timers from the same component). In that case, nothing needed to be done; all members of the working group were already members of the same component, and the countries they were affiliated with were already included in that component's set of associated countries.
- In the case of each of the four *Newcomer* working groups (consisting entirely of newcomers), a new separate component was created and the set of countries associated with it was the set of countries with which the members of that working group were associated. The members of the working group were assigned to the new component.

This process was repeated, year by year, for each of the working groups. Each year took the previous year's components as a starting point, and looked at the current year's groups to determine which groups had been merged, extended, or made into new separate components, and the effect of those changes on the components and the number of countries associated with each component.

Finally, for each year, the number of components at the end of that year's working groups, the total number of countries to date, a list of the new countries added that year, and the number of countries associated with each component were recorded. With that information, it was straightforward to compute minimum, maximum, and average numbers of countries per component, both overall and by year. This information was the basis for the graph shown in Fig. 3 in Sect. 4.

## 4 Results

In this section, we begin with an overview of collaboration and collaborators at the working groups, compared to ITiCSE, SIGCSE, and ICER. Next we define and characterize newcomers, both newcomers to our individual venues and to the community. Then we present results concerning collaborations between newcomers and oldtimers; connections between working-group authors and ITiCSE, SIGCSE, and ICER; follow-up publications by authors who first collaborated at a working group; and the way in which working groups connect researchers from different countries.

**Table 4** Basic characteristics of conference and working-group authors and papers. Note that the average authors per paper is the sum of the number of authors on each paper, divided by the number of papers—that is, a measure of how many authors a typical paper has

| | ITiCSE (1996–2019) | SIGCSE (1996–2019) | ICER (2005–2019) | ITiCSE WG (1996–2019) |
|---|---|---|---|---|
| Number of authors | 3483 | 4745 | 546 | 602 |
| Number of papers | 2253 | 2838 | 315 | 129 |
| Authors per paper (avg.) | 2.5 | 2.8 | 3.2 | 8.4 |
| Avg. papers per distinct author | 1.6 | 1.7 | 1.9 | 1.8 |
| Pct. authors with just 1 paper | 74.8% | 71.6% | 71.1% | 64.6% |
| Pct. papers with just 1 author | 29.2% | 22.7% | 11.1% | 0.0% |

## *4.1 Results: Basic Overview*

### 4.1.1 The Venues

Table 4 includes basic venue information, updated through 2019. The author and paper numbers are for the number of distinct authors and papers within each venue. Some values—papers per author and percent of authors who have only published a single paper at the venue—are quite similar across the venues.

The first obvious difference is in the scale of these venues. They represent a range of sizes. SIGCSE has had 4745 authors and 2838 papers; ITiCSE has had 3483 authors and 2253 papers; ICER has had 546 authors and 315 papers. Like ICER, the working groups are relatively small, at 602 authors and 129 papers. Of all the 7809 authors in our dataset, 44.6% have published at ITiCSE, 60.1% have published at SIGCSE, 7.7% have published at a working group, and 7.0% have published at ICER.

The second obvious difference is the number of papers with just one author: zero at the working groups. This is the most obvious feature of the working groups—they are all collaborative. There are no single-author working-group papers.

### 4.1.2 Collaborations and Collaborators

We can measure the interactions among authors by their collaborators—the number of distinct individuals with whom they have been co-authors—or by their collaborations. These measure different things. Our definition of *collaboration* is taken from Miró Julià et al. [18]. By this definition, each collaboration is a relation between two authors on the same paper. When counting these collaborations, however, each of the two authors gets half-credit for that collaboration.

For example, if authors A and B collaborate on one paper, each is counted as having 0.5 collaborations; if they collaborate on six papers, each is counted as having three collaborations; but either way, they each have a single distinct collaborator. If authors A, B, and C collaborate on a single paper, each author has

**Table 5** Authors, papers, and collaborators within each of our conferences. Note that each collaboration counts one half for each author, so the average/maximum collaborations can be fewer than the average/maximum distinct collaborators, but the averages are calculated in the same way as those in Table 1

| | ITiCSE (1996–2019) | SIGCSE (1996–2019) | ICER (2005–2019) | ITICSE WG (1996–2019) |
|---|---|---|---|---|
| Distinct authors | 3483 | 4745 | 546 | 602 |
| Distinct collaborators per author (avg.) | 3.3 | 4.2 | 4.6 | 13.4 |
| Max distinct collaborators per author | 44 | 64 | 41 | 93 |
| Collaborations per author (avg.) | 2.0 | 2.5 | 3.0 | 7.4 |
| Max collaborations per author | 38 | 47 | 41 | 57 |
| Pct. distinct authors with no collaborators | 7.5% | 5.4% | 2.4% | 0 |

two distinct collaborators. When counting collaborations, however, each author is part of two collaborations (one with each of his or her co-authors), and is counted as having $2 * 0.5 = 1$ collaboration as result of that paper.

Whether we look at collaborators or collaborations, the working groups are extremely collaborative. Table 5 presents the collaborator and collaboration data for our venues. In general the number of collaborators and collaborations can be quite large. For each of the venues, the maximum values for collaborators and collaborations are large relative to the averages. But the working-group authors have much higher values for average collaborators, maximum collaborators, average collaborations, and maximum collaborations than the other venues. In addition, as shown in the bottom row—and by design—all working-group participants have collaborators. There are no single-author working-group reports.

### 4.1.3 Newcomers

In the previous discussion, and consistent with Miró Julià et al., we define a newcomer author relative to the venue of the paper, e.g. when a person publishes a paper at SIGCSE, but has not published at SIGCSE in a previous year, they are a SIGCSE newcomer. As an alternative we could define newcomers over a set of venues, that is, that an author is a newcomer if he or she publishes for the first time in any of a set of venues. In effect, we consider the venues in our dataset as a proxy for the computing education community; the first time someone publishes at any of these venues, they are a newcomer in the community.

Using this idea, we additionally classified each author on each paper in our combined set as whether they were newcomers to the community or not. Table 6 presents the average numbers per year for authors, venue newcomers, and community newcomers at each of our venues. The percentage of community newcomers at working groups is less than seen at SIGCSE and ITiCSE, and roughly the same as ICER.

**Table 6** Venue and community newcomers across conferences. These do not include the first year of each conference's data. Counts given are average number of distinct authors per year

| | ITICSE (1997–2019) | SIGCSE (1997–2019) | ICER (2006–2019) | ITiCSE WG (1997–2019) |
|---|---|---|---|---|
| Distinct authors (avg./year) | 213.6 | 308.8 | 63.6 | 45.3 |
| Distinct newcomers to venue (avg./year) | 145.3 | 199.9 | 36.0 | 24.2 |
| Percent newcomers to venue | 68% | 65% | 57% | 54% |
| Distinct newcomers to community (avg.) | 120.3 | 183.1 | 17.4 | 12.1 |
| Percent newcomers to community | 56% | 59% | 27% | 27% |

Although lower than SIGCSE or ITiCSE, the percentage of working-group newcomers each year after the first year is, on average, over 50%. Considering distinct working group authors, there have been 296 distinct community newcomers out of the 602 distinct authors: for nearly half of the working-group authors, their first paper published across our venues was a working-group report. In terms of establishing collaborations, these community newcomers worked with, on average, just over eight new collaborators. In terms of the working groups, 102 of the 124 working groups after the first year had at least 1 community newcomer, and 123 of 124 (all but one) had at least one venue newcomer.

## *4.2 Results: Connections Between Newcomers and Old-Timers*

The data through 2019 show that the working groups continue to connect the large majority of their newcomers to old-timers. As shown in Table 7, the percentage of newcomer authors is more than 50% for each of our venues, with somewhat more at ITiCSE and SIGCSE. Working-group newcomers are much more likely to be working with old-timers than newcomers at the other venues are, however. While ITiCSE, SIGCSE, and ICER show some variation (ITiCSE and SIGCSE have relatively more all-newcomer papers, ICER has relatively more all old-timer papers), the differences between all three venues and the working groups are striking: 96% of the working-group papers have a mix of old-timer and newcomer authors, over twice the percentage of any of the other three venues. Consequently, the number of working groups that do *not* include both old-timers and newcomers is necessarily low: there has only been one working group with no newcomers, and only four where there were only newcomers.

Consider, for example, the Leeds working group in 2003 [10]. Of the four organizers, one, Lynda Thomas, was an experienced working-group member; the other three were newcomers. Of the other eight participants, three were old-timers and five were newcomers. With a total of four old-timers and eight newcomers, all but one of the collaboration pairs were people who had never published together before.

**Table 7** Mix of old-timers and newcomers across conferences. These do not include the first year of the conference data as all of the papers would be all newcomers

| | ITICSE (1997–2019) | SIGCSE (1997–2019) | ICER (2006–2019) | ITiCSE WG (1997–2019) |
|---|---|---|---|---|
| Papers with only old-timer authors | 499 | 548 | 76 | 1 |
| Papers with only newcomer authors | 1055 | 1085 | 94 | 4 |
| Papers with mixed old-timer/newcomer authors | 626 | 1127 | 129 | 119 |
| Pct with only old-timer authors | 22.9% | 19.9% | 25.4% | 0.8% |
| Pct with only newcomer authors | 48.4% | 39.3% | 31.4% | 3.2% |
| Pct with mixed old-timer/newcomer authors | 28.7% | 40.8% | 43.1% | 96.0% |

**Table 8** Collaboration-network-based analysis of connections between old-timers and newcomers. Note that the average authors per paper is the sum of the number of authors on each paper, divided by the number of papers—that is, a measure of how many authors a typical paper has. "Giant component authors" is the number of authors who were included in the giant component. The MEIN numbers are reported in the lower part of the table

| | ITiCSE (1996–2019) | SIGCSE (1996–2019) | ICER (1996–2019) | WG (1996–2019) |
|---|---|---|---|---|
| Total authors | 3483 | 4745 | 546 | 602 |
| Pct. newcomers (after year 1) | 66.3% | 61.9% | 53.1% | 53.6% |
| Giant component size | 18.1% | 45.1% | 45.2% | 98.8% |
| Giant component authors | 325 | 659 | 87 | 350 |
| Giant component diameter | 16 | 21 | 10 | 7 |
| Number of components | 897 | 848 | 83 | 2 |
| Merge papers | 4.1% | 7.1% | 6.7% | 7.3% |
| Extend papers | 26.5% | 36.0% | 38.8% | 88.7% |
| Internal papers | 21.0% | 17.5% | 23.1% | 0.8% |
| Newcomer papers | 48.4% | 39.3% | 31.4% | 3.2% |

A collaboration-network analysis further illuminates the way in which working groups make connections between newcomers and old-timers. Results of this analysis are shown in Table 8, in comparison with ITiCSE, SIGCSE, and ICER.

As found in earlier analysis of the working groups (shown in Tables 1 and 2), the working-group authors are more closely connected through collaboration relationships than those at the other venues. The working groups' 2019 giant component includes 98.8% of its authors, compared to 95% in 2011, and over twice the percent seen at the other venues. From 1998 (the third year of the working groups) to 2019, at least 93% of authors were in the giant component. In addition, the giant component's diameter is 7, one more than before and now smaller than ICER's.

At the non-working-group venues, not only are fewer of the authors included in the giant component, those who are not in the giant component are divided into many more separate groups. The working groups have a total of two components

(including the giant component), compared to 83 at ICER, 848 at SIGCSE, and 897 at ITiCSE.

Finally, working-group newcomers are joined to the giant component very quickly. As shown by the MEIN results (Table 8), Extend papers, which always have both old-timer and newcomer authors, are much more common in the working groups than in other venues. In addition, the working-group Merge papers all contain both old-timers and newcomers. Most newcomers joined either a Merge or an Extend paper; after 1997, these always included at least one author from the giant component, so the newcomers were added to the giant component immediately.

For the authors on Newcomer papers, joining the giant component took a little longer—since, by definition, all the authors were newcomers to working groups, they had no links to the giant component. But as soon as any member of the group collaborated with anyone in the giant component on another working-group paper, that would be enough to make a difference. This one collaboration would connect the whole Newcomer component to the giant component, and connect the whole group of newcomer authors from that group to the rest of the working-group community. Except for the seven authors on one Newcomer paper—none of whom has yet returned to do a second working group—this always happened within 3 years.

In summary, working groups have more authors (and collaborators) per paper, nearly everyone is connected by collaborations, these connections are made quickly, and the distance between authors is relatively short.

### *4.3 Results: Connections Between Working Groups and Other Venues*

In order to examine the connections between the working groups and the other three venues in our dataset, we computed a list of *sociable authors*, that is, authors who have a large number of distinct collaborators [21]. Specifically, we were interested in the most sociable authors at ITiCSE, SIGCSE, and ICER during the years included in our dataset.

The results are shown in Table 9, along with the number of working groups in which each of these authors has participated. Notably, 16 of the 22 most sociable authors have participated in working groups—in fact, their number of collaborators would be even higher if working-group collaborators were included. More importantly, the table illustrates that the working groups contain people who are well-connected in the community outside of the working groups. Newcomers who collaborate with these sociable working-group authors thus also gain collaboration links to venues outside of the working groups.

**Table 9** Top 22 authors, sorted by total distinct collaborators in our dataset *outside* of working groups (column 3)

| Author | Working groups | Distinct collaborators in venue | | | | |
|---|---|---|---|---|---|---|
| | | All but WG | ITiCSE | SIGCSE | ICER | ITiCSE WG |
| Rodger Susan H. | 4 | 75 | 30 | 64 | 0 | 31 |
| Sheard Judy | 9 | 61 | 44 | 3 | 28 | 59 |
| Simon Beth | 2 | 59 | 26 | 40 | 27 | 16 |
| Simon | 7 | 53 | 20 | 4 | 41 | 47 |
| Rebelsky Samuel A. | 0 | 52 | 0 | 52 | 0 | 0 |
| Edwards Stephen H. | 4 | 51 | 9 | 32 | 25 | 36 |
| Hellas Arto | 5 | 50 | 18 | 37 | 18 | 44 |
| Zingaro Daniel | 0 | 49 | 21 | 34 | 12 | 0 |
| Lister Raymond | 4 | 47 | 17 | 15 | 32 | 34 |
| Guzdial Mark | 1 | 46 | 8 | 28 | 24 | 9 |
| Barnes Tiffany | 0 | 46 | 29 | 24 | 8 | 0 |
| Cooper Stephen | 5 | 44 | 17 | 29 | 3 | 43 |
| McCartney Robert | 6 | 44 | 20 | 26 | 28 | 43 |
| Denny Paul | 1 | 44 | 17 | 29 | 12 | 11 |
| Porter Leo | 0 | 44 | 28 | 26 | 14 | 0 |
| Cassel Lillian N. | 4 | 42 | 34 | 17 | 0 | 24 |
| Boyer Kristy Elizabeth | 0 | 42 | 4 | 40 | 0 | 0 |
| Gray Jeff | 0 | 42 | 0 | 42 | 0 | 0 |
| Sanders Kate | 7 | 38 | 16 | 23 | 28 | 49 |
| Spacco Jaime | 3 | 38 | 16 | 19 | 17 | 30 |
| Morrison Briana B. | 1 | 38 | 6 | 21 | 26 | 10 |
| Shaffer Clifford A. | 1 | 38 | 11 | 32 | 3 | 10 |

## *4.4 Results: First-Time Collaborations*

Participants in working groups, whether newcomers or old-timers, are very likely to collaborate with one or more people they have never worked with before. Table 10 gives the averages for newcomer vs. old-timer authors, and new collaborations vs old collaborations, for all of the working groups. Here we define new collaborations broadly: a new collaboration is one between two authors who have not previously collaborated in *any* of our four venues.

For most working groups the number of new collaborations is much greater than the number of old collaborations; this has been the case for 124 of the 129 Working Groups. The five exceptions were all follow-ups to earlier working groups with similar topics and largely overlapping members. For example, one of these was the fifth working group on visualization in 2000 [5]; while this sequence of groups [3–6, 19] regularly included newcomers—and the fifth group included one newcomer as well—the other six of the seven authors in the fifth group had participated in some (or all) of the previous four visualization groups.

**Table 10** New and old collaborations over all working groups

| | Number of new collaborations | Number of old collaborations |
|---|---|---|
| Average | 30.5 | 4.3 |
| Maximum | 110 | 27 |

## *4.5 Results: Follow-Up Collaborations*

To quantify how the working groups led to future collaborations, we calculated the effects of collaborations that began at working groups by considering all of the collaboration sequences in our data set. The steps in this calculation were:

1. Generate all of the collaboration sequences for every collaborating pair of authors.
2. For each working group, we collect all of the collaboration sequences that start with that group. By venue, we count all of the subsequent collaborations and distinct publications that occur in these sequences, and report the number of collaborations and the number of distinct publications for that working group.
3. For each author, we collect all of the collaboration sequences for which they are an author that start in (any) working group. By venue, we count all of the subsequent collaborations and distinct publications that occur in these sequences, and report the number of collaborations and the number of distinct publications for that author.

For example, consider the collaborations between Vicki Almstrum and Barbara Boucher Owens; their collaboration sequence contains the following five papers:

[WG 2005 5], [ITICSE 2007 85], [WG 2008 1], [ITICSE 2011 91], [SIGCSE 2012 144].[3]

Extending this collaboration example, Elizabeth Adams collaborated with both Almstrum and Owens with the collaboration sequence

[WG 2005 5], [WG 2008 1].

By venue, from the data in this example, working group 2005-5 would be credited with one subsequent working group paper and three subsequent non-working-group papers, three members having a subsequent working group paper, two members having a subsequent non-working-group paper, and three members having a subsequent paper. Working group 2008-1 is not credited with anything, as it is not the start of a sequence.

By authors, Vicki Almstrum and Barbara Boucher Owens are each credited with one subsequent working group paper and three subsequent non-working-group papers, two subsequent working group collaborators, and one subsequent non-working-group collaborator; Elizabeth Adams is credited with one subsequent

[3] For convenience, we will display papers as [venue year number].

**Table 11** Given a working group, how many members will collaborate with someone for the first time there and also publish a paper with their new collaborator in the future (on average and at a maximum). WG papers refer to papers done at future working groups, non-WG papers refer to collaborations done at future ITICSE, SIGCSE, or ICER conferences. Statistics were calculated for all but 2019 working group as it has no future in the dataset

| | First-time collaborators with subsequent | | |
|---|---|---|---|
| | WG papers | Non-WG papers | Any papers |
| WG average | 2.4 | 1.2 | 2.9 |
| WG Max | 10 | 7 | 10 |
| WGs with any | 69 (58%) | 40 (33%) | 76 (63%) |

**Table 12** Given a working group, how many future collaborations (i.e., papers) will be written by authors who first collaborate at that working group (average and maximum). WG papers refer to reports from future WGs, non-WG papers refer to papers at future ITICSE, SIGCSE, or ICER conferences. Statistics were calculated for all but 2019 working groups as they have no future opportunities in the dataset

| | | Subsequent papers per WG | |
|---|---|---|---|
| | New collaborations | WG papers | Non-WG papers |
| WG average | 30.5 | 1.3 | 1.3 |
| WG Max | 110 | 9 | 32 |
| WGs with any | | 69 (58%) | 40 (33%) |

working group paper and two subsequent working group collaborators. Note that we are counting distinct collaborators, not total collaborations.

First, consider the contribution of the working groups to future publications. The results of this analysis (step 2) are given in Tables 11 and 12. Table 11 expresses the working-groups' contribution in terms of its effect on working-group members: on average, a working group enables 2.9 of its members to collaborate for the first time at that working group *and* then collaborate on a later paper. Of those, 2.4 collaborate on later working-group paper(s) and 1.2 collaborate on later paper(s) outside the working groups with their new collaborator(s). Not surprisingly, some do both, so the total number of working-group members with follow-up papers is less than the sum of those with working-group and those with non-working-group follow-up papers.

Table 12 expresses the working groups' contribution in terms of papers produced. On average, each working-group is followed by 1.3 distinct working group papers and 1.3 distinct non-working-group papers where a pair of authors who first collaborated at a working group are among the authors. Fifty-eight percent of all working groups have at least one follow-up that is a working-group paper, and 33% have at least one follow-up paper in SIGCSE, ITiCSE, or ICER.

Next, consider what a typical working-group author could expect (step 3). The results of this analysis are given in Table 13. On average, among all the authors who have been in working groups, each author has 0.54 future working-group papers and 0.58 future non-working-group papers with someone they first collaborated with in a working group. Similarly, each author has 1.24 future distinct collaborators

**Table 13** What a working-group author could expect in terms of future collaborations with authors with whom they first collaborated at some working group. a working group ("WG"). "WG and Non-WG papers" are based on counts of distinct papers; "collaborators" is based on distinct collaborators. New authors from 2019 working groups are not included, as no conferences in the data set followed it, so the total number of authors used here is 578

| | WG papers | Non-WG papers | WG collaborators | Non-WG collaborators |
|---|---|---|---|---|
| Average | 0.54 | 0.58 | 1.24 | 0.42 |
| Max | 11 | 25 | 19 | 8 |
| Authors with >0 | 166 | 106 | 166 | 106 |

on working-group papers and 0.42 future distinct non-working-group collaborators with whom they first collaborated at a working group.

Finally, we can extend this analysis to consider all of the collaboration sequences across the four venues. In Table 14, we look at all of the collaboration sequences. Papers that start collaborations (*first collaboration papers (FC)*), are any papers that are the first paper in the collaboration sequence of two authors; that is, these are the papers where these two authors first collaborated. We combine these sequences by FC paper: the successors of FC paper $p$ are all the distinct papers that are successors in any collaboration sequence starting with $p$. Roughly speaking, these are the papers that follow from all of the first-time collaborations in paper $p$.

As shown in Table 14, the average number of successors is highest for working group papers: 2.5, compared to 1.4 for ICER, 0.6 for ITiCSE and 0.7 for SIGCSE. The proportion of FC papers that have successors is also higher than the other venues: 64% for the working groups, compared to 28% for ITiCSE, 29% for SIGCSE, and 45% for ICER.

There are also differences in the percentage of papers that include first collaborations. Nearly all (99%) of the working-group papers include first collaborations, as opposed to 81% of the multi-author ITiCSE papers, 85% of the multi-author SIGCSE papers, and 67% of the multi-author ICER papers. The non-working-group numbers would be even lower if we included the single-author papers that cannot be a first collaboration.

Overall, these results indicate that the number of papers "generated" by the average working group is much higher than the average multi-author paper at the other venues.

## *4.6 Results: Connections with Other Countries*

One of the most distinctive features of the working groups is the opportunity they offer for multi-national collaboration. Out of the 129 working groups, 96.1% (124) were multi-national. On average, each working group included representatives from four different countries, with a maximum of eight.

**Table 14** Papers that start collaborations (First Collaborations (FC)) by venue and their successors in any venue. Average and maximum values are based on distinct papers. These numbers were not calculated for the papers whose first collaboration was in 2019 working group, as no conferences in the data set followed it

| | ITiCSE (1996–2019) | SIGCSE (1996–2019) | ICER (2005–2019) | ITICSE WG (1996–2018) |
|---|---|---|---|---|
| Average successors per FC paper | 0.64 | 0.70 | 1.37 | 2.53 |
| Maximum successors per FC paper | 20 | 34 | 18 | 39 |
| Number of FC papers | 1291 | 1861 | 188 | 119 |
| FC papers with >0 successors | 357 | 536 | 85 | 76 |
| Number of multi-author papers | 1596 | 2195 | 315 | 120 |
| FC papers/multi-author papers | 81% | 85% | 67% | 99% |
| FC papers with >0 successors/FC papers | 28% | 29% | 45% | 64% |

A similarly high percentage of the working-group authors, 98.5% (593 of 602) took part in one of the multi-national collaborations. The five single-nation working groups included a total of 30 distinct individuals, but 21 of those participated in multi-national working groups in a different year.

While individuals from the United States and the United Kingdom participated in the largest number of working groups (119 and 70, respectively), 45 different countries were represented. There were participants from North and South America, Europe, Africa, Asia, and Australasia. Despite the fact that the conference is usually held in Europe, both Australia and New Zealand were among the top five countries, with 38 working groups containing Australians and 41, New Zealanders. Mexico, Central America, and South America were represented in a total of only nine working groups, and Africa only in one. Notably, all of the nine working groups with members from Mexico, Central America, and/or South America met in or after 2014, the year that it was announced that ITiCSE 2016 would be held in Peru.

The average number of countries per component (after the first year) ranged from 8.5 to 23.5, as shown in Fig. 3. By the third year, 1998, all working-group participants up to and including that year were part of the same component (the giant component). After that, the number of components fluctuated between one and four, with new components being added by each Newcomer working group and subtracted by each Merge group (since two different earlier components are joined together). Most newcomers collaborated with old-timers, thus joining the giant component immediately. The other components were all very small relative to the giant component, only including members from 2 to 4 countries each.

The structure of the working-group collaboration network has a noticeable effect on the average number of countries per component, however. To see this effect, consider the change in Fig. 3 between 2009 and 2010. In 2009, there was a single component involving 25 countries. The next year, a Newcomer working group caused a new component to be added. There were then two components, one with 25 countries and one with 2, so the average plummeted to 13.5.

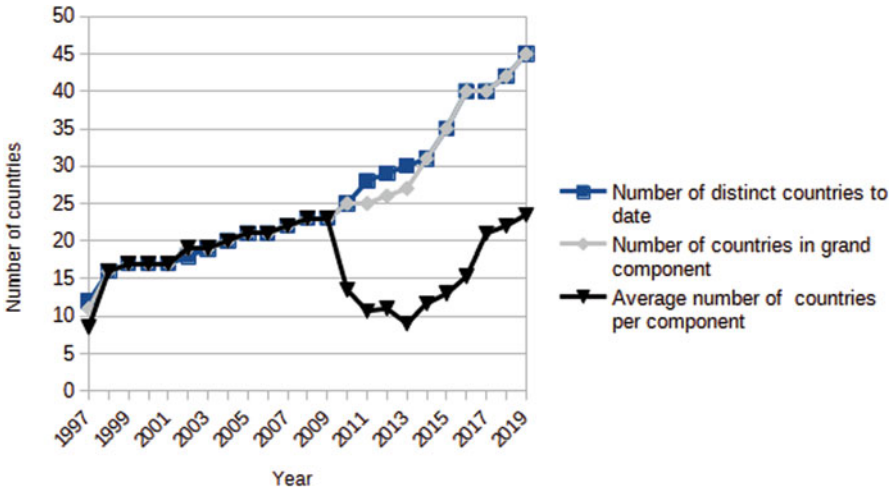


**Fig. 3** The average number of countries per component in the collaboration graph, compared with the number of countries in the giant component, which more closely tracks the total number of countries involved in working groups up to that point

Thus, while large, the average number of countries per component may actually understate the multi-national experience provided by the working groups. The two other lines in the graph, the total number of countries represented to date and the number of countries in the giant component, provide context about the component to which the large majority of the working-group members belong. Arguably, this context better captures the experience of most working-group members.

## 5 Discussion

In this section, we place the results in the context of related work and discuss some implications of the results.

### *5.1 Discussion: Connections Between Newcomers and Old-Timers*

The updated analysis of the working groups to 2019 confirmed the results of previous analysis through 2016 [13]: more than half of the participants are newcomers, on average, and these newcomers are very quickly and closely connected to the working-group community, as represented by the very large giant component of the

collaboration graph, which despite including nearly all of the participants, has a very small diameter.

In addition, we found that all working-group participants were likely to collaborate with people they had *not* worked with previously. Our analysis showed a very high percentage of new collaborations, regardless of whether the collaborating authors were old-timers or newcomers.

## 5.2 Discussion: Connections Between Working Groups and Other Venues

We identified a list of "sociable authors", using Simon's terminology [21]. This list, given in Table 9, includes those authors who have the most collaborators across all the venues we examined. Of these sociable authors, 16 out of 22 also participated in working groups (not surprising, given their affinity for collaboration). Thus, other authors who collaborate with them are becoming connected, not just to the working-group community, but to the other three venues as well.

In addition, the working groups provide a partial solution to the concern for "influential authors" and "dominant institutions" raised by Apiola et al. [1]. A number of the sociable authors listed in Table 9 and/or the institutions with which they are affiliated are also included in Apiola et al.'s lists of influential authors and dominant institutions. By participating in working groups, they are making themselves available to work with new collaborators, often including newcomers to the community.

## 5.3 Discussion: Follow-Up Collaborations

The results for follow-up collaborations all suggest that new collaborations at working groups lead to future collaborations, both in working groups and elsewhere. Looking at future collaborations by working groups, we see that on average around three of the newly-collaborating members publish with their collaborators in the future, and around three future papers include collaborators who first collaborated in that working group. In terms of new working group authors, on average each will publish about one future paper with their new working group collaborators.

The above numbers may understate the actual effects on future collaborations. For one, the only collaborations we measured were publications at these four venues; there are other venues, and other forms of collaboration, such as working together on conference committees. Additionally, it should be noted that the future collaborators we measured are a minority of the authors: in Table 13 for example, only 166 of 578 (29%) have future working group collaborators, and 106 of 578 (18%) have future non-working-group collaborators. (The 578 authors do not

include the members of 2019 working groups; as those are the last publications in our dataset, they could not have any follow-up publications.) If we were to compute our averages for the authors that *have* future collaborators, the averages would be much higher: 1.88 future working group papers instead of 0.54; and 4.31 future working group collaborators instead of 1.24. Similarly if we only consider the authors that have future non-working-group collaborators, the averages would be 3.14 future working group papers instead of 0.58, and 2.88 future working group collaborators instead of 0.42.

The examination of collaboration sequences suggest that working group papers lead to more future publications than papers at the other venues. On average the number of future publications is larger for the working groups that involve first collaborations, and a greater percentage of the working group papers include first collaborations among authors. An interesting side note is that the proportion of first-collaboration papers at ICER is less than those at ITiCSE and SIGCSE. This may be due to the ICER data starting 9 years later (with first collaborations having taken place at another venue), but it may be worth examining more closely.

### *5.4 Discussion: Connections with Other Countries*

The working groups offer an extraordinary opportunity to learn from and collaborate with researchers from other countries. As noted above, Zhang et al. [24] found that the average number of countries per component was approximately 1 for ICER, ITiCSE, and SIGCSE, indicating that the large majority of authors collaborate with people in their own country. By contrast, the average for the working-group graph components varies from 8.5 to 23.5 after the first year of the conference.

Like Apiola et al. [1], we found that there were relatively few representatives from Asia, Africa, or South America in our dataset. On the other hand, those who do participate, along with other newcomers, have the opportunity to work with experienced computing education researchers from some of the "dominant" institutions in the field. Thus, the working groups could be seen as a step towards decreasing inequality.

In addition, one encouraging sign is that there began to be participants from South America just before ITiCSE was held in Peru and that participation has continued since, though still at a fairly low level. Similarly, at ICER, which regularly rotates between Europe, North America, and Australasia, the Australasian attendance usually increases when the conference is located in Australasia. The working groups now being held at SIGCSE's new CompEd conference will also likely lead to an increased participation from Asia. Rather than attempting to increase the diversity at the SIGCSE Symposium, as suggested in Becker et al. [2], locating conferences in Africa and South America, particularly if they include opportunities for multi-national collaboration like the working groups, might be more effective in increasing participation from those regions. Future work might also investigate the geographical representation at LaTiCE, a relatively new computing education

conference that has been held in Mumbai, Hong Kong, Taiwan, and Auckland (http://www.latice-conference.org/index.html).

## 6 Limitations, Threats to Validity, and Future Work

There are a number of limitations to this work. Most importantly, our conclusions are limited by the venues we examined. Given the nature of academic conferences and journals, we expect that in other venues, the proportion of collaborations between newcomers and old-timers, between people who have never collaborated before, and between individuals from different countries, would still be strikingly different from the working groups. This conjecture remains to be tested by future work.

While the working groups are exceptionally multi-national, it is possible that other venues have (or will have) greater representation from areas such as Africa, Asia, and South America. The particular countries involved in collaborations are likely affected by the location of the conference in question. Just as the working groups added representation from Mexico, Central America, and South America around the time that ITiCSE was held in Peru, and the Australasian representation at ICER generally increases every 4 years when the conference is held in Australasia, it is likely that in general, more participants will submit to and attend a conference that is conveniently located. Future work might examine ACE, the Australasian Computing Education conference; the relatively new LaTiCE conference, which has been held in Mumbai, Taiwan, Hong Kong, and Auckland; and the working groups held in association with SIGCSE's new CompEd conference, located in Asia. It would also be very interesting to see the results of a computing-education conference located in Africa.

With regard to follow-up collaborations, the limitations in our dataset pose a threat to validity. SIGCSE, ITiCSE, and ICER do include a large segment of the computing education publications. It is possible, however, that two authors whose first collaboration in our dataset occurred at a working group, had previously published together in a venue outside our dataset. If so, it would lead to over-counting the number of follow-up collaborations. On the other hand, it is definitely the case that there have been follow-up publications in venues outside our dataset by authors who did work together for the first time at a working group. (See, for example, [14, 15]). In these cases, the follow-up publications have been under-counted.

There may be other confounding factors. For example, collaborators may have known each other previously without having any previous collaborations. For example, the four organizers of the Leeds working group had met at the Bootstrapping Computing Education workshop in 2002–2003. Three of the other participants had attended the related Scaffolding Computing Education workshop in 2003–2004 [7, 8]. While they had all met, only two of the seven had any published collaborations at the time of the working group.

It is difficult to measure the synergy between such events, but it is quite possible that it was the combination of the Bootstrapping/Scaffolding workshops and the Leeds working group, rather than either alone, that led to follow-up collaborations between these researchers. Similarly, researchers who have interacted informally at conferences for years, might find that collaborating in a working group led to later collaborations.

Another threat to validity is posed by the data cleaning. Until every author has some kind of unique ID, identifying duplicate authors will remain a challenge. We likely missed some duplicate authors, but probably not enough to affect the overall results.

A final limitation is due to the methodology: bibliometrics do not capture the working-group members' experience of participation. It would be very interesting to see a qualitative analysis of working group organizers and participants.

## 7 Conclusions

Based on this bibliometric analysis, the Working Groups provide a rare opportunity to collaborate with a group of international peers in computing education. Working-group members quickly become part of a community, closely connected to the large majority of other working-group members, past and present. They also have the opportunity to work with people they have never worked with before, people from other institutions and a wide variety of different countries, and to work beside and learn from widely published researchers in the field. They also may have the opportunity for follow-up collaborations with some of their new collaborators; most working groups lead to at least some such collaborations.

Bibliometric data is valuable, but it cannot convey the individual experiences of participating in a working group. There are many less visible results, such as learning about research methodology and tools, meeting people who can later provide letters of recommendation, and even simply seeing familiar faces at computing education conferences. In short, those who participate in working groups become part of a community.

Interesting future work might include a qualitative investigation of the ways in which organizers and participants experienced the working groups. Meanwhile, however, Lauri Malmi has captured the experience of many in a column in the *SIGCSE Bulletin* [12, p. 27]:

> When I entered the international computing education community early in 2000, the path was quite different. I did not start by attending SIGCSE Symposium with some 1000 participants. Instead, I participated in a working group of the Innovation and Technology in Computer Science Education (ITiCSE) conference, where I worked five days intensively with a dozen international colleagues, in addition to undertaking some preparatory work before the conference. The experience was totally different. I got to know them, we worked together, and I was contributing as a member of the group. Afterwards I have attended and twice cochaired five such working groups. Meeting the same people in other conferences later is like meeting old fellows. As a bonus I have learned a lot from different countries and

practices in their academic (and also non-academic) life. Based on this experience I have strongly recommended participation in working groups for my own PhD students, even when the topic of the group may not fit into the scope of their (future) thesis. Joining the international community and getting to know people are important steps in their scientific careers. Most of my students have participated in, and, in the postdoc phase, some of them have also proposed and even co-chaired their own working groups, thus taking a more senior role in the community.

## References

1. Apiola M, Saqr M, López-Pernas S, Tedre M (2022) Computing education research compiled: Keyword trends, building blocks, creators, and dissemination. IEEE Access 10:27041–27068, DOI https://doi.org/10.1109/ACCESS.2022.3157609
2. Becker BA, Settle A, Luxton-Reilly A, Morrison BB, Laxer C (2021) Expanding opportunities: Assessing and addressing geographic diversity at the SIGCSE technical symposium. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, p 281–287, URL https://doi.org/10.1145/3408877.3432448
3. Bergin J, Brodie K, Patiño Martínez M, McNally M, Naps T, Rodger S, Wilson J, Goldweber M, Khuri S, Jiménez-Peris R (1996) An overview of visualization: Its use and design: Report of the working group in visualization. SIGCSE Bull 28(SI):192–200,URL http://doi.acm.org/10.1145/237477.237647
4. Bergin J, Kumar A, Proulx VK, McNally M, Faulstich Brady A, Mutchler D, Hartley S, Rasala R, Kelemen C, Ross R, Klassner F (1999) Resources for next generation introductory CS courses: Report of the ITiCSE'99 working group on resources for the next generation CS 1 course. SIGCSE Bull 31(4):101–105, URL http://doi.acm.org/10.1145/349522.571916
5. Bergin J, Kelemen C, McNally M, Naps T, Goldweber M, Power C, Hartley S (2001) Non-programming resources for an introduction to CS: A collection of resources for the first courses in computer science. SIGCSE Bull 33(2):89–100, URL http://doi.acm.org/10.1145/571922.571963
6. Bland CG, Hartley SJ, Holliday MA, Lawhead PB, Lewis J, McNally MF, Nevison CH, Ng C, Pothering GJ, Teräsvirta T (1998) Java resources for computer science instruction. SIGCSE Bull 30(4):18–38, URL http://doi.acm.org/10.1145/306286.306324, Chairman-Bergin, Joseph and Chairman-Naps, Thomas L.
7. Fincher S, Tenenberg J (2006) Using Theory to Inform Capacity-Building: Bootstrapping communities of practice in computer science education research. Journal of Engineering Education 95(4):265–277, DOI https://doi.org/10.1002/j.2168-9830.2006.tb00902.x
8. Fincher S, Lister R, Clear T, Robins A, Tenenberg J, Petre M (2005) Multi-Institutional, Multi-National Studies in CSEd Research: Some design considerations and trade-offs. In: Proceedings of the First International Workshop on Computing Education Research, pp 111–121, DOI https://doi.org/10.1145/1089786.1089797
9. Hamer J, Cutts Q, Jackova J, Luxton-Reilly A, McCartney R, Purchase H, Riedesel C, Saeli M, Sanders K, Sheard J (2008) Contributing student pedagogy. SIGCSE Bull 40(4):194–212, URL https://doi.org/10.1145/1473195.1473242
10. Lister R, Adams ES, Fitzgerald S, Fone W, Hamer J, Lindholm M, McCartney R, Moström JE, Sanders K, Seppälä O, Simon B, Thomas L (2004) A multi-national study of reading and tracing skills in novice programmers. SIGCSE Bull 36(4):119–150, URL https://doi.org/10.1145/1041624.1041673
11. Lunn S, Marques Samary M, Peterfreund A (2021) Where is computer science education research happening? In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, pp 288–294, URL https://doi.org/10.1145/3408877.3432375

12. Malmi L (2015) Entering the research community. ACM Inroads 6(4):27–28, URL https://doi.org/10.1145/2834125
13. McCartney R, Sanders K (2018) ITiCSE working groups and collaboration in the computing education community. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '18, p 332–337, URL https://doi.org/10.1145/3197091.3197143
14. McCartney R, Moström JE, Sanders K, Seppälä O (2004) Questions, annotations, and institutions: observations from a study of novice programmers. In: Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education, pp 11–19
15. McCartney R, Moström JE, Sanders K, Seppälä O (2005) Take note: the effectiveness of novice programmers' annotations on examinations. Informatics in Education 4:69–86
16. McCracken M, Almstrum V, Diaz D, Guzdial M, Hagan D, Kolikant YBD, Laxer C, Thomas L, Utting I, Wilusz T (2001) A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. In: ITiCSE-WGR '01, pp 125–180, URL http://doi.acm.org/10.1145/572133.572137
17. Miró Julià J, López D, Alberich R (2012) Data from 2012 ICER paper. URL http://bioinfo.uib.es/~recerca/Colab/FinalDataSet/, accessed January 21, 2018
18. Miró Julià J, López D, Alberich R (2012) Education and Research: Evidence of a dual life. In: Proceedings of the Ninth Annual International Conference on International Computing Education Research, ICER '12, pp 17–22, URL http://doi.acm.org/10.1145/2361276.2361281
19. Naps T, Bergin J, Jiménez-Peris R, McNally MF, Patiño Martínez M, Proulx VK, Tarhio J (1997) Using the www as the delivery mechanism for interactive, visualization-based instructional modules (report of the ITiCSE '97 working group on visualization). In: The Supplemental Proceedings of the Conference on Integrating Technology into Computer Science Education: Working Group Reports and Supplemental Proceedings, ACM, New York, NY, USA, ITiCSE-WGR '97, pp 13–26, URL http://doi.acm.org/10.1145/266057.266062
20. Simon (2016) The Koli Calling Community. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli Calling '16, pp 101–109, URL http://doi.acm.org/10.1145/2999541.2999562
21. Simon (2016) A picture of the growing ICER community. In: Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16, pp 153–159, URL http://doi.acm.org/10.1145/2960310.2960323
22. Simon (2020) Twenty-four years of ITiCSE authors. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, pp 205–211, URL https://doi.org/10.1145/3341525.3387387
23. Simon, Sheard J (2020) Twenty-four years of ITiCSE papers. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, pp 5–11, URL https://doi.org/10.1145/3341525.3387407
24. Zhang J, Luxton-Reilly A, Denny P, Whalley J (2021) Scientific collaboration network analysis for computing education conferences. In: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '21, pp 582–588, URL https://doi.org/10.1145/3430665.3456385

# A Case Study: The Uppsala Computing Education Research Group (UpCERG)


**Mats Daniels, Anders Berglund, and Arnold Pears**

## 1 Introduction

We will in this chapter describe the journey of the Uppsala Computing Education Research Group, UpCERG (https://www.it.uu.se/research/group/upcerg), from when it started in the mid-90s till today. This journey began with two teachers wanting to have a better scientific foundation for conducting and developing computing education leading to the situation today when Computing Education Research (CER) is an established research area at Uppsala University with a research program and full professors. There are several ways to tell this story. We will present a roughly chronological outline of how the research group developed in terms of the research areas addressed and the methodologies and theories used. The characterization of research activities in UpCERG is that they are theoretically and methodologically broad, using multiple research approaches and covering many research topics. The chapter will first give a personal view of the development of UpCERG, followed by seeing UpCERG from a theory perspective. The latter part provides a more objective perspective and illustrates how theory has been a crucial part of the establishment of UpCERG. This part draws from a paper presented at the International STEM Education conference iSTEM-Ed [5].

M. Daniels (✉) · A. Berglund
Uppsala University, Uppsala, Sweden
e-mail: mats.daniels@it.uu.se; anders.berglund@it.uu.se

A. Pears
KTH Royal Institute of Technology, Stockholm, Sweden
e-mail: pears@kth.se



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_12

## 2 UpCERG: A Personal View

We start the case study by presenting UpCERG from the point of view of one of its founders, Mats Daniels, partly drawing from chapter two of his Ph.D. thesis [12]. This section provides an inside and personal view of the development of UpCERG. The first part covers the time leading to the Ph.D. defense, followed by the period to the present when UpCERG has a research program and is an established subunit at the department of Information Technology at Uppsala University.

### *2.1 The First 15 Years: A Story of Frustration Fostering Creativity*

There are many ways to start a story, and one is perhaps to observe that I started my Ph.D. studies a little over 40 years ago, on April 9, 1981. The first part of my life as a Ph.D. student related to traditional computer science in the form of using formal methods to describe and analyze communication protocols and computer hardware. It was, as such, not essential for the background of UpCERG, even though teaching and discussing education, both content and form, during this period had a strong influence. This first part of my academic career included earning a licentiate degree in 1985, then working as a lecturer, and spending the year 1989/1990 at La Trobe University in Melbourne, Australia, as a visiting professor.

The part relevant to the formation of UpCERG started when I became director of undergraduate studies in 1991. My work in UpCERG drew on research and experience from a journey that started with being frustrated about the lack of sound scientific foundations for decisions at degree program boards. This led to searching for relevant Computing Education Research to learn from and collaborate with. An essential component on this journey was leading the RUNESTONE project. This project was a stepping stone on forming theories related to Open-Ended Group Projects (OEGP) in computing education based on action research on the development and assessment of professional competencies in the IT in Society course (ITiS).

#### 2.1.1 Frustration

Working in the education field was often frustrating, but at the same time also highly inspiring. This contradiction became even more apparent when I got appointed to some of the boards of studies. Thus, I gained first-hand experience making decisions about the content and running of degree programs. Decisions made in these boards of studies significantly impacted how education was set up, and on numerous occasions, decisions were made without any scholarly evidence.

Typical issues were related to courses, e.g., inclusion or exclusion, their sequence, the needed prerequisites, the size, and even the way they were taught. There were also decisions related to degree program goals, how to follow up on students that achieved a degree or dropped out, and how to recruit students. Many of these issues were decided in rather non-constructive discussions during long meetings.

There was also frustration regarding my shortcomings in my role as an educator, especially after becoming director of studies at the department. An essential part of the latter role was to function as support for other educators and plan the running of the courses the department was responsible for. I felt a need for a scientifically sound foundation for how to act in those roles.

### 2.1.2 Computing Education Research

This frustration led to a search for answers and people who knew more about the issues I had encountered in board meetings and in my roles as educator and director of studies at the department. The time is now the mid-nineties, and we had Vicki Almstrum as a guest lecturer at the department. Through Almstrum, I contacted Nell Dale and her group at the University of Texas at Austin, USA, which was the only group conducting computing education research I could find at that time.

Further searching revealed groups at Open University (Marian Petre), the University of Kent at Canterbury (Sally Fincher) both in the UK, and Monash University (Dianne Hagan) in Australia. We formed a loose alliance called Computer Science Education Research Groups International (CSERGI) to discuss and conduct research building competence in the area. CSERGI ran a set of workshops, one in 1999 dedicated to exploring and defining the research area. This collaboration sparked more focused research in Uppsala, and a new research area was born. Interest in the area grew also with members joining from other divisions, and the recruitment of Arnold Pears from La Trobe University in Australia in 2000. In 2005 Anders Berglund defended the first PhD in the area. The area gathered momentum over the next 5 years, and by 2010, five Ph.D. theses had been defended in this research area at Uppsala University; Berglund [3]; Boustedt [8]; Cajander [9]; Eckerdal [15] and Wiggberg [47].

The research group at the department was first named Uppsala Computer Science Education Research Group but later changed to Uppsala Computing Education Research Group (UpCERG). By 2010 the group had grown to include members from three of the divisions in the Department of Information Technology; Computer Systems, Scientific Computing, and Human-Computer Interaction.

### 2.1.3 International Projects

In the early 2000s there were few, if any, sources from which to apply for research funding for Computing Education Research. However, the national council for

the renewal of higher education ("Rådet för högre utbildning") did support large development projects and attendance at conferences in computing education. In 1997 we successfully obtained funding for two three-year projects; Runestone [14] and Espresso [4, 6] My project was the Runestone project, or if speaking Swedish, "Runstenprojektet", in which we established an international student project collaboration between Uppsala University and Grand Valley State University in Michigan, USA.

Runestone was relatively well-financed and is, together with the Expresso project, the start of a real commitment to research in UpCERG. The importance of Runestone as a focus for research is evident from the three Ph.D. theses based on studying aspects of Runestone. These theses were done by (1) Anders Berglund at Uppsala University (Learning computer systems in a distributed project course The what, why, how and where [3]), (2) Mary Last at the University of Texas at Austin, USA (Investigating the Group Development Process in Virtual Student Software Project Teams [25]), and (3) Martha Hause at the UK Open University (Software development performance in remote student teams in international computer science collaboration [21]).

Several aspects of Runestone were interesting, but my particular interest was the issues related to international collaboration. This interest derived partly from an enriching year as an exchange student at Case Western Reserve University in Cleveland, USA, 1979/1980. I wanted to find ways in which more than just a few students could have similar experiences. Runestone provided many opportunities to reflect on how similar experiences could be achieved by adding an international component to our local education setting.

I also started a smaller international collaboration, the NZ project, with Auckland University of Technology, New Zealand, in 1998, after meeting Tony Clear at a conference in Dublin. It was intended to be the first taste of international collaboration for the IT engineering students as a part of their introductory course. This collaboration was prominent in Tony Clear's master thesis 2000 and his Ph.D. thesis [11]. A significant spin-off from my cooperation with Clear was that two students from Uppsala University completed their master's theses [19] in Auckland with Clear as supervisor. These students participated in the NZ project, the Runestone project, and the IT in Society course sequence.

### 2.1.4 Open-Ended Group Projects

Runestone, and project semesters, are examples of courses that were rewarding for students, but there were questions about their educational value. This situation was in the back of my mind when I met two colleagues from the UK, Xristine Faulkner and Ian Newman, at a conference in 2001. It turned out that we had similar frustrations, and we ended up having long discussions about our experiences with this type of course. The more we talked, the more we felt we had a lot in common, both in terms of what we did in our courses units and reactions from students and, especially, education coordinators. We saw potential in how we organized project

course units but also obstacles. It soon became clear to us that we more or less told the same story.

What we talked about was exposing the students to a real problem that had no obvious solution and preferably encompassed aspects from many different areas. In short, an open-ended problem. The settings we discussed all included students working in groups and where the problem they addressed was impossible for one individual to deal with alone. Our involvement as educators was limited to offering advice and being there for discussions about the students' progress, with an emphasis on observing the quality of how they worked rather than focusing on how good the solution to the problem turned out to be. Another common denominator was that we saw and accepted that the students could assume very different roles in the projects as long as there was a real collaboration in a group.

We realized that we needed a name for what we discussed and coined the term Open-Ended Group Projects (OEGP). Faulkner later earned a Ph.D. [17] at her university, London South Bank University, UK, mainly based on work with OEGP.

### 2.1.5 The IT in Society Course Unit

My work focused on the IT in Society course unit. This course unit was introduced into the IT engineering degree program as a response to industry feedback collected using questionnaires and meetings before the commencement of the degree program in 1995. This input emphasized that scaffolding the development of teamwork and communication skills was a priority for our industry stakeholders.

Running this course has been a challenge every year since 1998, and it has been a quite inspiring challenge. The development of vocabulary and theories related to open-ended group projects was a vital component in meeting this yearly challenge. The open-ended group project idea suited this course well, but the unique content (e.g., societal aspects) added complexity to setting up a productive learning environment. Such a setting was confusing for the students, since they only had familiarity with highly technical preparation in their other degree courses. Much effort over the years concerns devising appropriate scaffolding to support the students without compromising the underlying ideas behind the open-ended group project concept. My thesis summarizes much of that research.

### 2.1.6 Action Research

The way I worked with developing the IT in Society course evolved in parallel with the development of an educational research framework [13, 36]. This combination of development and research led to a model for scholarly educational development and research, a model that was combined with the action research methodology. The action research cycle fitted the yearly occurrence of the IT in Society course, and the methodology provided a suitable structure for dealing with the research-based development of a complex learning environment.

## 2.2 *The Following Decade(+): A Story of Struggles and Consolidation*

A fairy tale often ends with "Then they lived happily ever after.". The "happy ever after" is not the case, and I think it would have been rather dull if there were no challenges or disappointments to deal with.

### 2.2.1 Point of Departure and Continues Work

One thing standing out, looking back at the story, was that most of what I had been working with up until my Ph.D. fell under the professional competence hat. Another reflection was that there had been an integrated process between conducting research-based development and developing a research framework. Professional competence has continued to be a strong interest for me, with working on providing a better theoretical understanding of the concept and especially how to construct educational settings to provide students opportunities to deal with complex real-world issues holistically.

### 2.2.2 Struggles and Consolidation

Interest and devotion can take you far, but it is an uphill struggle without proper funding. Being an interdisciplinary discipline, Computing Education Research (CER) has meant that there are no apparent sources to send grant applications. There is also the dilemma of not having a clear home. For instance, at Uppsala University, there is a faculty of Educational Sciences, but much of the existing Discipline Based Education Research has traditionally been done in the disciplines. UpCERG started at the Department of Computer Systems (from 1999, the Department of Information Technology) without regular research funding. Our work was supported by us receiving development and research grants but also done on a non-funded basis.

Establishing CER as a research area with faculty funding was on my agenda for many years, especially after my dissertation. There were many disappointments, but our results eventually led to establishing CER as one of two areas, the other being AI, that the department pushed for in an internal evaluation exercise at the Faculty of Technology and Natural Sciences in 2018. This push led to the establishment of CER as a new research program, including a stable research budget, in 2020.

Perhaps less of a struggle was to advance my academic status. It had taken 30 years to get my Ph.D., but 2 years later, I became Docent, and in 2017 I was appointed full professor. One year after Arnold Pears became the first full professor in CER at Uppsala University. Another struggle has been the place in the organization. Members of UpCERG belonged to four of the five divisions at the Department of Information Technology. This diversity had its advantages, as

members were close to the particular aspect of the computing discipline they were interested in, but it had apparent drawbacks regarding visibility and funding. From 2022 the members of UpCERG are in one unit at one of the divisions.

## 3 UpCERG: Seen Through a Theory Perspective

The previous section gives a rather personal view of the establishment of UpCERG. This story is colored by the narrator but provides one strand of what has formed UpCERG. In this section, we will present the role theory has played in the development of UpCERG. We will briefly discuss theory in CER in general, before getting to how theory became a living part of UpCERG. More in-depth coverage of theory in CER is provided elsewhere in this book, especially in chapter "Theory and Approaches to Computing Education Research".

### *3.1 Why Discussing Theory in CER?*

The term theory is a multifaceted and complex concept. This is a section with a focus on the use of theory in Computing Education Research (CER), and not on theory per se. This quote from Klette in Norwegian Educational Research towards 2020—UTDANNING2020, [24, pp. 3–4], on the role of theory in educational research provides a summary of what theory can mean.

> Simply speaking, theory refers to a particular kind of explanation. Leedy and Ormrod [27, p. 4] state: "A theory is an organized body of concepts and principles intended to explain a particular phenomenon". Thus, theories explain how and why something functions the way it does [23, p.7]. As pointed out by Boss, Doherty, LaRossa, Schumm, and Steinmetz [Boss et al., 2008 [8], p. 20]: "Theorizing is the process of systematically formulating and organizing ideas to understand a particular phenomenon. Thus, a theory is the set of interconnected ideas that emerge from this process". Following McMillan and Schumacher [33], a theory can develop scientific knowledge congruent with the following criteria: first, provide simple explanation about the observed relations regarding their relation to a phenomenon; second, be consistent with an already founded body of knowledge and the observed relations; third, provide a device for verification and revision; and fourth, stimulate further research in areas in need of investigation.

Accepting this discussion as a perspective on what a theory is, we can now focus on the role of theory and its applications in CER. Here we find inspiration in Suppes' pivotal article from 1974 [38] and particularly in section 1, "Why theory?". His first argument is an argument by analogy from the more mature sciences (i.e., mathematics, physics), which can support the need for theory in other sciences, among them educational research. The second argument refers to the reorganization of experience, where Suppes offers the law of inertia, replacing Aristotelian physics as his core example. Another example from CER is to discuss the decline of teaching by transfer as a dominating theory of teaching and learning. Suppes presents the

reorganization of experience as his third argument. He argues that what can be found under the surface could be more complex than what can be seen at first sight. Theory offers broader explanations of a phenomenon and thus supports seeing and understanding an underlying complexity. A clear example could be the replacement of the Ptolemaic worldview with the more theoretically sound helio-centric. His final argument is that bare empiricism would be trivial. This shortcoming should be evident for a teacher if unable to refer to theory when explaining something to his or her students.

Certainly, Suppes is not alone in arguing for the usefulness of being theory-aware in educational research, but his arguments are clearly described and consistent over time. Tenenberg and Malmi are editors for two special issues on Theory in computing education research for the journal Transactions on computing education (TOCE). In their editorial [41], they discuss the role of theory in CER. They point out that questions such as: what can be borrowed from other disciplines, how to build theory within CER, how to use theory appropriately, how to combine theory, whether it is necessary to use theory in reporting research or instructional designs, and what we take theory to be, have raised interest in the CER community. Examples are literature surveys, such as [30] on theoretical underpinnings of CER and [28] analyzing ICER papers. Tenenberg and Malmi also point out that journals and conferences often explicitly ask for papers with a clear theoretical foundation. They also share their experiences as editors and program chairs with reviewers finding it challenging to evaluate what is an appropriate use of theory.

That it is a challenge for researchers to select suitable theories is evident in a paper by Szabo and Sheard [39], where they investigate the use of learning theories in CER. They have found that many learning theories are suitable for addressing a given learning phenomenon and that integrating several theories can better explain learning. Similarly, Tedre and Pajunen [40] also observe that there are a plethora of uses of theories in CER, and they focus on the lack of consensus regarding the concept of "theory". They discuss the use and non-use of learning theories from several perspectives, especially the different goals for using theories. They propose a model-based view to avoid the "baggage" associated with the theory concept and that the philosophy of engineering would be more appropriate than the philosophy of (natural) science. Tedre and Pajunen argue that the CER community should work towards its own paradigm, including defining the relationship with theory. They start out their discussion by addressing the maturity of CER, which Malmi et al. [29] also address in their recent work. Their 2022 paper is a survey of papers published at three major CER venues over the time period 2005–2020, investigating the use of domain-specific theories and theoretical constructs in CER. They have observed a progression of domain-specific theory and propose a framework for developing new theoretical constructs in CER.

The CER community has matured, and there is an overall progression related to the use of theory. The latter is evident from the recent papers commented on above. However, it is also apparent that the context of studies still needs to be captured in order to use theory properly in CER papers.

### 3.2 Introduction of Theoretically Robust Research: The First Generation

Uppsala Computing Education Research Group, UpCERG, can trace its first publications to 1996. They were descriptive and mainly presented the teachers' experiences and impressions, possibly with some statistical analyses as a complement, and corresponded often to what Valentine [43] refers to as Marco Polo papers.

The first Ph.D. thesis was produced by Berglund [3], followed by that of Eckerdal [15]. In contrast to the first publications, the theses of Berglund and Eckerdal applied a theoretically well-developed phenomenographic research approach [31, 32] to their studies. Phenomenography is a qualitative research approach that aims to describe how something (called a phenomenon with the terminology from the approach) is understood (or experienced) within a cohort of learners, for example, how university students in IT understand a particular network [2] or the concept of evolution as understood by master students in biology [22] could be the phenomena of interest. The outcome of a phenomenographic study is a set of categories, each of which describes a certain way in which the phenomenon is perceived in the cohort.

Berglund contextualized the results from his phenomenographic studies, using activity theory [16], to describe the learning of CS in an internationally distributed student project. In this way, it became possible to see the learning of particular phenomena within IT, as they were experienced by the students. At the same time, the learning was seen as a part of a broader setting [46]. Eckerdal, inspired by the dualism between and, at the same time, the interaction between theory and practice in students' learning of programming, discussed her phenomenographic outcome space in terms of students' learning of fundamental programming practice. In both these theses, theory was made explicit, both in terms of focus on the CS content and in the use of a robust, qualitative, interpretative theoretical basis. With the work of Berglund and Eckerdal, the importance of grounding research in sound theoretical underpinnings became a part of the "life" and "meaning" of the UpCERG team. This introduction of phenomenography served as a platform and example for how methodologically and theoretical rigorous research could be used to gain insights into computing education.

### 3.3 The Next Generation

Building on these insights, the subsequent theses from the team had clear foci on their research questions and contexts while still writing theoretically well-founded theses. Also, the theoretical and methodological repertoire was extended by selecting research approaches that by the authors deemed relevant for tackling their research questions [13].

In her thesis on students' development of an identity as a computer scientist, Peters theoretical point of departure is the work of Lave and Wenger [26] and studies

how participation in a cohort affects and constraints their individual becoming and how the participants shape each other. Boustedt [8] studies the opportunities that can help to overcome the gaps between newly hired and experienced CS professionals by taking a phenomenographic approach. Learning in project courses and the possible gap between students' experiences and teachers' expectations is the core topic of Wiggberg's work [47]. He developed a method focusing on capturing the students' experiences and exploring choices to engage in learning vs pressure to ensure successful implementation outcomes. Alghamdi [1] took a different perspective on capturing teachers' and students' experiences. He studied the experience of teachers and female learners and how to enhance CS education in the context of Saudi Arabia through a set of case studies [42]. Daniels's thesis [12], despite being anchored in case studies as well as action research (see, e.g. [37], still differs from the previously mentioned work. The difference is that the core part of the theoretical development lies in the object of the students' learning, in his case in understanding and developing insights in professional competencies (see [12, section 5.5.1]), and not mainly in the methodology.

This development among the Ph.D. students has developed through rich and lively discussions in the entire UpCERG team on what CER is and how it stands out as different from research in education, sociology, or computer science. This has resulted in several publications discussing the nature of theory and the use of theories in CER [13, 34, 35], and also on the multitude of possible theoretical approaches needed to meet diverse research goals in relation to the culturally situated nature of CS; e.g. [7], on qualitative research in CS education.

### *3.4 Current Development*

The current work of UpCERG focuses on the theoretical achievements of the earlier Ph.D. theses, but also demonstrates a more significant variation in the use of theories and methodologies. An example of this is the work by Kristina von Hausswolff [44]. A recent development is that ethical and moral values are made visible and become essential in the research in a way inspired by that advocated by Clear [10]. Anne Peter's work on sustainability in computing and computing education, as well as Virginia Grande's research on role models, e.g. [18], can here serve as illustrations. Another illustration is the work of Tina Vrieler who uses the computer science capital concept to reflect on instructional design and teaching practice [45]. In her work, she draws on Bourdieu's sociological theory of capital. Still, what unites the current work of UpCERG is its theoretical awareness. The research questions vary between the projects and researchers, but the importance of anchoring research in theory can be found in most of the publications of the last decade.

## 4 Conclusions

This case study on the Uppsala Computing Education Research Group, UpCERG, illustrates the importance of being connected to an international research community and being accepted as a research group in the local context. It also shows how being persistent and having an open mind to how the research field is developing are essential components in establishing a research group in a new area. The example of the journey of one of the researchers, Daniels, gives a personal illustration of the formation of an established research program from a conviction that computing education should rely on a solid theoretical foundation.

The formation of UpCERG has been a joint effort, and the hallmark of the group is the openness to different theoretical approaches and an interest in new ideas. Another guiding light is addressing challenging issues regarding both understanding how computing education can be improved in specific cases and contributing to the grand challenges of society today. An essential common denominator in the work done in UpCERG is to base it on solid theoretical foundation, which is in strong alignment with the general message of this book.

UpCERG has been a part of, and continues to contribute to, the theoretical development in Computing Education Research. The team has a focus on evidence-based learning [20], combined with empirically based research with a rigorous theoretical stance. In summary, as argued by Suppes [38], theory has served to offer analogies, reorganize empirical findings, see the complexity and avoid bare empiricism. Further, the theoretical foundation has provided a language to learn from others and share conclusions. We hope that this case study will provide inspiration and guidance for others to pursue computing education research.

## References

1. Alghamdi, F.: Dimensions of Professionalism : A Study of Computer Science Teaching in Saudi Arabia. Acta Universitatis Upsaliensis (2020). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-418925
2. Berglund, A.: On the understanding of computer network protocols (2002)
3. Berglund, A.: Learning computer systems in a distributed project course : The what, why, how and where. Acta Universitatis Upsaliensis (2005). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-5754
4. Berglund, A., Daniels, M., Hedenborg, M., Tengstrand, A.: Assessment to Increase Students' Creativity: Two Case Studies. European Journal of Engineering Education (2006). URL https://www.tandfonline.com/doi/abs/10.1080/0304379980230106. Publisher: Taylor & Francis Group
5. Berglund, A., Daniels, M., Pears, A.: Through the eyes of a research team: Using theory to enhance STEM Education. In: 2021 6th International STEM Education Conference (iSTEM-Ed), pp. 1–4. IEEE, Pattaya, Thailand (2021). URL https://ieeexplore.ieee.org/document/9625125/

6. Berglund, A., Foyer, P.O., Karlsson, V., Svärdström, A.: Full scale study with new approaches to examining the students on the engineering physics programme in Uppsala (1996). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-40255
7. Berglund, A., Thota, N.: A glimpse into the cultural situatedness of computer science : Some insights from a pilot study. In: International Conference on Learning and Teaching in Computing and Engineering (LaTiCE 2014), pp. 92–99. IEEE Computer Society (2014). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-226531
8. Boustedt, J.: On the Road to a Software Profession : Students' Experiences of Concepts and Thresholds. Acta Universitatis Upsaliensis (2010). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-122304
9. Cajander, Å.: Usability – Who Cares? : The Introduction of User-Centred Systems Design in Organisations. Acta Universitatis Upsaliensis (2010). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-122387
10. Clear, T.: Valuing computer science education research? In: Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006, Baltic Sea '06, pp. 8–18. Association for Computing Machinery, New York, NY, USA (2006). URL https://doi.org/10.1145/1315803.1315806
11. Clear, T.: Supporting the work of global virtual teams: the role of technology-use mediation. Thesis, Auckland University of Technology (2008). URL https://openrepository.aut.ac.nz/handle/10292/650. Accepted: 2009-06-14T23:48:49Z
12. Daniels, M.: Developing and Assessing Professional Competencies: a Pipe Dream? : Experiences from an Open-Ended Group Project Learning Environment. Acta Universitatis Upsaliensis (2011). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-145983
13. Daniels, M., Pears, A.: Models and methods for computing education research. In: Proceedings of the Fourteenth Australasian Computing Education Conference - Volume 123, ACE '12, pp. 95–102. Australian Computer Society, Inc., AUS (2012)
14. Daniels, M., Petre, M., Almstrum, V., Asplund, L., Bjorkman, C., Erickson, C., Klein, B., Last, M.: RUNESTONE, an international student collaboration project. In: FIE '98. 28th Annual Frontiers in Education Conference. Moving from 'Teacher-Centered' to 'Learner-Centered' Education. Conference Proceedings (Cat. No.98CH36214), vol. 2, pp. 727–732 vol.2 (1998). https://doi.org/10.1109/FIE.1998.738780. ISSN: 0190-5848
15. Eckerdal, A.: Novice Programming Students' Learning of Concepts and Practise. Acta Universitatis Upsaliensis (2009). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-9551
16. Engeström, Y.: Learning by Expanding: An Activity-Theoretical Approach to Developmental Research (2014). URL https://www.cambridge.org/core/books/learning-by-expanding/6D0648C3DEDE20157B359E464AFDB8C1. ISBN: 9781139814744 9781107074422 9781107640108 Publisher: Cambridge University Press
17. Faulkner, X., Daniels, M., Newman, I.: Open ended group projects (OEGP) : A way of including diversity in the IT curriculum. In: Diversity in information technology education : Issues and controversities, pp. 166–195. Information Science Publishing, London (2006)
18. Grande, V., Peters, A., Daniels, M., Tedre, M.: "Participating Under the Influence": How Role Models Affect the Computing Discipline, Profession, and Student Population. In: 2018 IEEE Frontiers in Education Conference (FIE), pp. 1–9 (2018). DOI https://doi.org/10.1109/FIE.2018.8658944. ISSN: 2377-634X
19. Hamrin, P., Persson, M.: Exploring the Notion of Space in Virtual Collaborations : Finding Prerequisites for Success in Virtual Teams. undefined (2010)
20. Hattie, J.A.C.: Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement, 1st edition edn. Routledge (2008)
21. Hause, M.L.: Software development performance in remote student teams in international computer science collaboration. phd, The Open University (2004). URL http://oro.open.ac.uk/54622/

22. Holm, K.: Perceptions of the Concept of Evolution among Undergraduate Biology Students. In: EARLI Special Interest Group 9. Phenomenography and Variation Theory: Disciplinary knowledge and Necessary Conditions of Learning, University of Oxford, pp. 15–15 (2014). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-233032
23. Johnson, R.B., Christensen, L.B.: Educational Research: Quantitative, Qualitative, and Mixed Approaches, 3rd edition edn. SAGE Publications, Inc, Los Angeles (2007)
24. Klette, K.: The Role of Theory in Educational Research, (2011). https://9pdf.net/document/q5mr2357-the-role-of-theory-in-educational-research.html. Accessed 27 Jun 2022
25. Last, M.Z.: Investigating the group development process in virtual student software project teams. phd, Kingston University (2003). URL http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.275111
26. Lave, J., Wenger, E.: Situated Learning: Legitimate Peripheral Participation, 1st edition edn. Cambridge University Press, Cambridge England ; New York (1991)
27. Leedy, P.D., Omrod, J.E.: Practical research: Planning and design. Pearson Educational International and Prentice Hal, Englewood Cliffs, N. J (2005)
28. Lishinski, A., Good, J., Sands, P., Yadav, A.: Methodological Rigor and Theoretical Foundations of CS Education Research. In: Proceedings of the 2016 ACM Conference on International Computing Education Research, ICER '16, pp. 161–169. Association for Computing Machinery, New York, NY, USA (2016). URL https://doi.org/10.1145/2960310.2960328
29. Malmi, L., Sheard, J., Kinnunen, P., Simon, Sinclair, J.: Development and Use of Domain-Specific Learning Theories, Models and Instruments in Computing Education. ACM Transactions on Computing Education (2022). URL https://doi.org/10.1145/3530221. Just Accepted
30. Malmi, L., Sheard, J., Simon, Bednarik, R., Helminen, J., Kinnunen, P., Korhonen, A., Myller, N., Sorva, J., Taherkhani, A.: Theoretical underpinnings of computing education research: what is the evidence? In: Proceedings of the tenth annual conference on International computing education research, ICER '14, pp. 27–34. Association for Computing Machinery, New York, NY, USA (2014). URL https://doi.org/10.1145/2632320.2632358
31. Marton, F.: Necessary Conditions of Learning. Routledge (2014). Google-Books-ID: bbzcAwAAQBAJ
32. Marton, F., Booth, S.A.: Learning and Awareness. Psychology Press (1997)
33. McMillan, J.H., Schumacher, S.: Research in Education: A Conceptual Introduction, 5th edition edn. Allyn & Bacon, New York (2000)
34. Pears, A., Seidman, S., Eney, C., Kinnunen, P., Malmi, L.: Constructing a core literature for computing education research. ACM SIGCSE Bulletin **37**(4), 152–161 (2005). URL https://doi.org/10.1145/1113847.1113893
35. Pears, A., Thota, N., Kinnunen, P., Berglund, A.: Harnessing theory in the service of engineering education research. In: 2012 Frontiers in Education Conference Proceedings, pp. 1–5 (2012). DOI https://doi.org/10.1109/FIE.2012.6462292. ISSN: 2377-634X
36. Pears, A.N., Daniels, M.: Structuring CSed research studies: connecting the pieces. ACM SIGCSE Bulletin **35**(3), 149–153 (2003). URL https://doi.org/10.1145/961290.961553
37. Reason, P., Bradbury-Huang, H. (eds.): Handbook of Action Research: Participative Inquiry and Practice, 1st edition edn. SAGE Publications Ltd, London ; Thousand Oaks, Calif (2001)
38. Suppes, P.: The Place of Theory in Educational Research. Educational Researcher **3**(6), 3–10 (1974). URL https://doi.org/10.3102/0013189X003006003. Publisher: American Educational Research Association
39. Szabo, C., Sheard, J.: Learning Theories Use and Relationships in Computing Education Research. ACM Transactions on Computing Education (2021). URL https://doi.org/10.1145/3487056. Just Accepted
40. Tedre, M., Pajunen, J.: Grand theories or design guidelines? Perspectives on the role of theory in computing education research. ACM Transactions on Computing Education (2021). URL https://doi.org/10.1145/3487049. Just Accepted

41. Tenenberg, J., Malmi, L.: Editorial: Conceptualizing and Using Theory in Computing Education Research. ACM Transactions on Computing Education (2022). URL https://doi.org/10.1145/3542952. Just Accepted
42. Thomas, G.: How to Do Your Case Study, second edition edn. SAGE Publications Ltd, Los Angeles (2015)
43. Valentine, D.W.: CS educational research: a meta-analysis of SIGCSE technical symposium proceedings. ACM SIGCSE Bulletin **36**(1), 255–259 (2004). URL https://doi.org/10.1145/1028174.971391
44. von Hausswolff, K.: Practical thinking in programming education: Novices learning hands-on. Acta Universitatis Upsaliensis (2022). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-461455
45. Vrieler, T., Salminen-Karlsson, M.: A Sociocultural Perspective on Computer Science Capital and its Pedagogical Implications in Computer Science Education. ACM Transactions on Computing Education (2021). URL https://doi.org/10.1145/3487052. Just Accepted
46. Vygotsky, L.S.: Mind in Society: Development of Higher Psychological Processes. Harvard University Press (1980). URL http://www.jstor.org/stable/10.2307/j.ctvjf9vz4. Cole, Michael and Jolm-Steiner, Vera and Scribner, Sylvia and Souberman, Ellen
47. Wiggberg, M.: Computer Science Project Courses : Contrasting Students' Experiences with Teachers' Expectations. Acta Universitatis Upsaliensis (2010). URL http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-120081

# Future Technology Lab: A Plug-in Campus as an Agent of Change for Computing Education Research in the Global South


**Maria Ntinda, Mikko Apiola, and Erkki Sutinen**

## 1 Introduction

Computing Education (CE) and Computing Education Research (CER) at universities in the Global South (GS) continue to fall behind their counterparts despite the advancement in Information Communication Technologies (ICT). This could be attributed to the adaptation of universal computing curricula which were not developed for the realities in the GS. The imported curricula were not designed to suit the local context of underinvestment and under-resourcing of higher education in the GS [1], nor to fit the societal or business context in the GS and its demands, which are different from those in the Global North (GN). The contextual challenges often mean that the supposedly generic curricula by professional global associations such as the Institute of Electrical and Electronics Engineers (IEEE) and the Association for Computing Machinery (ACM), typically applied in developed countries, do not work well in the GS [1].

For a successful implementation of CE and CER, there is a need to rethink CE and CER in the context of the GS [2]. Universities are therefore adopting open innovation strategies to access and integrate external sources of knowledge for better collaboration opportunities to revitalise their education, research, or societal impact [3]. Researchers and practitioners are collaborating to gain an understanding of multiple aspects of teaching and learning processes of various topics in the computing curriculum to build generalizable evidence about problems in students'

M. Ntinda (✉) · E. Sutinen
University of Turku, Department of Computing, Turku, Finland
e-mail: mnntin@utu.fi; erkki.sutinen@utu.fi

M. Apiola
University of Eastern Finland, Joensuu, Finland
e-mail: mikko.apiola@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_13

learning, to understand the efficacy of new teaching approaches to solve these problems which might be context-dependent [4].

CE is often characterised as a field that draws on approaches and methods from cognitive psychology, education, and computer science [4], thus there are no specific methods and approaches for teaching computer education.

For CE to reshape based on the expectations of the GS, cross-border activities are taking place through studying abroad and via online exchange programmes [5]. Cross-border refers to teachers, students, institutions, or course materials crossing national jurisdictional borders [5], and this form of education has evolved in which established universities expand their services through the establishment of satellite campuses also known as branch or offshore campuses [5].

While these campuses are self-contained and fully functional, parent universities retain full autonomy to run these satellite campuses [3], attracting experienced professors, researchers and funding, thus contributing to their host country's economic and human capital development [5]. Satellite campuses however entail the risks of crowding-out local universities, which may result in those universities receiving less public funding, consequently facing greater challenges and difficulties in attracting the best academics, researchers, and students [6]. Calling for collaboration between universities instead of competition is thus of prime importance for mutual learning and reform.

A plug-in campus is an alternative to a satellite campus, introduced by a university in the GN as a catalyst to accelerate research, innovation, and development in a host university in the GS [3]. In our case, the Future Tech Laboratory (FTLab) is a concrete example of a plug-in campus of the University of Turku (UTU), which was established to accelerate educational practice and related CER for the University of Namibia (UNAM). UTU and UNAM have a mutual understanding and a bidirectional relationship of collaboration, growth, and cross-inspiration. This chapter provides a case study of how CE, research, and innovation are being reshaped through the FTLab and draws lessons to transform CE in Namibia. The chapter presents a range of contributions that the FTLab has made in Namibia by engaging in different missions: teaching, research, and community engagement. It is worth noting that all initiatives adopted from Finland were contextualized to suit the Namibia context, as they were developed for Finland rather than for the Namibian realities. The present study adopted the design reality gap framework to analyse, comprehend, evaluate and improve the implementation of the FTLab initiative. This research is the first to study the contribution of a plug-in campus to improve CER in Namibia.

This chapter is structured as follows: Section 2 presents how Hevner's [7] Design Science Research (DSR) was presented in this study to identify demands that better fit to contextualise CER in the GS. Section 3 presents the results, discussing the Relevance of the environment, which is the Namibian society, focusing on the expectations of CE, especially as identified by the activities of the FTLab. Section 4 presents the results on the Rigor (based on literature), where the Design Reality Gap (DRG) model [7] was introduced as a tool to show the discrepancy of a European plug-in campus in the GS. In Sect. 5, results of the Design cycle are presented,

suggesting solutions to the requirements identified in Sect. 3. Section 6 presents the discussion and recommendation. Conclusions and future work are presented in Sect. 7.

## 2 Research Design

To contextualise CE and CER to better fit the demands identified in the GS, DSR was applied in this study [7]. DSR is a pragmatic and constructive research approach that makes use of research to design a solution, called an artefact, to an identified problem, as expressed in a given environment, in a way that considers the contemporary knowledge base.

Thus, DSR applies in parallel three threads, called relevance, rigor, and design cycles, as depicted in Fig. 1. The relevance cycle ensures that the design process is based on the real demands of the given environment, the rigor cycle identifies and integrates the key aspects of the contemporary knowledge base and updates the knowledge base by the input of the DSR instance, and the design cycle integrates design and evaluation iterations for constructing the solution [7].

The DRG will be applied on how to make CE and CER relevant in the GS, instead of being no more than an imported product. In our case, the cycles are instantiated in Sects. 3, 4, and 5.

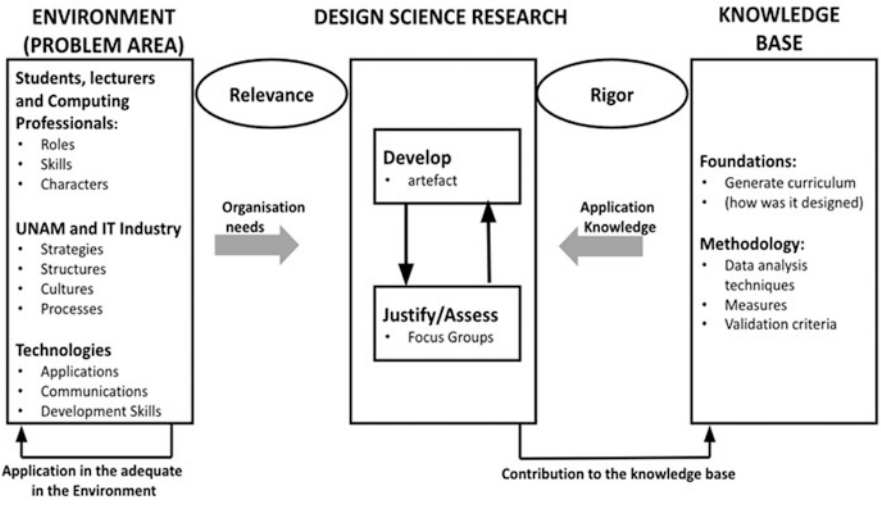


**Fig. 1** DSR as used in this study. (Adapted from Ref. [7])

## 3 Relevance

The relevance cycle is based on the expectations for CE and, hence, CER, expressed by the Namibian society and exemplified by the observations in the FTLab activities.

### *3.1 CE and CER Requirements in Namibia*

Computing Education is a much younger field than other branches of science education in Namibia. As such, CE is primarily taught at university level and mostly as an elective in private schools. Lately, a few public schools have introduced computing education. There are however no specific requirements to enroll for CE in primary and secondary school in Namibia. Also, the computing curricula at secondary school and at university level in Namibia are disconnected as the university does not continue with what was taught in secondary school. Hence, students who apply for computing-related degrees at universities in Namibia do not need to have prior computing knowledge.

Computing Education has gathered momentum over the past years, focusing on teaching programming and teaching methods in computing. However, missing or limited infrastructure, power systems delivering electricity access, as well as internet access, computing teaching efficacy, and the bureaucratically heavy process of updating CE curricula in both secondary and higher education in Namibia are among the challenges hindering the progress of CE. Research on how to teach programming using robotics in Namibia is at the forefront of educational institutions [8]. Research on computing education is notable in Namibia [8], for example, in studies aimed at narrowing the gap between academia and industry [9].

### *3.2 The Concept of the FTLab Plug-in Campus*

To accelerate computing research, innovation, and development in Namibia, the FTLab was implemented at the UNAM. The FTLab is an example of a plug-in campus, an alternative to a full-scale branch or satellite campus that makes use of the host university's infrastructure by renting a small space from its host [3]. The term plug-in refers to the campus's flexibility to its students and making use of the host university's services. A plug-in campus concept evolved from an arrangement between UTU and UNAM whereby UTU wanted to reshape itself to the challenges of another geographical and demographic context and UNAM wanted to renew its education, research, or societal impact [3]. The flexible nature of the plug-in campus allows students to work from anywhere at any time as they are not bound to its physicality. Apart from being plugged into a host university, it can also be plugged into a professional life or an individual life, as illustrated in Fig. 2.

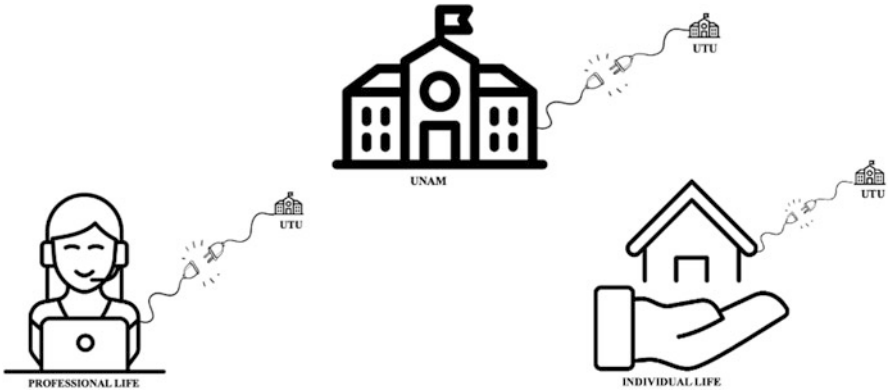


**Fig. 2** Concept of the plug-in campus

Through various initiatives such as learning and teaching, projects, research, and innovation happening at the FTLab, required skills can be stimulated. According to [9], the competencies requirements in the SE industry and postgraduate studies in Namibia do not match the skill sets of graduates. Various activities are therefore introduced at the FTLab to assist with narrowing the gap between the SE industry and academia in Namibia. Moreover, strategies are developed to accelerate CER in Namibia and the GS. Those initiatives at the FTLab are being accelerated with the assistance of all involved universities, and external stakeholders: public sector or government, private sector, and other employers, start-up community, and the Non-Governmental Organisations or civil society. The foundation and products of the FTLab are depicted in Fig. 3 using the concept tree.

The roots of the tree illustrate the attributes such as values and resources etc. that the plug-in campus was built for. The trunk of the tree illustrates the activities to be enhanced, the branches illustrate activities happening while the fruits are the end products (achievements) such as degrees, partnerships, etc. of the FTLab.

## *3.3 Expectations from CE and CER as Identified by the Research, Development, and Innovation Projects at the FTLab*

Projects at the FTLab provide new CER opportunities that need to be addressed. The main project at the FTLab is the establishment of the remote presence project that aims to design and develop the use of remote presence technologies to create a greater sense of togetherness for its community [10]. Projects at the FTLab involve local communities in co-designing and co-developing applications to suit the context of those communities, such as the projects on climate services for small-scale farmers [9].

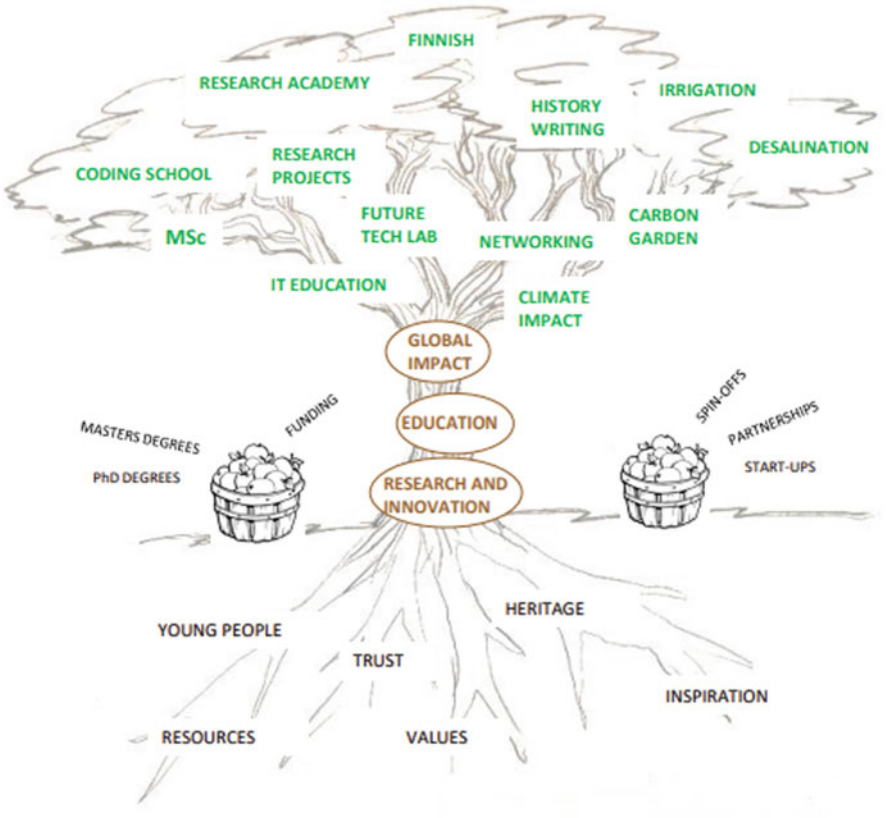


**Fig. 3** Concept tree of the plug-in campus

Since the inception of the FTLab in 2019, more than 40 research articles have been published based on joint research projects [10], that resulted from collaborations mostly between Finnish and Namibian researchers. Finland is known for its high-quality education and research which Namibia could benefit from and learn from some of its approaches.

## *3.4 Accelerating CE in Namibia Via the FTLab*

According to the original plans, the FTLab aimed to offer opportunities to students in Namibia through its coding school, micro-credentials or short courses, a master's degree in SE, and a doctoral degree in Computer Science [10]. Both degree programmes offered at the FTLab are adapted from the UTU main campus and contain contextual elements in projects and demonstrations [10]. The MSE programme is

structured in a way that would allow students to design, develop, implement, and evaluate software solutions and understand IT-related challenges in the GS to meet local and international requirements [10]. Graduates from the doctoral programme will be able to conduct both independent and collaborative research and possess leadership competencies.

Micro-credentials or short courses are offered on a need basis, and currently the Sustainability Engineering module of 20 ECTS credit points received funding and will be offered at the FTLab. Co-teaching also happens at the FTLab, for example, an Artificial Intelligence course registered under the Department of Computing, Mathematics and Statistical Sciences (DCMSS) at UNAM was co-taught by a lecturer from the DCMSS, a professor from the FTLab, and professionals from the industry [11]. Short, concise, and hands-on courses based on domain-specific programming languages are also offered at the FTLab, see [10].

Although one of the aims of the FTLab was for students enrolled in the PhD and MSE programme to work closely with students from UTU's main campus and both local and international SE companies, unfortunately, the FTLab only managed to attract students for the PhD programme.

### *3.5 Community Outreach at the FTLab*

The FTLab has assisted in the acceleration of CER in Namibia through both online and physical events and workshops over the years since its establishment [3]. Both projects carried out at the FTLab contributed to the enhancement of CER in Namibia. In 2021, the FTLab hosted a one-day event that introduced learners from the C.I.D.S Centre to the basics of robotics to spark the learners' interest in Science, Technology, Engineering, and Mathematics (STEM) [10]. In addition, a crash course on the Finnish language was offered at the FTLab to spark interest in the Finnish language. The FTLab has also been on the wheels, towards being a mobile plug-in campus shaping as a metaversity [12], offering robotic lessons at different institutions such as Nakayale in Ruacana, De Duine High School, and Walvisbay Primary school, as well as at the UNAM Engineering campus in Ongwediva.

## 4 Rigor Cycle

The rigor cycle summarises the relevant knowledge of satellite campuses – of which the FTLab is an example – and presents Heeks' design-reality gap [1] as an analysis tool for identifying, understanding, and correcting the shortcomings of direct imports of design from the GN so that they could work in the GS.

### 4.1 Satellite Campuses as Agents of Change for CER in the Global South

Satellite campuses have become important dimensions of higher education as their establishments are often motivated by institutions' willingness to increase their quality of teaching and research activities, visibility, and attraction to public funding [13]. These campuses are normally set up in areas previously lacking a university or where local universities are under-resourced to meet the high demand of their localities by engaging in all their missions: teaching, research, and business and community engagement [14]. Satellite campuses are outcomes of either regional institutions that are merged into larger universities or are related to strategies of territorial diversification of large universities [15] and are likely to attract the best students, staff, and visitors such as guests, institutional visitors, seminars, congress, and event attendees [16]. According to Miller-Idriss and Hanauer [5], most satellite campuses are found in the Middle East. Other satellite campuses include Carnegie Mellon University, a US university in Kigali, Rwanda [17]. Monash University also has a satellite campus in South Africa [18]. Similarly, satellite campuses are being set up in countries in the GN as well, for example, the Limkokwing University of Creative Technology, a Malaysian-based university has a branch campus in London [19]. In Namibia, Limkokwing University of Creative Technology is busy setting up a satellite campus in Windhoek. Botho University [20] is another international university with a branch campus in Namibia.

Satellite campuses belonging to international companies bring competition to public universities instead of collaborations. These universities also experience challenges – cultural diffusion, governance, enrollment, establishment and operation risks, and quality education – that need to be addressed when setting up satellite campuses [14].

There is no doubt that satellite campuses yield benefits for the host country, but the success of these campuses depends on consulting local stakeholders to identify local strengths. Since the FTLab is similar to satellite campuses (see differences between the plug-in campus and satellite campus [3]) the beneficiaries are of different socio-economic and cultural dimensions, hence those designs might not fit the local realities. Heek's design reality gap model [1] was then adopted to evaluate the implementation of the FTLab to assess the gap that might have occurred in the design expectation and the realities of the FTLab.

### 4.2 Evaluating Progress and Challenges Via Design-Reality Gap (DRG) Analysis

The Design-Reality Gap (DRG) model was developed to analyse organisational change and the risk associated with it, and argues that the design expectations of an organisation may match or mismatch the real situation found in the context of

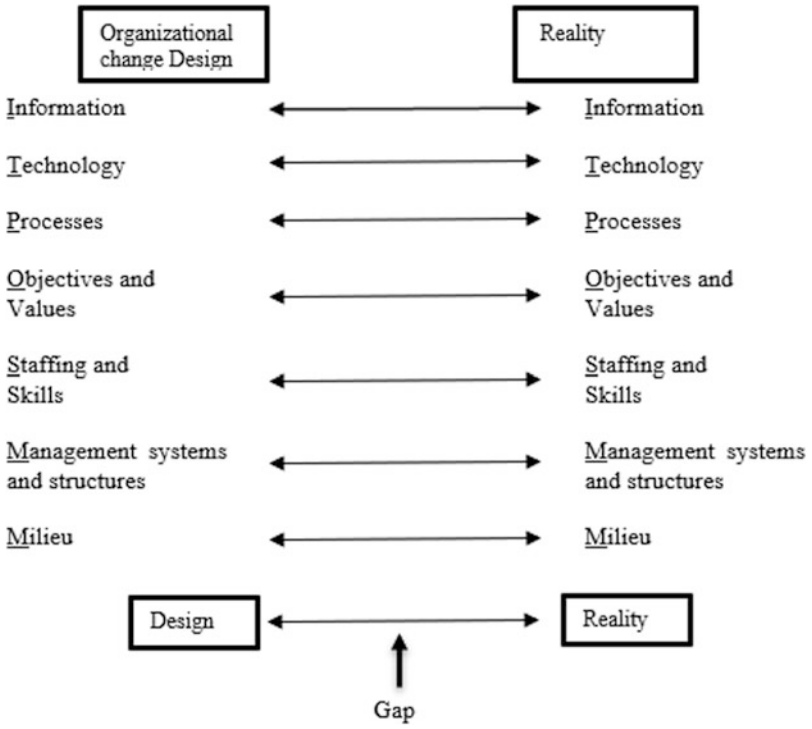


**Fig. 4** The DRG model for analysing change [1]

the implementation [1]. There are eight extensions of the DRG, and they can be summarised with the OPTIMISM mnemonic as illustrated in Fig. 4 [1].

The DRG model can be applied as either a risk analysis tool or a project evaluation tool [2]. When the DRG model is used as a risk analysis tool, the design and reality are assessed cross-sectionally at a particular moment in time to assess the gap. The larger the gap, the greater the risk of project failure and vice versa [1]. On the other hand, when the DRG model is used as a project evaluation tool, the design and reality are assessed longitudinally; with the expectations within the design compared to the reality sometime later after implementation (though recognising that implementation is often an ongoing process that can rarely be seen as completed) [2]. This study adopted the latter to enable us to assess the extent of success or failure of the implementation of the FTLab since its inception in 2019. Moreover, areas in which further changes are required are identified for further improvements. Two staff members of the FTLab who are based in Windhoek were interviewed and the results are presented in Sect. 5.

# 5 Design Results

The design cycle collects the process as a solution for removing the gap in the tension between design (a universal model of CE and CER) and reality (contextual demands of CE and CER in the GS). The progress of the implementation of the FTLab is examined by adopting the DRG model with the assumption that there will be gaps in the design of the ideal FTLab as initially proposed, and the reality observed in its current implementation.

## *5.1 Identifying and Analysing the Design-Reality Gap at the FTLab*

According to Bass and Heeks [1], the success and failure of a system are multifactorial, hence the importance of considering the environment where an initiative is implemented. We present the gaps using each of the eight OPTIMISM dimensions of DRG in turn [1]. Besides the DRG in the FTLab itself, we also enlist gaps identified by the activities run at the FTLab.

**Objectives and values (both formal strategies and culture, and informal goals)**

*Design Expectations*

The FTLab aims to catalyse and accelerate CE and in turn CER at UNAM with the emphasis on contextual innovation, collaboration, and mutual interaction between UTU, UNAM, Finnish–Namibian industrial partners, and local communities [3]. The objective is to be met through co-working on projects, co-designing applications, and co-learning different technologies. To realise its objectives, the FTLab was designed to include the implementation of state-of-the-art remote presence technologies for online learning of short courses, a masters, and PhD degree programmes [21]. Since degree programmes have the same requirements as the main campus of UTU, students in Namibia and Finland need to attain equal ECTS (European Credit Transfer System) credit points and complete a thesis to graduate with both masters and PhD degrees.

*Reality*

Co-creation of applications and co-working on projects is ongoing at the FTLab. Currently, both local and international students are enrolled in the PhD programme. Although there have been calls for applications for the master's degree, there are no students enrolled in the master's programme. This could be attributed to the English language test entry and the cost of the master's programme. Also, activities at the FTLab slowed down during the Covid-19 pandemic.

*Design-Reality gap*

The observation in the study shows that there appears to be a relatively good match between the design expectation and reality in terms of objectives and values at the FTLab. However, an ethnocomputing-based approach in ICT-related education could be implemented, based on an indigenous community's context-driven principles [22]. We therefore recommend collaboration between students and lecturers to work together to contexualize the curriculum. The FTLab could harness software development to develop an online digital library to motivate users to contribute "resource-scarce languages" on a Web-based portal, as similarly done in a study in South Africa [23].

**Processes (from individual tasks up to broader business processes)**

*Design Expectations*

There are ongoing collaborations between the Finnish-Namibia community, universities, and industry. The processes are supported via writing, grants, sponsorships, and community services. The emphasis at the plug-in campus relies on graduating students who understand IT-related challenges to meet local and international requirements.

*Reality*

These initiatives are effective to support these processes. Students at the FTLab work closely with industry and there are ongoing collaborations across different disciplines. For example, the UNESCO summer school received sponsorship from international industry, see [24], community projects [25], and co-writing of grant proposals amongst both Finnish-Namibian communities are ongoing. However, no grant has been successful thus far. The FTLab has been successful in research, and this is evident in articles published by researches from the FTLab, see [10]. There is however a need to come up with initiatives to stimulate industry-academia collaboration as there is a disconnect between academia and industry in Namibia. Neither party seems to see the relevance of collaborations or a need to strengthen existing ties.

*Design-Reality gap*

There has not been an observation in terms of the masters degree programme. For the PhD programme, the FTLab did not attract a lot of students. However, other initiatives are a success, so the study shows that there appears to be a relatively good match in other initiatives and an unmatched in the masters and PhD programmes.

**Technology (ICTs and other relevant technologies)**

*Design Expectations*

The FTLab has a project that aims to remove the critical research hurdles holding back the use of sensory immersive 3D video as an alternative to ordinary video-

conferencing. The state-of-the-art immersive telepresence entails the design and development of a custom live 3D capture system for a higher fidelity immersive experience, targeting local and remote collaboration of small groups of two (2) to six (6) [26]. Also, there is stable Internet connectivity at the FTLab. These technologies are to enable collaborations from Finland in a remote presence. Other technologies like printers, projectors and robotics sets are also available at the FTLab. Current students have access to the online library and classes at the main campus of UTU.

*Reality*

The state-of-the-art immersive telepresence to support the practical and collaborative group work element in the remotely taught degrees as initially proposed was not implemented but is under development in the Academy of Finland funded project in 2021–25. However, the internet connectivity is stable, and learning is still possible at the FTLab.

*Design-Reality gap*

The observation in the study shows that there appears to be a mismatch between the design expectation and reality in terms of technology at the FTLab. The immersive telepresence as initially proposed was not implemented and is under development but could be better integrated to CER. There is however basic technology to operate at the FTLab.

**Information and access to it (data stores, data flows, etc.)**

*Design Expectations*

Effective implementation of the FTLab depends on effective information flows between stakeholders: students, staff, local and international communities, and industry.

*Reality*

- *Access to the Lab*

  Students and local communities and industry can make use of the FTLab to co-design and co-develop projects. There is a campus coordinator, and students can always get the keys to the FTLab when they want to use the facilities to host workshops, teach or study.

- *Access to the lecturers*

  Most courses are offered online from the base university and for short courses offered physically, students and other participants can either email lecturers or set up meetings physically or online.

- *Access to industry*

  One of the objectives of the FTLab is to enhance collaboration with external stakeholders, both locally and internationally. The FTLab also provides space for

member partners to use the FTLab. Also, there have been webinars for industry-academia interaction [10].

*Design-Reality gap*

The observation in the study shows that there appears to be a match between the design expectation and reality in terms of information (data stores, data flows, etc) at the FTLab. There are expectations of open data being available to be used outside or inside of the FTLab. These data include but are not limited to data from research articles published or climate data from developed climate applications. There is however a need to increase the involvement of local industry to guide activities at the FTLab.

**Management structures and systems**

*Design Expectations*

Since the FTLab is an offshore campus of the UTU, its management and structure, and faculties are of the base university.

*Reality*

The majority of staff members are based at UTU's main campus. There are however two full-time staff: a professor and the campus coordinator based at the FTLab. When the need arises, UNAM offers its staff members to assist with activities at the FTLab. The flexible nature of the FTLab allows for UNAM staff to assist in co-design and co-working with the Namibian community. The management structure and systems are those of the base university.

*Design-Reality gap*

The observation in the study shows that there appears to be a mismatch between the design expectation and reality in terms of management structures and systems at the plug-in campus. Although there is mutual collaboration between UNAM and UTU, the FTLab faces a challenge in offering co-courses with UNAM as the integration of courses is difficult due to different accreditation systems. For example, UTU offered courses such as tiny machine learning to UNAM students, but these students cannot use credits from such courses for their studies at UNAM. Joint courses and joint professorship between both institutions could be initiated.

**Financial Investment**

*Design Expectations*

Financial support comes from heterogeneous, mutually independent sources: industry of the base, host and third countries, grants, participants attending workshops, summer schools, or alike, and students registered for degree programmes [3].

*Reality*

There have not been intakes of master's students and no grants application has been successful so far. Funds at the FTLab come from the base university and industry of the base country who pay membership fees, and from participants attending workshops, and summer schools. The master's degree programme is expensive compared to masters degrees programmes offered by local universities. Also, companies did not send their employees for studies as anticipated.

*Design-Reality gap*

The observation in the study shows that there appears to be a relative mismatch between the design expectation and reality in terms of financial investment at the plug-in campus. Companies did not send their employees or sponsor students who will work on real research projects for those companies in return. Although there were industry members who would pay membership fees, the FTLab did not attract enough members, especially locally. Local companies do not see the benefit of applying for joint projects funded by external sources as they see UTU primarily as a customer or a donor. The FTLab mostly depends on financial support from its base university. Hence there is a need for the FTLab to create initiatives to generate revenue to complement resources provided by the base university.

**Staffing and skills**

*Design Expectations*

The FTLab focuses on ensuring the quality of its main campus is met but in the context of the satellite campus [14]. PhD courses are taught by lecturers from the base university as well as the supervision of the thesis. For workshops, UNAM offers its staff members to assist in collaboration with UTU or industry practitioners.

*Reality*

There is a full-time professor at the plug-in campus, supervising PhD students. These students also have co-supervisors based in Finland or elsewhere. Most of these professors are affiliated with UTU. There is also a campus coordinator who is responsible for all issues at the FTLab. Staff members at UNAM also assist by co-teaching workshops and are involved in community projects. Local and international industry partners also give seminars at the FTLab.

*Design-Reality gap*

The observation in the study shows that there appears to be a good match between the design expectation and reality in terms of staffing and skills at the FTLab given the number of students currently registered. However, there is a need for more lecturers to join if the number of students increases.

**Milieu (the external political, economic, socio-cultural, technological, and legal environment)**

*Design Expectations*

The qualification offered at the FTLab is offered following the European standards. Legal and political contexts in Namibia are not violated.

*Reality*

The success of the FTLab is multifactorial, and the financial, cultural, and environmental factors are discussed in [3]. For the legal and political context, the government of Namibia supports the implementation of satellite campuses and collaborations between foreign and Namibian universities. Since these degrees will be accorded by the base university, there is no need to involve the national accreditation body in Namibia to accredit the degree programmes. However, upon completion, students are expected to submit their qualifications for evaluation with the Namibia Qualification Authority, a body responsible for accrediting foreign qualifications [27].

*Design-Reality gap*

The design expectation and reality in terms of the milieu of the FTLab are met. There are no graduates of the plug-in campus yet. CER for designing CE solutions for the economic and digital freedom of computing graduates needs to be stimulated to serve the communities better. In our view, we perceive Africanization, e.g. in CER, as an enriching approach, rather than designing applications that are not for the African reality, as sometimes has been happening.

## 6 Discussion

Computing education at most African universities follows the IEEE and ACM curriculum, a curriculum developed to solve problems for developed countries [1]. With the way computing education is currently taught at universities in the GS, there is a need to reform the curriculum to train graduates who will get inspiration from local challenges and learn to solve problems using the best expertise available worldwide [28]. We adopted DSR to find a solution that better fits to contextualise CE and in turn enhance CER in the GS. We also adopted the DRG model to provide lessons learned during the first 3 years since the establishment of the FTLab. We can observe a cornucopia of new challenges and opportunities which – when attended to with imagination, curiosity but also hard work over decades of cross-cultural collaboration – can transfer the field of CER onto a new level of relevance, also globally, inspired and cross-fertilized by the dialogue between the CER communities in the Global South and the Global North. We summarise the agenda below.

*Contextualization* One of the key lessons we have learned is the importance of contextualization of Computing education. While the theory of computing is

universal and context-independent, the users of the artefacts created by computer scientists are local with their related requirements and demands, based on their everyday milieu. This is the reason that the imported Computing curricula, especially when they are not rethought of in the context, do not work, but rather lead to queues of unemployable computing graduates. The conventional process of first importing (I) a Global North curriculum to the South, then transferring (T) it to a given place, applying (A) the learning contents by localising, and last and usually least contextualising (C) whenever there is time and resources left [29] could be considered. The main effort is in the I phase, also called education export, and the least is left for the C stage. Vesisenaho (2006) [29] has turned the concept around into his CATI model, where the primary emphasis is on contextualization. Only two universities in southern Africa: the University of Pretoria and the University of Johannesburg have adopted the CATI model while the rest of the universities follow the traditional approaches [12].

In Namibia and the rest of the continent, contextualization can be called Africanization [30]. But unlike the occasional, dismissive use of the concept, leading for example to reducing requirements and teaching programming in Africa in a superficial way, Africanization means taking the continent's exceptional while still much-hidden talent pool seriously and, thus, extending, deepening, and reforming CER by the challenges and potential of Africa, its people, and cultures. A process of Africanizing CE might also involve ethnocomputing [22], or finding an alternative entry point to computing, without sacrificing the discipline's core.

*Curriculum reform* The current Computing curricula in most universities in the GS are theory-based adaptations of the universal Computing curriculum, and students' learning outcomes are mostly measured by conventionally written examinations, emphasising memorization. Among other obstacles preventing the much-needed transformation towards problem-based curricula, it seems that the universities' financial challenges, teachers' heavy teaching loads, and latent loyalty to international curricula as well as the massive number of students have made faculty very conservative and resistant toward modern teaching and learning methods. However, based on the observations of the courses offered at the plug-in campus, students learn and get excited when solving real-life problems. This is evident in the courses in robotics [10].

*CER in informal and non-formal CE* Besides conventional CE degree programmes, the challenges of the highly demanding settings in the GS call for alternatives. Besides professional development courses, various micro-credentials have been suggested as informal and non-formal CE solutions in the GS. Instead of a degree-oriented approach, students can learn competencies that they want in the order they see fit, and, in some cases, the process can end up with a highly individualised degree. Interestingly, due to the fast-growing and fast-changing competencies, micro-credentials have also gained importance in the European Union, indicating the global importance of the CER in the GS. However, in the GS, micro-credentials might be frugal pedagogical innovations, with micro-loans as their financial paragon.

*Off-shoring* One of the key challenges for CE and CER is the employment of graduates. For example, major international companies increasingly employ software engineers in India; the arrangement is referred to as off-shoring. One of the FTLab PhD students is devising approaches by which Namibia could be an offshoring destination, but for SMEs of the GN.

*Fast-tracking learning* The luxury of studying K-12 years at pre-primary, primary, and secondary school and attending university for the following 4–10 years is not possible for most young people in Namibia, and even less in most other countries in the GS. The GN model for education does not work, and neither does CER, which assumes the northern educational structures, principles, and practices. At the same time, societies in the global south require well-educated and trained employees, experts, and entrepreneurs to advance the field of CE. This means that the CER community needs to come out from their zones protected by well-funded institutions and radically imagine, invent, and innovate fast-tracked learning for computing.

*Devising future technologies* The whole concept of the plug-in campus is based on the expectation that a university can grow to an increasingly relevant form when it is located outside its original milieu. The COVID-19 pandemic showed the nonnecessities of a traditional university: much of lecture rooms and other physical facilities were not required but could be replaced by their digital counterparts. These observations and experiences that were imposed by the pandemic chased the university community outside their former comfort zone. Concepts such as the metaverse [12] and remote presence would pave the way toward novel solutions in this direction.

As of the interdisciplinary stage behind CER, the requirements from the GS call for an extension. Relevant and meaningful CE and, hence, CER in the GS requires close collaboration beyond the three current academic fields of Computing, Cognitive science, and Education. Contributions from business studies, social science, and cultural and development studies are critical.

*Shortcomings of the study* The GS certainly exceeds the boundaries of Namibia and, therefore, our results cannot be generalised to the whole South. However, DSR always starts from an identified but limited environment for ensuring concrete results. In the future, we intend to extend our environment to a set of diverse environments from the Global South, thus also extending the DSR methodology.

Secondly, our design cycle did not include the critical build task, but only came up with a list of suggestions to enhance the contemporary CER agenda, reflecting the requirements of the GN. However, Heeks's tools [1] for the design gave the scheme to integrate evaluation within the design task.

## 7 Conclusion

We presented how the Future Technology Lab (FTLab), a collaboration between the University of Turku, in the Global North, and the University of Namibia, in the Global South, responded to the computing education and computing education research respectively in the Global South. We applied the research methodology following Hevner's Design Science Research: relevance, rigor, and design cycles without really reaching the design stage to understand the local environment and existing knowledge on computing education and computing education research in the Global South. We applied Heeks's Design Reality Gap as a tool to show the discrepancy between a plug-in campus initiative and the realities in the Global South. Since this was a longitudinal analysis, the process is ongoing and thus cannot be seen as complete. The gap was only identified from activities at the FTLab since its inception in 2019.

Although all eight dimensions of the Design Reality Gap presented challenges in our study, there appears to be a relatively good match between the design expectation and the reality at the FTLab. Based on the results of the study, employment of graduates is one of the key challenges for CE and CER in the Global South, hence a need to move away from importing the curriculum and instead reform and fast-track the curriculum. Metaversity and remote presence concepts could replace traditional universities to pave the way toward novel solutions. Shifting universities online could enhance collaborations between both local and international universities and industry, hence opening collaboration opportunities and exposing students to offshoring.

**Acknowledgments** We would like to acknowledge the designers of the concept tree of the plug-in campus. The concept tree was designed by Tuula Kaisto with the assistance of Maria Ntinda, Hilma Aludhilu, Sebulon David and Erkki Sutinen. We would also like to thank the industry and the community involved in different projects at the plug-in campus.

## References

1. J. M. Bass and R. Heeks, "Changing Computing Curricula in African Universities: Evaluating Progress and Challenges via Design-Reality Gap Analysis," *Electron. J. Inf. Syst. Dev. Ctries.*, vol. 48, no. 1, pp. 1–39, Aug. 2011, https://doi.org/10.1002/J.1681-4835.2011.TB00341.X.
2. S. Dasuki, S. Dasuki, P. Ogedebe, R. Kanya, H. Ndume, and J. Makinde, "Evaluating the Implementation of International Computing Curricular in African Universities: A Design-Reality Gap Approach," *Int. J. Educ. Dev. using ICT*, vol. 11, no. 1, 2015.
3. M. N. Ntinda, T. K. Mufeti and E. Sutinen, "Plug-in campus for Accelerating and Catalyzing Software Engineering Education in the Global South," 2020 IEEE Frontiers in Education Conference (FIE), Uppsala, Sweden, 2020, pp. 1–4, https://doi.org/10.1109/FIE44824.2020.9274200.
4. L. Malmi, J. Sheard, Simon, R. Bednarik, J. Helminen, P. Kinnunen, A. Korhonen, N. Myller, J. Sorva, and A. Taherkhani. 2014. Theoretical underpinnings of computing education research: what is the evidence? In Proceedings of the Tenth Annual Conference on International

Computing Education Research (ICER '14). Association for Computing Machinery, New York, NY, USA, 27–34. https://doi.org/10.1145/2632320.2632358
5. C. Miller-Idriss and E. Hanauer, "Transnational higher education: Offshore campuses in the Middle East," *Comp. Educ.*, vol. 47, no. 2, pp. 181–207, May 2011, https://doi.org/10.1080/03050068.2011.553935.
6. J. Guimón, "Promoting university-industry collaboration in developing countries," researchgate.net, 2013, Accessed: Apr. 12, 2022. [Online]. Available: https://www.researchgate.net/profile/Hazim-Tahir/post/How_can_we_improve_academia-industry_collaborations_to_improve_quality_of_education_and_research/attachment/59d6580979197b80779ae1d4/AS%3A536712578777088%401504973658352/download/PromotingUniversityInd
7. A. Hevner, "A Three Cycle View of Design Science Research," *Scand. J. Inf. Syst.*, vol. 19, no. 2, Jan. 2007, Accessed: Feb. 01, 2022. [Online]. Available: https://aisel.aisnet.org/sjis/vol19/iss2/4
8. "In Namibia, A New Generation Learns How to Use Robotics for Social Good – Flying Labs Blog." https://blog.flyinglabs.org/2021/12/14/in-namibia-a-new-generation-learns-how-to-use-robotics-for-social-good/ (accessed May 01, 2022).
9. M. Ntinda, M. Apiola and E. Sutinen, "Mind the Gap: Aligning Software Engineering Education and Industry in Namibia," 2021 IST-Africa Conference (IST-Africa), South Africa, South Africa, 2021, pp. 1–8.
10. E. Sutinen and L. Uwu-khaeb, "Connecting Creative Continents – UTU @ WINDHOEK," Mar. 2022. https://ftlab.utu.fi/ (accessed Mar. 17, 2022).
11. A. Shipepe, L. Uwu-Khaeb, E. A. Kolog, M. Apiola, K. Mufeti, and E. Sutinen, "Towards the Fourth Industrial Revolution in Namibia: An Undergraduate AI Course Africanized," *Proc. – Front. Educ. Conf. FIE*, vol. 2021-October, 2021, https://doi.org/10.1109/FIE49875.2021.9637356.
12. V. Ruwodo, A. Pinomaa, M. Vesisenaho, M. Ntinda and E. Sutinen, "Enhancing Software Engineering Education in Africa through a Metaversity," 2022 IEEE Frontiers in Education Conference (FIE), Uppsala, Sweden, 2022, pp. 1–8, https://doi.org/10.1109/FIE56618.2022.9962729.
13. F. Rossi, V. Goglio (2020) Satellite University Campuses and Economic Development in Peripheral Regions, Studies in Higher Education, 45:1, 34–54, https://doi.org/10.1080/03075079.2018.1506917.
14. S. Nikolic and W. Li, "Facilitating student and staff engagement across multiple offshore campuses for transnational education using an immersive video augmented learning platform," *Proc. 2016 IEEE Int. Conf. Teaching, Assess. Learn. Eng. TALE 2016*, pp. 77–81, Feb. 2017, https://doi.org/10.1109/TALE.2016.7851774.
15. EUA, "Do satellite campuses contribute to the access of higher education in Europe?," 2019. https://eua.eu/partners-news/333-do-satellite-campuses-contribute-to-the-access-of-higher-education-in-europe.html (accessed Apr. 13, 2022).
16. P. Langa and E. Mondlane, "A disjointed multi-campus system: the neo-liberal expansion and fragmentation of Mozambican higher education," *Tert. Educ. Manag. 2016 231*, vol. 23, no. 1, pp. 23–40, Mar. 2017, https://doi.org/10.1080/13583883.2016.1214286.
17. C. L. Hoover, M. Shaw, and N. R. Mead, "The Carnegie Mellon University Master of Software Engineering specialization tracks," in *Proceedings of 9th Conference on Software Engineering Education*, pp. 100–118. https://doi.org/10.1109/CSEE.1996.491366.
18. "South Africa – Monash University." https://www.monash.edu/about/our-locations/featured-items/row-6/south-africa (accessed May 01, 2022).
19. "Limkowing London Campus @ Limkokwing University of Creative Technology." https://www.limkokwing.net/united_kingdom/about/campus/ (accessed May 01, 2022).
20. "Botho University Namibia." https://namibia.bothouniversity.com/ (accessed Apr. 30, 2022).
21. M. Lahti, S. P. Nenonen, and E. Sutinen, "Co-working, co-learning and culture – co-creation of future tech lab in Namibia," *J. Corp. Real Estate*, 2021, https://doi.org/10.1108/JCRE-01-2021-0004/FULL/PDF.

22. E. Sutinen, M. Vesisenaho. Ethnocomputing in Tanzania: Design and Analysis of a Contextualized ICT Course. Research and Practice in Technology Enhanced Learning, 2006, 01:03, 239–267. https://doi.org/10.1142/S1793206806000238
23. S. Heleta, "Decolonisation of higher education: Dismantling epistemic violence and Eurocentrism in South Africa," *Transform. High. Educ.*, vol. 1, no. 1, Oct. 2016, https://doi.org/10.4102/THE.V1I1.9.
24. UNESCO, "Towards universal access to higher education: international trends," Nov. 2020. https://unesdoc.unesco.org/ark:/48223/pf0000375686/PDF/375686spa.pdf.multi (accessed Mar. 17, 2022).
25. University of Turku, "Projects | UTU @ Windhoek," 2019. https://ftlab.utu.fi/projects (accessed Apr. 08, 2020).
26. N. Pope, M. V. Apiola, H. Salmento, N. Islam, M. Lahti, and E. Sutinen, "The Latest in Immersive Telepresence to Support Shared Engineering Education," *Proc. – Front. Educ. Conf. FIE*, vol. 2020-October, Oct. 2020, https://doi.org/10.1109/FIE44824.2020.9274106.
27. NQA, "Namibia Qualifications Authority – Welcome," Jan. 2022. http://www.namqa.org/ (accessed Jan. 30, 2022).
28. A. E. Van Der Poll, I. Van Zyl, and J. Kroeze, "Towards Decolonization and Africanization of Computing Education in South Africa," Apr. 2020, Accessed: Aug. 19, 2022. [Online]. Available: https://uir.unisa.ac.za/handle/10500/26361
29. M. Vesisenaho, J. Kemppainen, I. Sedano, C. Tedre, and M. Sutinen, "Contextualizing ICT in Africa: The Development of the CATI model in Tanzanian Higher Education," *African J. Inf. Commun. Technol.*, vol. 2, no. 2, pp. 88–109, Jul. 2006, https://doi.org/10.5130/ajict.v2i2.23.
30. M. W. Malegapuru W. Makgoba, "Mokoko: the Makgoba Affair: a reflection on transformation," Florida Hills, R.S.A. : Vivlia Publishers, p. 243, 1997.

# Computing Education Research in Baltic Countries


**Valentina Dagienė, Mart Laanpere, and Juris Borzovs**

## 1 Introduction

Very similar school systems were built in all three Baltic countries during the Soviet occupation (1940–1991), where the dominating school type was 8- or 9-year basic school and the remaining one-third of schools taught children from age 7 up to 19. Also, computing education was strictly standardized in all Soviet republics. After the breakdown of Soviet Union in 1991, the three Baltic countries followed slightly different paths, both in regard to modernizing their school system and their computing education. Despite the low status of computer science as an elective subject in Estonian schools today, successful tertiary education in computing has allowed Estonia to become a leading country in digitalization. A strong foundation to computing education on tertiary level has been laid by high quality mathematics and science education in schools. In 2018, Estonia ranked number one in Europe in all three domains of assessment in PISA: Reading, Science, and Mathematics. Latvia has its own unique developmental trajectory of Computing education, focusing on local job market needs. Compared to the two other countries, research on computing education has been significantly more extensive in Lithuania, which is demonstrated by the establishment of the international research journal "Informatics in Education" by Vilnius University in 2002. Lithuania has initiated and coordinated

V. Dagienė (✉)
Vilnius University, Vilnius, Lithuania
e-mail: valentina.dagiene@mif.vu.lt

M. Laanpere
Tallinn University, Tallinn, Estonia
e-mail: martl@tlu.ee

J. Borzovs
University of Latvia, Riga, Latvia
e-mail: juris.borzovs@lu.lv



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_14

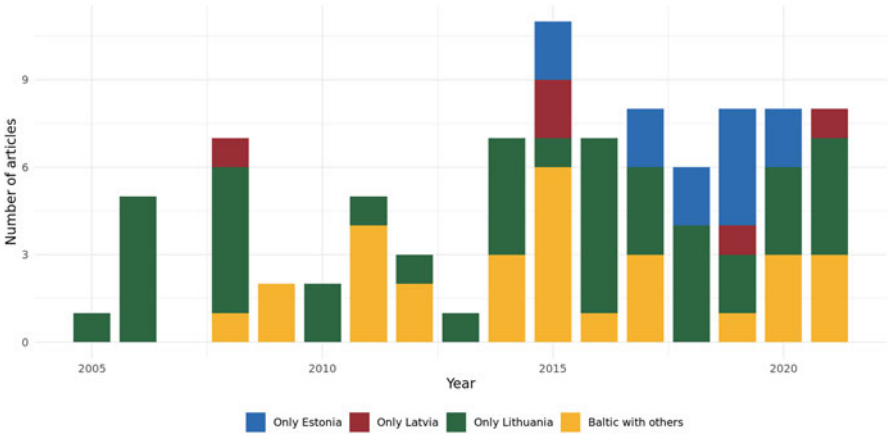


**Fig. 1** Distribution of CER papers where the publications are either by a single Baltic country author or include also authors from outside, but apparently they never include authors from multiple Baltic countries and no outside ones

the Doctoral Consortium on CER in schools, as well as the Bebras Challenge, a challenge on Informatics and Computational Thinking, attracting millions of school students globally every year [1]. A lot of research publications have been published with the data collected by Bebras.

A scientometric analysis revealed 89 papers that contained at least one author from Baltic countries (Fig. 1). The bibliometric dataset employed was a subset of the computing education research dataset described in chapter 'Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research' of this book [2], including only those articles with at least one author affiliated to a Baltic institution at the moment of publication.

According to scientometric analysis, most cited papers with at least one author from Baltic countries are presented in Table 1 and ten most cited authors in Fig. 2.

But let us begin this chapter with something that has been in common for Computing Education Research in Estonia, Latvia and Lithuania, namely: the emergence of Computing Education in 1986–1991, and the Baltic Olympiads in Informatics.

## *1.1 Computing Education in Three Baltic Countries: Prehistory*

Tertiary education on computing started in Estonia, Latvia and Lithuania in 1960. The movement was led by local enthusiasts who have been studying the emerging discipline in the best universities of St. Petersburg (former Leningrad) and Moscow. While there were attempts in all three countries to build local computer hardware

**Table 1** The top cited papers where at least one author is from Baltic countries

| Title of paper | Authors | Authors' countries | Year | Cit. index |
|---|---|---|---|---|
| Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies [3] | Ihantola P; Vihavainen A; Ahadi A; Butler M; Börstler J; Edwards S; Isohanni E; Korhonen A; Petersen A; Rivers K; Rubio Ma; Sheard J; **Skupas B**; Spacco J; Szabo C; Toll D | Australia; Canada; Finland; **Lithuania**; Spain; Sweden; US | 2015 | 147 |
| Computational Thinking in K-9 Education [4] | Mannila L; **Dagiene V**; Demo B; Grgurina N; Mirolo C; Rolandsson L; Settle A | Finland; Italy; **Lithuania**; Netherlands; Sweden; US | 2014 | 125 |
| Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks [5] | **Dagiene V**; Futschek G | Austria; **Lithuania** | 2008 | 94 |
| A Global Snapshot of Computer Science Education in K-12 Schools [6] | Hubwieser P; Giannakos Mn; Berges M; Brinda T; Diethelm I; Magenheim J; Pal Y; Jackova J; **Jasute E** | Germany; India; **Lithuania**; Norway; Slovakia | 2015 | 68 |
| Bebras – A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking [7] | **Dagiene V; Stupuriene G** | **Lithuania** | 2016 | 66 |

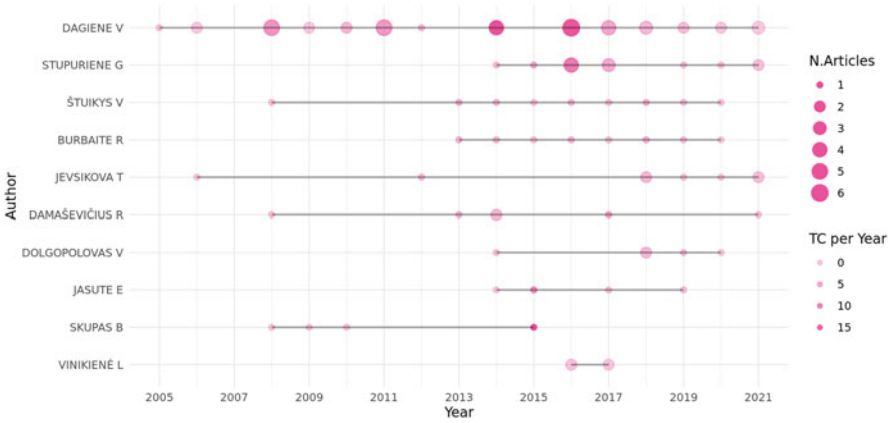


**Fig. 2** The most productive authors from the Baltic countries (actually all of them are from Lithuania), TC means "Total Cites"

and software, the first programming courses ran on Soviet factory-built machines (Minsk, Ural, *Edinaja Sistema* – ES). Computing was taught mainly as a part of applied mathematics and engineering study programs, later also in economics.

The starting point for introducing Computing Education in secondary schools in all three Baltic republics was Mikhail Gorbachev's talk on the 27th Congress of the Communist Party of the Soviet Union (March 1986), where the new leader made a radical call for "acceleration of the scientific-technological process" through the computerization of the Soviet economy. He saw this as the only chance for competing with "capitalism of the age of electronics and informatics, of computers and robots" [8]. Actually, M. Gorbachev had launched an ambitious computer literacy program a year before (ibid). Then, all three Baltic countries rushed to develop and produce their own local school computers [9]: Juku in Estonia (1986), VEF Mikro in Latvia (1985) and Sigma Poisk in Lithuania (1987). These locally-built computers were used in parallel with Russian-made school computers BK-0010SH, Agat and Korvet, but also imported Yamaha YIS-503IIIR machines. However, the new school subject called Informatics that was rolled out in all schools of the Soviet Union in 1986, did not assume the use of computers at all – the content was theoretical. The author of the first informatics textbook for secondary schools was the leading Soviet computer scientist, academician Andrei Ershov, who was invited by M. Gorbachev to lead the "informatization" of his perestroika project. Unfortunately, this progressive campaign for achieving universal computer literacy collapsed even before the Soviet Union and A. Ershov's difficult textbook remains in the memory of teachers and students of that era as an example of "Soviet absurdity" [8]. Already from the beginning of the 1990s, all three Baltic countries redirected their school informatics towards using Western IBM PC compatible computers for teaching more practical computing skills.

## *1.2 Baltic Olympiads in Informatics*

The Baltic Olympiad in Informatics was established as a joint initiative of Estonia, Latvia, and Lithuania in 1995. A few years later, it grew to include seven more countries around the Baltic Sea: Denmark, Finland, Germany, Iceland, Norway, Poland, and Sweden [10]. In addition, teams from other countries may be invited as guests. The main goals concentrate on providing the participating students with experience of an international Olympiad, encouraging communication and exchange of ideas between the developers of national contests in informatics, as well as assisting delegation leaders in selecting participants for the international Olympiad.

The Baltic Olympiad in Informatics (BOI) is a programming contest for secondary school students from countries around or close to the Baltic Sea. Each year approximately 60 school students from 10 countries compete against each other, solving difficult problems of algorithmic nature. Each participating country sends 6 contestants from their national Olympiads organized beforehand.

BOI shares its competition format with the International Olympiad in Informatics (IOI), which is the most prestigious annual world programming competition for secondary school students established in 1989 (see chapter 'Computing Education Research in Schools', section 'Extracurricular Activities'). On each of two contest days, the contestants participate in a 5-h exam. They are given a number of algorithmic problems and are required to solve these by writing computer programs. Their programs are then evaluated and scored based on both efficiency and correctness. Usually, the participating countries take BOI results into account when selecting their teams for IOI.

The BOI is a short-term (lasting 3–4 days) and inexpensive event. It can be distinguished for cozy and good neighboring atmosphere, which is highly important when motivating students for self-help. Even though the BOI is a mini model of the IOI, it has significant differences from the cultural and learning perspectives. The organization of the scientific part of BOIs is based on mutual trust of the participating countries. The leaders of all the participating countries offer problems for the nearest BOI. At first draft task texts are offered, then the ideas are exchanged via e-mail and discussed, and some problems are rejected, while other problems are suggested to be modified and later are accepted.

Most of the problems are translated to the native languages by the leaders before going to the Olympiad. This is a unique possibility for country representatives to gain experience in organizing the scientific part of a relatively small international Olympiad as well as to raise their qualifications in algorithms.

The organizers of BOIs try to follow as close as possible the newest IOI trends in problem types, compilers, platforms, and contest systems. Even though all the tasks are of an algorithmic nature, they represent cultural and methodical differences. Since in the BOI most of the preparatory work has been done in advance, team leaders can discuss the tasks, possible solutions, and technical issues, and the BOI can be considered as a prearranged international way of learning.

The Baltic Olympiad in informatics has a long and interesting history. There are many activities in connection to BOI: countries have been preparing brochures of the used problems with detailed programming solutions, write papers about interesting task cases or testing environments, provide discussion on programming languages or testing innovations, and conduct various studies on programming education [10]. BOI has brought a message to society that informatics and programming are important for young people, and that they can be smart. There are many people involved in hosting Olympiads: companies, startups, researchers, professors, teachers, students, and policy makers.

List of venues and dates of the Baltic Olympiads in Informatics:

Lübeck, Germany, April 2022
Lübeck, Germany, April 2021 online
Ventspils, Latvia, July 2020 online
Tartu, Estonia, April 2019
Stockholm, Sweden, April 2018
Bergen, Norway, April 2017
Helsinki, Finland, May 2016
Warsaw, Poland, March 2015
Palanga, Lithuania, April 2014
Rostock, Germany, April 2013
Ventspils, Latvia, May 2012
Lyngby, Denmark, April 2011
Tartu, Estonia, April 2010
Stockholm, Sweden, April 2009
Gdynia, Poland, April 2008
Güstrow, Germany, April 2007
Heinola, Finland, May 2006
Pasvalys, Lithuania, May 2005
Ventspils, Latvia, April 2004
Tartu, Estonia, April 2003
Vilnius, Lithuania, April 2002
Sopot, Poland, June 2001
Haninge, Sweden, July 2000
Riga, Latvia, April 1999
Tartu, Estonia, June 1998
Vilnius, Lithuania, April 1997
Riga, Latvia, April 1996
Tartu, Estonia, April 1995

In 1996–1998, the Baltic School of Algorithmization, a distance teaching project for school students was initiated by the Institute of Mathematics and Informatics of Lithuania. School students from Estonia, Latvia and Lithuania participated in it. The purpose of this project was to teach high school students of the three countries competitive programming skills, and to compete and compare their skills with those

of the students from other countries. Over one hundred school students participated in the Baltic School of Algorithmization.

All learning took place through e-mail. Each country picked one topic (non-overlapping), prepared the learning material, and created five tasks from the topic, e.g. recursion, backtracking, combinatorics, or big numbers. Each country translated the learning material and the tasks to their own language. Student submissions were evaluated by the country which created the task. After testing the solutions, the task authors prepared an overview of the received submissions and sent them out together with sample solutions and data sets. The School was completed with a competitive programming competition by email. Each country prepared one task for the competition. During the project, the students mainly communicated in their native languages through project coordinators in their countries.

## 2 CER in Estonia

### *2.1 Prehistory*

The first computer that was used for educational purposes in Estonia was switched on in the University of Tartu in 1959: it was Soviet-made Ural 1. The first students of computational mathematics with programming skills graduated a year later, their teachers had studied computer science in Leningrad and Moscow [11]. Six years later, the same Ural 1 computer was moved to the nearby Nõo Secondary School, where it became the first school computer in the whole Soviet Union. For the next 20 years, it remained the only school computer in Estonia. Nõo Secondary School was not an ordinary one: it was a boarding school for mathematically gifted children who have excelled in math competitions and invited to take part in an experimental study program focusing on math, computing and science.

In Estonia, the M. Gorbachev's call for providing universal computer literacy resulted in the development of the locally designed school computer Juku, whose mass production of which started in 1988 [9]. The last computer lab with 20 functioning Juku computers remained functional at Nõo Secondary School until 1997. The informatization campaign resulted in the preparation of qualified informatics teachers that first started in Tallinn Pedagogical Institute (from 1987, the first cohort graduated in 1989) and a few years later also in the University of Tartu.

However, there was almost no research conducted on computing education within these first three decades (1960–1990). The only computer scientist interested also in educational aspects of teaching with and about computers was Ustus Agur in Tallinn Polytechnical Institute, who co-authored with Inge Unt and Kalju Toim (educational scientists) an extensive book on Programming Learning [12]. The first Estonian who pursued a doctoral degree specifically in Computing Education Research was Mati Maksing from Tallinn Pedagogical University, but he did not remain in academia after his PhD studies in Germany in mid-1990s. At the same time in Tartu, Eno

Tõnisson was leading a group of teachers experimenting with Turtle Logo in primary schools. Since 1994, when internet arrived in first 5–10 schools in Estonia, Anne Villems was organizing in the University of Tartu several online simulation games for upper-secondary school students [13].

The third period of computing education in Estonia started in 1996 after launching the new national curriculum and also the national school computerization program "Tiger Leap". While the latter resulted with equipping all schools with modern computer labs, internet connectivity and teacher training by year 2000, the former redefined completely the goals and content of the computing education in the country. The new concept of computing education did not include any flowcharts, programming, algorithms or data structures that were the core of A. Ershov's textbook. Instead, the new subject called Informatics was focusing on generic ICT skills: word processing, spreadsheets, presentations, internet. Another major difference from A. Ershov's concept was that the new Informatics could not be taught without computers. As a few schools had full-size computer labs at the time, the subject remained elective and marginal. By 2001, when the Tiger Leap program had modernized the computing infrastructure in all schools, the new version of national curriculum was published without mentioning any computing-related subjects or topics. To compensate this loss, there was a recommendation to integrate development of ICT skills into other subjects [14]. From 2003 until 2005, the national test of ICT skills was conducted by the National Examination Centre, the results were researched by A. Villems and L.-M. Tooding (2006) [15]. The test did not include any programming tasks. Within the next decade, the universities adjusted their computer science study programs and related entry requirements in line with an assumption that the majority of students enter university without any knowledge or skills of programming. For instance, Tallinn University of Technology introduced the introductory coding course based on MS Visual Basic in 1998 [16].

In 2011, the next major curriculum reform took place and informatics made its way back as an elective subject in primary and lower secondary school, still excluding coding, algorithms and other elements of traditional computing education. Although upper-secondary school curriculum did not mention informatics, it introduced several new elective courses that were related to computing: "Using computers for inquiry", "Basics of programming and software engineering", "Robotics and mechatronics", "3D-modeling", "Geoinformatics". Each of these new elective courses was accompanied by an original textbook, online courseware, videos and other digital learning resources. However, uptake of these new courses remained limited due to lack of teachers – less than 20% of upper-secondary schools offered these courses to their students in 2017 [17]. The only exception was the data analysis course ("Using computers for inquiry", taught in more than 80% of schools), as it appeared to be a necessary preparation for compulsory inquiry project every high school graduate has to defend before graduation.

Another round of curriculum innovation is currently (2022) being finalized, introducing significantly the renewed, widened and deepened concept of school informatics. This concept is balancing Design Thinking and Computational Thinking, as it introduces coding with educational robots already from Grade 1. From

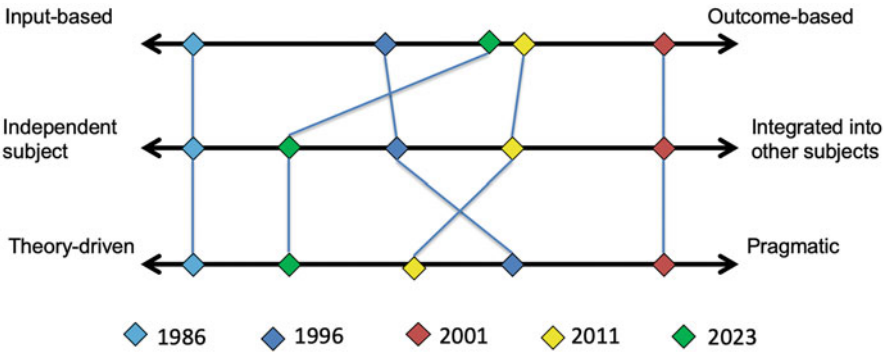


**Fig. 3** The pendulum effect in Estonian school informatics curriculum

Grade 4 onwards, it recommends switching to coding with Scratch and Kodu, while also addressing digital safety topics. In Grade 7, the students are expected to learn about information society technology: digital signature, information systems, online communities, Internet of Things, Virtual/Augmented Reality, Robotics, Big Data, Cyber Security. All students are expected to complete a Creative Project in Grade 8, which can be conducted in a new format: a Digital Prototyping Project that applies above mentioned new technologies. The e-textbooks matching the new informatics curriculum for Grades 1–6 and 10–12 are already available, Grades 7–8 are still waiting for it. Radical changes in Estonian informatics curriculum between 1986 and 2022 created a pendulum effect (Fig. 3).

The swift departure from Soviet-style input-based, theoretical curriculum resulted with replacing informatics as a school subject with integrated teaching of ICT usage skills through other subjects in 2001. Then, dissatisfaction of various stakeholders with this situation pushed the pendulum back towards restoring the informatics subject and re-introducing coding and computational thinking as its core in 2022. Unfortunately, this latest curriculum change process has been only briefly captured by research, specifically by K. Salum et al. [18] and P. Niemelä et al. [19].

## *2.2 Informal Computing Education*

Increasing availability of personal computers in schools and informal education centers enabled the growth of coding clubs across Estonia at the end the 1980s and the beginning of the 1990s. Fidonet was launched in Estonia in 1989, from 1991 this network attracted a rapidly growing community of computing enthusiasts who also met physically for so-called informal BBSummer events. BBSummer'93 was attended by 140 participants. Fidonet Bulletin Board Systems remained hubs of informal computing education until the Internet replaced it from 1994 onwards.

A very important role in promoting informal, but systematic computing education was played by the Science School of the University of Tartu, where the department of school informatics was established by Rein Prank in 1993. Printed computing tasks were mailed to registered secondary school students, who then returned their solutions for review. The first online course "Java Programming" was launched by Tartu Science School in 2003. Today, 98% of their courses are delivered online. It is difficult to estimate the effect of correspondence courses delivered by Tartu Science School to the promotion of computing education and careers among secondary school students within the last three decades. Unfortunately, these important developments in computing education were not addressed by Estonian researchers.

While informatics and coding disappeared from the national curriculum by the turn of the century, computing activities remained active in 10–15% of Estonian schools through informal education. Extracurricular activities were supported financially by Estonian IT and banking industry through Look@World Foundation (e.g., the "SmartLabs" programme involved more than 600 students annually in 2012–2018), but also by the National Agency of IT in Education (HITSA) who conducted the "ProgeTiger" programme in 2014–2021 [20]. More than half of Estonian primary and lower secondary schools participated in the "SmartLabs" and "ProgeTiger" programs.

## *2.3 Research on School Informatics*

The first Estonian academic journal that addressed the issues of computing education research was "Arvutustehnika ja andmetöötlus" (Computing Technology and Data Processing) that was published by Estonian Institute of Information from 1987 till 1992, then by Mainor Ltd. between 1992 and 1997. From 1998 till 2009 this journal was published by Tallinn University of Technology under the name A&A, chief editor was prof. Paul Leis. The section editor and most active contributor in the computing education section of this journal was prof. Leo Võhandu from Tallinn University of Technology, who discussed approaches to teaching coding based on various languages, but also interdisciplinary applications of computing education: computational linguistics, data visualization, geoinformatics, mathematics of computation, etc. The Estonian computing education research community is very small and has not been able to set up a new local journal that would address computing education topics. Rather, Estonian computing education researchers tend to publish in international journals, less frequently in the Baltic Journal of Modern Computing (published in Latvia) or Informatics in Education (published in Lithuania).

In the early years, the research on computing in general was conducted in four locations:

- Electrotechnical Research Institute (est. 1958) and,

- Institute of Cybernetics (est. 1960), both belonging to Estonian Academy of Sciences.
- Tallinn Polytechnical Institute, where the department of computing technology was established in 1964 by lecturer Kaarel Allik.
- University of Tartu, where the department of statistics and programming was opened in 1969 by prof. Ülo Kaasik.

In 1980, the computing center was established also in Tallinn Pedagogical Institute, contributing to development of educational computing tasks and learning resources.

Today, there are three research groups in the Computing Education field in Estonia. The largest and oldest one is led by associate professor of didactics of informatics Piret Luik in the Institute of Computer Science, University of Tartu. Their research focuses mainly on motivation and dropout of students taking a MOOC on programming [21], assessment of Computational Thinking [22], and automated assessment of programming tasks [23].

Another research group is led by Mart Laanpere, professor of mathematics and computing education in the Centre for Educational Technology, Tallinn University. Their research addresses mainly modelling and assessment of digital competence [24], digital transformation in schools [25], learning design, and learning analytics [26].

The third research group is led by senior researcher Birgy Lorenz in the Centre for Digital Forensics and Cyber Security at TalTech. Their research interests are related to cyber security education and contests, and gender issues in computing education [27].

These research groups have hosted a few conferences related to CER in Estonia: ICALT 2019, EC-TEL 2017, ISSEP 2020. For the last 4 years, the National Agency of Education and Youth has financially supported the development of informatics teacher education in Tallinn and Tartu, informatics teachers' summer schools and winter conferences, a MOOC on programming for secondary school students and the publishing of seven new e-textbooks of school informatics.

## 3 CER in Latvia

### *3.1 Prehistory*

The beginning of the computer era in Latvia is considered to be 1957, when the Latvian State University docent Eižens Āriņš raised the need for an electronic digital computing machine for scientific and technical development in Latvia. In November of 1959, the Computing Centre of the State University of Latvia was founded, with E. Āriņš as its director. The first serial computer in Latvia БЭСМ-2 was launched in a Computing Centre in 24-h mode in April 1961.

In the meantime, computer construction was also developing in Latvia. In 1960, under the leadership of Jānis Daube, a computer LM-3 was built. LM-3 was

commissioned in July 1960, the first computer in the Baltic States. However, this direction did not gain further development and constructors of the LM-3 moved to the University of Latvia's Computer Centre or to the newly established Institute of Electronics and Computing Engineering of the Latvian SSR Academy of Sciences. LM-3 was dismantled in 1964.

The first programming textbook for computing machines (in Russian) was published by Ilze Irēna Ilziņa in 1962. On 16 May 1963, the first programming textbook in Latvian was published by E. Āriņš, S. Hozioskis and V. Līnis. In 1968, БЭСМ-2 was used to teach programming in machine code as no programming language for this machine existed yet, and the teacher had to learn to program simultaneously with the students.

The teaching of programming was also launched in 1961 by the Faculty of the Riga Polytechnic Institute (now Riga Technical University) in Automatics and Computing Engineering study program.

Research into how to systematically teach and learn usage of computers was not considered at the time.

## *3.2 Computer Education Research Chronology in Latvia*

The terms computing (*datorika* in Latvian) and informatics (*informātika* in Latvian) are often used as synonyms (e.g., https://www.informatics-europe.org/). In the Latvian education system, it is well established that informātika covers high-quality usage skills of an existing software product (e.g., MS Word, Excel, Drive), while *datorika* in addition contains software development theory and practice. Thus, *informātika* mainly refers to primary and secondary education, while *datorika* refers to professional secondary and higher education. Of course, some elements of the *datorika* can also appear in secondary and even primary education.

Then, the computing education research should be understood both as how to teach *informātika* and how to study/teach *datorika*. These studies are inevitably based on and depend on the technical basis of the period concerned (Table 2).

In the context of CER, these activities were crowned in several doctoral theses: "Software to Support Some Topics in Basic Course of Informatics at General Secondary Schools" of M. Vītiņš in 1993, "Informatics at School" of V. Vēzis in 2005, and "Competencies based school computing education content" of O. Krūmiņš in 2022.

### 3.2.1 Decades of Ershov-Monachov-Vītiņš

The informatics subject was introduced in secondary educational establishments in the USSR in the school year 1985–1986 [28]. The objective of the course was to develop computing skills without using computers, due to a limited availability of hardware in schools at that time. When the supply of computers to schools

**Table 2** CER chronology in Latvia

| Year | Technology | Users | Domain | Education | CER |
|---|---|---|---|---|---|
| 1950–1965 | Few computers, small series | Professionals – University graduates | Military, science, economics | Universities: *datorika* | Prehistory |
| 1965–1995 | Serial computers for professional usage | Mass usage in professional fields, university graduates | Military, science, economics | Universities and some secondary schools: *datorika* | Decades of Ershov-Monachov-Vītiņš |
| 1995–2005 | Personal computers for professional usage | Mass usage in professional fields, university graduates | Military, science, economics, household | Universities and some secondary schools: *datorika* | |
| 2005–2015 | Global web and personal computers for professional and home usage | Mass usage in professional fields, secondary/tertiary education | All fields of economics, majority of households | Higher education: *datorika, informātika*; secondary schools: *informātika* | Decade of Vītiņš-Borzovs-Vēzis |
| Since 2015 | Global web and dominance of social networks | Mass usage in all sectors and households; non-ICT professionals | All fields of economics, majority of households | Higher education: *datorika, informātika*; all schools: *informātika* | Decade of Vēzis-Krūmiņš |

improved by the end of decade, the course was gradually transformed to make use of computers. The course syllabus was expanded and more lessons were allocated for the subject. Programming elements and working with office software still remained a part of the course. Latvia was the first from the former republics of the USSR to introduce a computer-based version of A. Ershov's informatics course in all secondary educational establishments.

### 3.2.2 Decade of Vītiņš-Vēzis

All Latvian schools had received at least one computer class around 2000 that enabled them to react constructively to the rapid spread of the global web. There was an objective need to train both adults and students to use the most common software products (Word, Excel, e-mail, etc.) qualitatively. It was organized by M. Vītiņš and J. Borzovs through the Latvian Information and Communication Technology Association (LIKTA) by introducing European Computer Driver License (ECDL) certification in Latvia [29]. A set of educators lead by V. Vēzis developed and published textbooks for preparing for ECDL exams. These initiatives launched a major campaign for the acquisition of informatics skills in Latvia. Latvia was the first country in the world to introduce the ECDL curriculum in both secondary and primary schools. These changes did not affect programming training, leaving it to the few schools that already practiced such a subject.

### 3.2.3 Decade of Vēzis-Krūmiņš

In the second decade of the twenty-first century, an acute shortage of computer-system developers appeared to be continuing. There was an objective need to teach not only informatics in schools, but also computer literacy. The pressure of the ICT industry lobby on the government resulted in a state-funded project designing new subjects in informatics and computing. This was done by the Working Group under the leadership of V. Vēzis and O. Krūmiņš [30]. They based on Association for Computing Machinery's recommendations, especially:

- digital literacy should be taught to anyone under 12 years of age, including not basic skills only, but also aspects of effectiveness, ethics and security;
- all students should study computing as a separate field of science, including by using it in other fields of study.

More detailed analysis for teaching informatics and computing at Latvian schools are provided in [31].

### *3.3 CER in Other Latvian Universities*

Although, due to historical circumstances, the part of the research aimed at creating informatics curricula for primary and secondary schools was carried out at the University of Latvia (see Sects. 3.2.1, 3.2.2, and 3.2.3), a number of other Latvian institutions of higher education also conducted CER.

At Riga Technical University since 1997, there is the Distance Education study Centre, led by Atis Kapenieks, where ecosystems for distance learning are explored and developed (collaborators I. Daugule, A. Gorbunovs, M. Jirgenson, K. Kapenieks, J. Kapenieks Jr. I Kudina, G. Stale, Z. Timsans, I. Vitolina, V. Zagorsky, B. Zuga and others).

Led by Anita Jansone at the University of Liepaja, D. Barute, I. Konarev, K. Mackare, I. Magazeina, R. Nacheva, L. Ulmane-Ozolina, M. Zigunov have studied guidelines for designing e-study materials and higher education e-learning frameworks.

In cooperation between the University of Latvia and the University of Daugavpils, under the guidance of Laila Niedrite, V. Vagale, S. Ignatjeva et al. have focused on development of a personalized e-learning. Independently, S. Cakula and M. Sedleniece carried out similar research at the University College of Vidzeme.

L. Niedrite, D. Solodovnikova, N. Kozmina and J. Borzovs conducted a series of studies in an attempt to find the human properties for successful information technology studies.

University of Latvia in cooperation with Vilnius University and other Latvian universities and institutes conducts the Baltic Journal of Modern Computing (established in 2008) where the Computing Didactics, led by V. Dagienė from Vilnius University and M. Vītiņš from University of Latvia, is one of the nine journal's content areas. The topics in the area of Computing Didactics are quite broad. The research is more related to use of digital technologies in education than to computer science education, but few papers are devoted to CER, for example [30, 32].

## 4 CER in Lithuania

Teaching informatics (computing/computer science) in secondary schools in Lithuania was implemented in 1986. On occasion of the 20th anniversary of informatics in schools, the second international conference ISSEP (Informatics in Schools: Situation, Evolution, and Perspectives) was arranged in Vilnius, and the book "The Road of Informatics" [33] was published in both Lithuanian and English, which provides an overview of the teaching of computing over the past 20 years. Before the official introduction of informatics in schools, all school students had the opportunity to learn programming elements remotely 5 years earlier, starting in 1981. This was the School of Young Programmers by correspondence. At the same

time, researchers in the Institute of Mathematics and Informatics (now Institute of Data Science and Digital Technologies) began discussions on how to teach programming to secondary school students.

## *4.1 Prehistory: School of Young Programmers by Correspondence*

In 1979, an experimental School of Young Programmers by correspondence was conducted. Two years later, the nationwide School of Young Programmers named JPM (acronym for Lithuanian *Jaunųjų programuotojų mokykla*) was established.

Ministry of Education of Lithuania approved the teaching materials and tasks for publication in the most popular youth daily newspaper at that time (*Komjaunimo tiesa*). Ordinary postal service was used for communication between teachers and school students. In the early 1980s, this school was the only educational institution enabling to get a primary acquaintance with algorithms and computer programming for most students in Lithuania, especially for those living in provinces. The JPM provided an excellent opportunity for CER. Many papers, especially those focused on methodology of teaching programming for young people, were published in local publications [34–36].

### 4.1.1 Main Curriculum of the School of Young Programmers

In order to convey the foundations of contemporary programming methodology to the students of the JPM, theoretical knowledge is necessary as well. However, for most children, the theory is less attractive than practical activities. Thus, the basic principles of the theory were delivered in an indirect way through problem solving. The set of programming problems was chosen in accordance with the requirements dictated by appropriate programming style and creativity [37].

All the teaching materials of JPM till 1993 consisted of several teaching chapters, for example: (1) Names, variables, values, assignment statement and sequence of statements; (2) Branches of actions; (3) Repetitions of actions; (4) Program and its running by computer; (5) Logical values; (6) Functions and procedures; (7) Recursion; (8) Discrete data types; (9) Real numbers and records; (10) Arrays; (11) Programming methodology (i.e. style); (12) Program design.

Many tasks were small and simple (e.g., to find the perfect numbers, friendly numbers, etc.), though their solutions without a computer is cumbersome. The benefit of a computer for solving such tasks may be demonstrated too. A special attention was given to the Boolean type and recursion. Logic is a base of the programming as a whole and the recursion may be considered as a bridge between two programming paradigms, the procedural and nonprocedural programming.

SEPTINTOJI

PAMOKA

Skyrelį tvarko LTSR MA Matematikos ir kibernetikos instituto jaunesnioji moksliné bendradarbė V. DAGIENĖ

CIKLAI CIKLE

menų sumos. 2) Imti visas galimas skaitmenų poras. Iš jų sudaryti dviženklį skaičių ir tikrinti, ar šis skaičius dalosi iš skaitmenų sumos. Sprąsime antruoju būdu.

```
kiek:=0;
for a:=1 to 9 do
  for b:=0 to 9 do
    begin
      sk:=a*10+b;
      s:=a+b;
      if sk mod s=0 then
        kiek:=kiek+1
    end
```

Čia dviženklis skaičius pažymėtas vardu sk, jo skaitmenys — vardais a ir b, skaitmenų suma — s, o rezultatas — vardu kiek. Išoriniu ciklu perrenkami visi skaitmenys nuo 1 iki 9 — tai pirmasis dviženklio skaičiaus skaitmuo. Vidiniu ciklu perrenkami visi skaitmenys nuo 0 iki 9 (antrasis dviženklio skaičiaus skaitmuo gali būti ir nulis). Taigi išorinis ciklas atliekamas 9 kartus, o vidinis 9×10=90 kartų.

**Fig. 4** The fragment of the newspaper with lesson in programming. In the upper left corner appears the logo of JPM – letters J (for "Young"), P (for "Programming"), and M (for "School") combined together

### 4.1.2 The Growth of the School of Young Programmers

Since 1981 until 2014, there have been many changes in the teaching of informatics in general and in programming in particular due to the increase in the number of computers in educational institutions and introduction of informatics as compulsory discipline in secondary schools. These changes had a considerable effect on the teaching in the JPM [38–40]. The changes can be characterized by five periods:

1. Universal (general) programming teaching (1981–1986).
2. Learning effectively: differentiation by students' abilities (1986–1993).
3. Intensive teaching of gifted students (1993–1999).
4. Preparing students for the Olympiads (1999–2005).
5. Using new media while learning algorithms (2005–2014).

The primary course of programming used to be published in the daily newspaper (Fig. 4), one lesson per week (1981 January – May, 1983 September – December and 1985 September – December). The lessons covered all the material in programming needed for beginners: text and tasks for self-control as well as certification tests. The JPM lessons in newspaper were the only possible way to get primary acquaintance with computers and programming for many youngsters at that time.

The primary course of programming covered only four chapters of the JPM curriculum: all operations with integer numbers, assignment statements, a sequence of statements, conditional and loop statements. Program execution by computer was discussed as well. Students were taught to solve interesting and attractive problems by algorithmic approach [41]. Tests were presented about once per month, so that the students studying the primary course could carry out four or five tests per year.

The second course of the JPM was more difficult and lasted a bit longer – from a year and a half to two years. There were considerably less students studying this part than the first one. During 1993–1994 the JPM was reorganized because of increasing numbers of computers in schools and growing participation in national Olympiads in informatics [41]. The studies were divided into two parts (courses): (1) constructs of algorithms and programming, and (2) methods of algorithms.

Reading (analyzing) tasks made up to a quarter of each student's homework. All the tasks were attractive and developed reasonable thinking. The first part of the course covered the whole curriculum of JPM and consisted of five training chapters and a test: (1). An algorithm. Variables. Assignment statement. (2). Control structures: conditional, compound and loop statements. (3). Functions and procedures. Recursion. (4). Scalar (simple) data types. (5). Data structures.

The second part of the course consisted of five tasks classified according to the nature of the algorithms' methods. There were usually five topics (they could be slightly different each year), i. e.: (1) the big numbers, (2) units of measurement (regular and irregular), number systems, calendars, (3) searching for solutions, backtracking method, puzzles, (4) coding and ciphering, finding and correcting mistakes in data, (5) data sorting, (6) dynamic programming, (7) graph algorithms.

During the period 1999–2001, the JPM was reorganized again. The main reason for the reorganization was the changes in curriculum of informatics in schools and the spread of new communication technologies, i. e., an electronic mail system [38, 42]. The goal of the changes at that period was to differentiate the learning course by adding one year. Fundamental principles of programming and developing of algorithms were the basis for teaching. The second year was devoted to the learning of various algorithms, e.g., data sorting and searching, recursion, backtracking, and graph algorithms. The third year was intended to analyze more advanced tasks of informatics (similar to tasks of Olympiads in informatics).

In 2005–2006, a virtual learning environment, developed by Lithuanian Olympiad student A. Paltanavičius, started to be used at JPM [43] and all students get tasks and deliver solutions using the virtual environment. The JPM curriculum has not been changed.

### 4.1.3 Training Approaches and Research

A specialized Pascal translator (compiler) for schools for the soviet computer EC ЭВМ (a clone of IBM System/360) was developed by young researchers V. Dagienė and A. Petrauskienė led by Dr. Gintautas Grigas. The translator had an error-detection module that allowed students to receive useful feedback, and even corrected part of errors in beginner's programs, and reported on what and how it changed. This was a very important feature since access to the computer was limited. Several studies were conducted and research papers published [44].

Studying in JPM was voluntary and individual. Training and evaluation approaches had been changing during the time. At the beginning (1981–1986) the difficulty of each task and different maximum points for different tasks given

were estimated. However, it appeared that often JPM participants could not solve rather easy tasks and sometimes surprisingly solved tasks which were estimated to be hard and high points were given for them.

During the whole period of JPM's participating students were receiving advice and guidance although their work was evaluated strictly enough. This circumstance as well as proving of the characteristics required for a programmer (e.g., algorithmical thinking, creativity, diligence, accuracy, and attention) and the difference of the extramural studies from the usual ones at school determine that just a part of students who have tried themselves at JPM are awarded with its certificate.

During 34 years of JPM's existence, over 7000 school students were introduced to programming basics. The JPM was one of the long-existing school on programming by correspondence available for all school students all over world. Main issues can be highlighted following the long experiences in teaching programming:

- Attention to the programming and algorithmic style as a part of information culture. Meaningful names of variables, procedures and other objects are selected, and commented.
- Introducing the reading's tasks. Analyzing algorithms is proved important while learning programming. By reading a task, more complicated algorithm can be introduced.
- Priority is given to tasks which requires creativity in programming. Teaching programming rather than programming language.

The JPM triggered a number of scientific and methodological articles, books (however, all of them are in Lithuanian and only printed versions), contests, and other activities. The team was enthusiastically inspired by the idea to contribute to computing education at schools despite lacking even the most elementary resources, e.g., there was only one typewriter in the entire institute.

## *4.2 Compulsory Informatics in Schools: From 1986 Until Now*

A significant role in the history of designing informatics education in secondary schools was played by the scientists of the Institute of Mathematics and Informatics. Comprehensive materials were prepared including tasks, tests and didactical explanations for teaching computing with main focus programming. Just like in Estonia and Latvia, informatics subject was introduced in Lithuanian schools in 1986, under the slogan by Soviet academician A. Ershov: "*Programming is the second literacy*" [45]. This quote is used even today, often without any reference to the original author. These were not empty words: in order to reach this objective, the Soviet Union invested significant resources and efforts.

### 4.2.1 Teaching Algorithms: From Logo to Pascal

In 2006, when Lithuania celebrated 20 years of informatics in schools, the book "The Road of Informatics" was published and it was written: "*At the beginning there was Logo. Then everything happened*" [33]. Teaching informatics started with Logo developed by Seymour Papert, the "father of Logo".

The book "Mindstorms: Children, Machines, and Powerful Ideas" by S. Papert [46] was translated into the Lithuanian language. Educators throughout the world became excited by the intellectual and creative potential of Logo. Their enthusiasm fueled the Logo boom of the early 1980s. The book begins with an affirmation of the importance of making a personal connection with one's own learning and ends with an examination of the social context in which learning occurs.

In 1985, the Logo Computer Systems corporation developed an interpreter "LogoWriter", which was localized into Lithuanian and used in schools for many years. "Mindstorms" (1980) [46] and "Logo Writer" were tools that helped teachers to grow up combining computing and innovative pedagogy such like constructionism. Teachers were influenced by S. Papert's ideas that learning to communicate with a computer (to program the computer) may change the way other learning takes place.

Later a "Comenius Logo" for Windows was purchased for all schools in Lithuania and localized. It had fascinated graphics, an easy animation capability, but teachers and students needed to be creative and willing to do something new. That's about innovative pedagogy [47]. Many exercises for learning Logo were developed, and many books published on Logo-inspired ideas to introduce children to programming [48]. The ideas from Logo reoccur time after time; through development of learning tools in many countries and in several companies when new educational products are developed. For many years, Vilnius University has been cooperating with the CER group of ETH Zurich, Switzerland led by Juraj Hromkovič.

There has been many important events in connection with teaching Logo and algorithms all over the world, a detailed Logo Tree Project is provided by P. Boytchev [49].

Pascal, the great programming language that served in education for many years, was designed in 1968 by Nicklaus Wirth, with the goal to encourage good programming practices to novices by using structuring programming and data structuring. Pascal had big influence in programming education in many countries. In Lithuania, Pascal was chosen as language to communicate to big machines and express algorithms especially for secondary school students. It was a way to algorithmical thinking. Pascal was backbone for Young Programmers School [43].

Informatics education in Lithuanian schools (grades 10–12) was based on Pascal, the language that fit perfectly to think of and develop algorithms. Pascal's advantage was its simple syntax and logical structures. Pascal was used also to develop algorithms on paper and pencil and run them later on a computer, which was an important aspect in the early eighties when schools had very limited access to machines.

### 4.2.2 Informatics Curriculum Developments

In 1991, the second year of Lithuanian independence, a first original textbook for grades 10 and 11 was published. Attention was drawn to main concepts of informatics rather than to the particular technical details of computing techniques and programming. The authors V. Dagienė and G. Grigas [50] emphasized the design of algorithms and of algorithmics style. An innovative approach was the reading of well-prepared algorithms (reading tasks) and answering challenging questions. Thousands of such "reading tasks" for learning algorithms were published in a series of books.

The course in informatics started to be taught in Lithuanian comprehensive schools 35 years ago. The contents of the course, evaluation, and even the name were changed several times. Nevertheless, informatics has remained as a separate subject in schools of Lithuania. In 2002, informatics subject was renamed as information technologies (IT) in grades 11th and 12th while it was still called informatics in grades 9 and 10 [38]. The objective was to teach both, information technologies skills and computer science concepts including programming.

A revision of the informatics core curriculum was initiated in 2005, expanding the scope from 2 to 4 years teaching time (in total 136 h) with more focus on developing algorithmic thinking and applications [51]. The teachers were formally qualified, usually with a bachelor or master degree in informatics combined with mathematics. Fifth and sixth grade pupils are introduced to the basics of informatics based on Logo or Scratch. In grades 9 to 10, more advanced students are recommended to enroll in the optional module of algorithm design and coding.

During years, the main aim of teaching informatics in general education is to develop students' information culture (digital literacy) in a broad sense. The information culture is a wide concept, considerably wider than information skills or abilities to work by computer. The conception of information culture covers various abilities and skills [52]. The content of information culture's notion is constantly changing and is reliant on technological transformations, it embraces a broad range of students' cognitive and other abilities and attitudes.

Concerning informatics curriculum development and implementation in schools there were many studies also comparison with other countries done for example [53–55] and recommendations taken into account for further developments [38, 51, 56–58].

In 2019, the Lithuanian Ministry of Education, Science and Sport approved new guidelines for pre-school, primary, basic and secondary education [59]. Informatics is included in primary school level as well (Fig. 5). In 2020, one hundred primary schools started to pilot the new developed informatics curriculum (www.mokykla2030.lt). The pilot targeted to develop learning resources and textbooks, as well as teacher training. The full-scale implementation of the new informatics curriculum for all grades commences in 2023.

The new informatics curriculum includes fundamental Informatics topics such as programming, problem solving and algorithms, data mining, data representation and information, networks and communication, digital technology and human computer

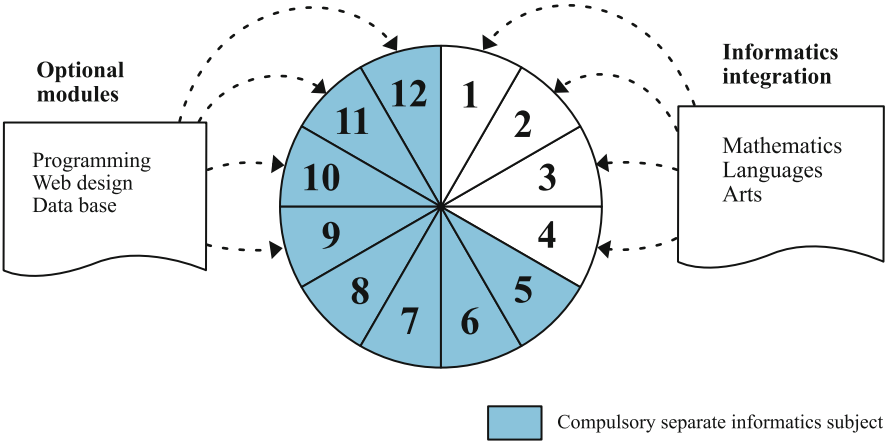


**Fig. 5** Informatics course in schools in Lithuania

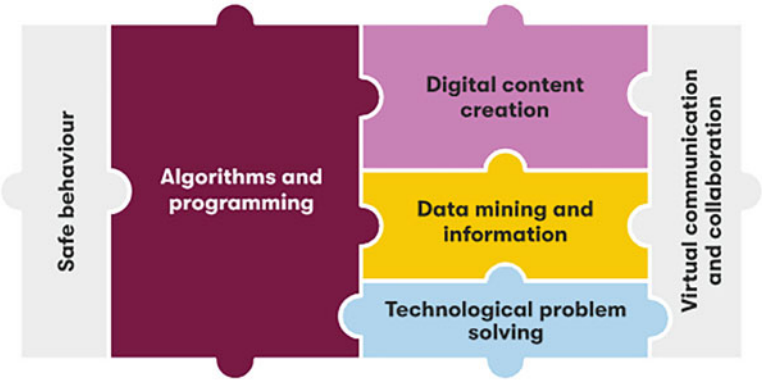


**Fig. 6** Informatics course in schools in Lithuania

interaction, security, and privacy, and ethical considerations. Six areas of informatics education with a main focus on the first four core areas were identified and used in the new curriculum in Lithuania (Fig. 6).

The six areas are applied to all levels of school education in Lithuania, starting from grade 1 of primary school (and even including pre-school education) all the way up to grade 12 of upper secondary school. In Table 3, the achievements of students and essential skills are briefly presented.

The challenge is to redesign an informatics course of upper secondary school (grades 11–12) so that it would be modern and cover new technologies such as artificial intelligence, machine learning, big data, and deep learning.

**Table 3** Six areas of a new established informatics curriculum in primary and secondary schools

| Area | Essential skills |
|---|---|
| Digital content creation | Familiarize with digital content diversity and forms, recognize computational thinking concepts<br>Use digital content to learn in various subjects<br>Create digital content using various technologies: draw, write, compose, record, film, make video, create mind maps, tables, graphs<br>Evaluate, improve and share digital content |
| Algorithms and programming | Understand the importance of algorithms and programs, provide samples from everyday life<br>Apply commands, logical operations and programming interfaces<br>Identify sequencing, branching, loop actions and express them by commands, apply logical operations<br>Create and run programs using programming tools and virtual environments<br>Test, debug and improve programs |
| Technological problem solving | Identify problems occurring when using digital technologies and devices<br>Creatively use digital technologies to learn various subjects<br>Select appropriate digital technologies to solve tasks<br>Self-educate and self-evaluate own digital skills |
| Data mining and information | Understand the importance of data and information, recognize computational thinking concepts<br>Search information purposefully using digital technologies<br>Perform various actions on data: collect, store, group, search, visualize<br>Evaluate relevance and reliability of information |
| Virtual communication and collaboration | Understand purpose and importance of virtual communication<br>Collaborate, share experiences and resources, communicate using digital technologies<br>Discuss possibilities and risks of virtual communication |
| Safe behavior | Perceive the necessity to protect digital devices from malicious software and spam<br>Discuss copyright and piracy issues<br>Protect health and environment while using digital technologies<br>Behave safety in virtual space |

### 4.2.3 National School Leaving Exam in Informatics

To meet the needs of higher education institutions and the economic development of the country, the nationwide maturity exam to evaluate students' skills in programming was established in 1995 [60]. Those who pass the informatics exam have enhanced opportunities to enter computing-related studies in higher education.

The main function of an exam is the evaluation of learning results [61, 62]. The content of the exam is closely related to the subject's curriculum. The proper selection of the exam's goals, and the emphasis (or lack thereof) on one or another aspect of the subject, have a strong impact on the quality and content of learning as well as on the students' motivation to learn the discipline. Lithuanian teachers and students pay attention to exams and therefore this situation should be exploited. By creating the content of the exam, a double goal could be achieved: to evaluate the students' knowledge and to encourage a student to cultivate his or her skills in the chosen discipline.

Students are acquainted with the formulation of topics in advance. The exam questions were formulated in a more concrete or constricted way, nevertheless their character was broad enough, requiring reasoning and a fair knowledge of the field. The topics embrace the compulsory course and the main conceptions associated with common human information activity. By answering the questions, students not only show their knowledge in a certain field but also have a chance to demonstrate their ability to express thoughts in a clear, logical and correct manner.

During the exam, students have to write programs of two tasks. The tasks consisted of several parts. The goals are to evaluate students' practical skills, the ability to choose and to create data types for the tasks, as well as to apply algorithms and tools for program structure in particular situations [63]. The content of the exam in informatics is based on the curriculum of grades 11–12. Since the course is quite modest, only 68 h, the exam is fairly compact.

## *4.3 Two International Journals on Informatics Education*

In 2002, Vilnius university established an international journal "Informatics in Education" (https://infedu.vu.lt/journal/INFEDU), a peer-reviewed journal that provides an international forum for presenting the latest original research results and developments in the fields of CER. At the beginning, the journal was promoting research among educators both in the Baltic countries and in Eastern and Central Europe, but now there are authors from all over the world. The topics range across diverse aspects of computing education research including empirical studies, statistical research on big data related to computing education, educational engineering focusing mainly on developing high quality original teaching sequences of different computer science topics that offer new, successful ways for knowledge transfer and development of CT, design of educational tools that apply technology in novel ways.

**Table 4** Submissions of papers in 2016–2021

| Number of papers/Years | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 |
|---|---|---|---|---|---|---|
| Total submissions | 91 | 112 | 151 | 154 | 172 | 195 |
| Accepted and published | 16 | 15 | 20 | 20 | 30 | 30 |
| Rejected after peer review | 42 | 38 | 32 | 40 | 35 | 11 |
| Rejected before peer review | 35 | 61 | 90 | 86 | 101 | 154 |
| Rejection rate | 85% | 88% | 81% | 82% | 79% | 85% |

Recently "Informatics in Education" is published by Vilnius University Institute of Data Science and Digital Technologies in cooperation with Swiss Federal Institute of Technology (ETH), Zurich, Centre for Computer Science Education. The journal's visibility is growing and well-written paper submissions are increased. In 2021, the journal received a total of 195 papers. The acceptance rate is quite low, less than 20% (Table 4).

The journal is indexed in many data bases including Web of Science Emerging List (176th place among 739 journals in Education & Educational Research category, 2022). The journal citation rate in Scopus is quite high (4.1 in CiteScore 2021). From 2021, the journal is indexed in the ICI Journals Master List database with Index Copernicus Value (ICV) = 128.60 (more information at https://journals.indexcopernicus.com/).

Another journal established by Vilnius university in 2007, is an international scholarly journal on Olympiads in informatics "Olympiads in Informatics", it is described in chapter 'Computing Education Research in Schools' in section on Olympiads.

## *4.4 Hosting International Conferences of CER*

Several international CER conferences were arranged by Lithuanian researchers. The most important are: ISSEP, ITiCSE, Eurologo, named later by Constructionism, and specialized IFIP conferences. In 2006, on the occasion of the twentieth anniversary of teaching informatics in schools in Lithuania, ISSEP conference was organized. Researchers from 37 countries were participated. More than one hundred papers were selected and published: the best 29 papers were published in "Lecture Notes in Computer Science" by Springer, and 70 papers were published in the conference proceedings.

In 2018 in Vilnius, the Constructionism conference celebrated its fifth anniversary under this name, building on the 27-year tradition of biennial Eurologo conferences established by the European Logo community. S. Papert coined the term constructionism: in order to define its meaning, he started from the comparison with the term constructivism: "Constructionism shares constructivism's connotation of learning as 'building knowledge structures' irrespective of the circumstances of the

learning. He then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it's a sand castle on the beach or a theory of the universe" [65].

The Constructionism conference was truly international with about 150 submissions from 40 countries. The accepted submissions consisted of 18 keynotes talks, 57 research and practice papers, 3 panels, 7 working group proposals, and 27 proposals for posters, demonstrations, and workshops. In addition, a special Teachers' Day is organized before the conference. Selected research papers were published in the international peer-reviewed journals "Informatics in Education" and "Problemos". The best papers were selected and published in the "Constructivist Foundations" journal.

### *4.5 Doctoral Consortium*

Doctoral consortium is a yearly three-day event, shaped as an international version of the seminar, in which doctoral students meet and collaborate with peers, supervisors and scientific experts from other countries. Institute of Data Science and Digital Technologies of Vilnius University established international Doctoral Consortium on Informatics Education and Informatics Engineering Education Research in 2010 and organizes it each year on the first or second week of December in Druskininkai, Lithuania.

The Doctoral Consortium provides an opportunity for doctoral students to explore and develop their research interests in a workshop under the guidance of distinguished senior researchers.

The Consortium is designed as a student-cantered event and offers:

- A friendly forum for doctoral students to discuss their research topics, research questions and design in the field of their research;
- A supportive setting for feedback on students' current research and guidance on future research directions;
- Comments and fresh perspectives for each student on his/her work from researchers and students outside their own institution, as well as help with choosing suitable methodology and strategies for research;
- Support networking with other researchers in the informatics engineering education research field, and promote the development of a supportive community of scholars and a spirit of collaborative research;
- Support for a new generation of researchers with information and advice on research and academic career paths.

This event is attended annually by 12–18 doctoral students, and at least 6 senior researchers representing several countries who give lectures and work with small doctoral student groups. The methodology used in the Doctoral Consortium is project-based, going through methodological stages, students develop posters of

their research project, share, and discuss their project with the Consortium's community.

The consortium was designed primarily for students who are currently enrolled in any stage of doctoral studies with a focus on CER and education research or with a focus on other areas of research in connection to globalization, modern technologies, and education. Senior researchers in the field provided feedback and suggestions for improvement of the research proposals. The Doctoral Consortium is a friendly forum for doctoral students to discuss their research topics, research questions and design in the field of education, to receive constructive feedback from their peers and senior researchers, to help with choosing suitable methodology and strategies for research.

In December 2021, the Doctoral Consortium was organized by two divisions of Vilnius University: Institute of Data Science and Digital Technologies and Institute of Educational Sciences. 29 doctoral students participated from 10 countries (Germany, Austria, Sweden, Estonia, Switzerland, Hungary, Netherlands, Finland, Slovenia, Lithuania) and 22 senior researchers from 12 countries (United Kingdom, Netherlands, Portugal, Italy, Slovenia, Ukraine, Austria, Hungary, Estonia, Macedonia, Switzerland, Lithuania). Senior researchers stressed that, despite deepening scientific knowledge, this consortium provided opportunities for intercultural exchange, both within the international student group and meeting local students and teacher.

## *4.6 Research on School Informatics*

Computing education research in Lithuania has started almost a half century ago. A significant role in designing methods for teaching programming was played by the scientists at the Institute of Mathematics and Informatics; more about history can be found on the website: https://www.mii.lt/en/about-the-institute/history. So, in 1978–1979, the institute scientists designed methods and resources for teaching programming at schools and conducted various small-scale studies.

In 1984, a Department of Programming Methodology was established led by G. Grigas, later in 2002, it was renamed to Department of Informatics Methodology and V. Dagienė became a chair. This small group of active scientists were among the first to create a programming learning environment where the teaching of informatics took shape. These scientists established the School of Young Programmers (see Sect. 4.1), designed and developed informatics curricula, teaching methodology, textbooks, and maturity exam (see Sect. 4.2), established two international research journals (see Sect. 4.3), hosted national and international conferences, seminars, workshops, the doctoral consortium (see Sects. 4.4 and 4.5) as well as conducted doctoral studies.

Since 2010, the Institute of Data Science and Digital Technologies has become part of Vilnius University, but informatics research continues within Education Systems Group led by V. Dagienė. The group conducts research on teaching

and learning informatics at schools, approaches and methodology, informatics curriculum development for primary and secondary schools, teacher training with focus on informatics, also computing engineering education research especially in connection to educational software localization, and development of terminology.

More or less, the research on CER is taken care of by several other universities:

- Klaipėda University where several studies on informatics teacher education and educational software development were conducted led by Vitalijus Denisovas;
- Šiauliai University (became part of Vilnius University since 2021) participated in several studies on informatics teacher education led by Sigita Turskienė;
- Kaunas University of Technology has formed strong leadership in research on distance education mainly focusing on higher education (including informatics) led by Aleksandras Targamadzė.

Today the main research group in informatics education (CER) is the Education System group in Institute of Data Science and Digital Technologies of Vilnius University. One of the most important results of this group is the doctoral studies in the area of CER. More than 20 doctoral theses in connection to CER were supervised and defended, for example, "Concept-Driven Informatics Education: Extension of Computational Thinking Tasks and Educational Platform for Primary School" by G. Stupurienė in 2019, "Software learning objects for scientific computing education: teaching parallelization with recurrence based stochastic models" by V. Dolgopolovas in 2018, "Design of adaptive programming teaching tools" by J. Urbonienė in 2014, "A method for semi-automatic evaluation and testing programming assignments" by B. Skūpas in 2013.

## 5 Discussion and Conclusions

Computing education in schools of Estonia, Latvia, and Lithuania is entering the fifth decade of its existence. All three countries put a lot of effort in developing curricula, textbooks, and other activities in computing education and research. As the ongoing innovation is always in dialogue with today's challenges as well as yesterday's decisions, we explored in this chapter the historical perspectives on the development of computing education and related research in these three Baltic countries.

It is not easy to design curricular change that aims to shift the focus from acquisition of static content knowledge to integration of deeper procedural and conceptual knowledge, while considering local culture, language, etc. The impact of digital technologies on global and local contexts is forcing us to constantly redefine the forms of literacy and skills that are needed to survive in the world of tomorrow. These cognitive skills are based on higher-order thinking that might (or should?) involve computing. Hence, it is not surprising that computational thinking has been increasingly prioritized by policy makers around the world [66]. This is both a challenge as well as an opportunity for Computing Education Research today.

However, some lessons learned from the history of computing education in Baltic countries are valuable also today. Here are some conclusions we can draw from our experience:

- students benefit from developing a broad understanding of computing, as it prepares them better for adopting digital innovations in the future;
- instead of drilling the mechanic operations, the focus should be on problem solving and computational thinking including algorithmic thinking;
- to build deeper understanding, computing should be taught independently of application software, programming languages and environments;
- it helps if computing is taught using real-world problems while not avoiding computer science concepts and approaches;
- computing education should prepare students for the professional use of computing in other disciplines and fields;
- teaching and learning computing must encourage students to be creative.

Although including computing into formal education through curriculum as a separate school subject is very important, it is also important to support the informal education on computational thinking, especially for talented kids. Informatics curricula in all three countries are continuously being developed and updated according to recommendation of important interest groups and recommendations of international organizations such as ACM, CECE, CSTA, and UNESCO. Finally, we are glad that in the Baltic countries there is a strong involvement of computing education researchers in curriculum development and implementation, textbook publishing, teacher education and education policy development.

**Acknowledgments** The authors would like to thank Sonsoles López-Pernas, University of Eastern Finland, for providing scientometric analysis.

## References

1. Dagienė, V. & Sentance, S. (2016, October). It's computational thinking! Bebras tasks in the curriculum. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 28–39). Springer, Cham.
2. López-Pernas, Saqr, M., & Apiola, M. (2023). Scientometrics: A concise introduction and a detailed methodology for the mapping of the scientific field of computing education. In: *Past, Present and Future of Computing Education Research*. Springer.
3. Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., ... & Toll, D. (2015). Educational data mining and learning analytics in programming: Literature review and case studies. *Proceedings of the 2015 ITiCSE on Working Group Reports*, 41–63.
4. Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014, June). Computational thinking in K-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference* (pp. 1–29).
5. Dagienė, V., & Futschek, G. (2008, July). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In *International conference on informatics in secondary schools-evolution and perspectives* (pp. 19–30). Springer, Berlin, Heidelberg.

6. Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., Pal, Y., Jackova, J., & Jasute, E. (2015). A global snapshot of computer science education in K-12 schools. In *Proceedings of the 2015 ITiCSE on working group reports* (pp. 65–83).
7. Dagienė, V., & Stupurienė, G. (2016). Bebras-a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in education*, 15(1), 25–44.
8. Afinogenov, G. (2013). Andrei Ershov and the Soviet information age. *Kritika: Explorations in Russian and Eurasian History*, 14(3), 561–584.
9. Kanger, L. (2013). *Domestic PC production in the Soviet Baltic States 1977-1992* (Doctoral dissertation, University of Edinburgh).
10. Poranen, T., Dagiene, V., Eldhuset, Å, Hyyröo, H., Kubica, M., Laaksonen, A., Opmanis, M., Pohl, W., Skupiene, J., Söoderhjelm, P., & Truu, A. (2009). Baltic olympiads in informatics: challenges for training together. *Olympiads Informatics*, 3, 112–131.
11. Tõugu, E. (2018). Arvutid, küberruum ja tehismõistus. Tallinn.
12. Agur, U., Toim, K., & Unt, I. (1967). *Programmõpe ja õpimasinad*. Tallinn: Valgus.
13. Villems, A. (2009). Rollimängud ja simulatsioonid. In *Tiigriõpe: Haridustehnoloogia käsiraamat* (pp. 167−180). Tallinn.
14. Laanpere, M. (2001). Koolinformaatika ja teine kirjaoskus. A&A, 14(1).
15. Villems, A., & Tooding, L.-M. (2006). Study on ICT Competency of Estonian Pupils. In *Information Technologies at School* (pp. 436−446). Vilnius: TEV.
16. Amitan, I., Vilipõld, J., & Võhandu, L. (1999). Programmeerimise õpetamisest VBA baasil Exceli keskkonnas. *A&A*, 4, 30−34.
17. Leppik, C., Haaristo, H.-S., Mägi, E., Kõiv, K. (2017). IKT haridus Eesti üldhariduskoolides ja lasteadades. Tallinn: Praxis.
18. Salum, K., Luik, P., Lepp, M., & Laanpere, M. (2021). Teaching informatics in upper secondary school - preparing students for the future. ICERI2021 Proceedings (pp. 3544−3553).
19. Niemelä, P., Pears, A., Dagienė, V., & Laanpere, M. (2021, August). Computational Thinking–Forces Shaping Curriculum and Policy in Finland, Sweden and the Baltic Countries. In *Open Conference on Computers in Education* (pp. 131–143). Springer, Cham.
20. Education Estonia (2021). ProgeTiger – Estonian way to create interest in technology. https://www.educationestonia.org/progetiger
21. Luik, P., Suviste, R., Lepp, M., Palts, T., Tõnisson, E., Säde, M., Papli, K. (2019). What motivates enrolment in programming MOOCs? *British Journal of Educational Technology*, 50(1), 153−165.
22. Palts, T. (2021). A Model for Assessing Computational Thinking Skills. PhD thesis. Tartu.
23. Muuli, E., Papli, K., Tõnisson, E., Lepp, M., Palts, T., Suviste, R., Säde, M., & Luik, P. (2017, September). Automatic assessment of programming assignments using image recognition. In *European Conference on Technology Enhanced Learning* (pp. 153–163). Springer, Cham.
24. Sillat, L. H., Tammets, K., & Laanpere, M. (2021). Digital competence assessment methods in higher education: A systematic literature review. *Education Sciences*, 11(8), 402.
25. Pata, K., Tammets, K., Väljataga, T., Kori, K., Laanpere, M., & Rõbtsenkov, R. (2021). The patterns of school improvement in digitally innovative schools. *Technology Knowledge and Learning*, 1–19.
26. Eradze, M., Rodríguez-Triana, M.J., Milikic, N., Laanpere,M., Tammets, K. (2020). Contextualising learning analytics with classroom observations: a case study.
27. Lorenz, B., Kikkas, K., & Sõmer, T. (2021, October). IT as a Career Choice for Girls: Breaking the (Self-Imposed) Glass Ceiling. In *ECEL 2021 20th European Conference on e-Learning* (pp. 266–274).
28. Ershov, A. P. et al. (1986). Ершов А., Монахов В., Витиньш М. и др. Изучение основ информатики и вычислительной техники. Часть II. Москва: Просвещение,
29. Borzovs, J., Otas, A., & Telesius, E. (2001). ECDL in Latvia and Lithuania. In *Baltic IT&T 2001 Forum: eBaltics.–Ryga, The Information Technology Committee of the Baltic Council of Ministers* (pp. 217–220).

30. Vezis, V., & Krumins, O. (2019). New Competencies-Based Curriculum of Computing in General Basic Education Schools in Latvia and its Testing. *Baltic Journal of Modern Computing*, 7(3), 364–379.
31. Vezis V., & Krumins O. (2018). Fifty-Five Years of the Teaching of Informatics at Latvian Schools. *Baltic Journal Modern Computing*, 6(2), 107–118.
32. Kalelioğlu, F., Gülbahar, Y., Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing*, 4(3), 583–596.
33. Dagienė, V. (2006). *The Road of Informatics*, Vilnius: TEV.
34. Baliūnaitė, A., Dagienė, V, & Grigas, G. (1980). Osobenosti realizaciji transliatora jazyka PASCAL dlia učebnych celej. *Lietuvos matematikos rinkinys*, XX(3), 189–190.
35. Dagienė, V., & Grigas, G. (1981). Metodika raščiota vremeni vypolnenija operatorov jazyka programirovanija. *ESM programavimas*, 3. Vilnius: LTSR MA MKI, 105–124.
36. Dagienė, V., & Grigas, G. (1986). Rekursivnyje tipy danych i algebričeskije specifikaciji. *ESM programavimas*, 10, Vilnius: LTSR MA MKI, 9–17.
37. Grigas, G. (1990). Some aspects of teaching the art of programming by correspondence. *Informatica*. 1(1), 156–166.
38. Dagiene, V. (1999, June). Programming-Based Solutions of Problems In Informatics Curricula. In *IFIP WG 3.1 and 3.5 Open Conference "Communications and Networking in Education: Learning in a Networked Society* (pp. 88–94).
39. Dagiene, V. (2002). The Model of Teaching Informatics in Lithuanian Comprehensive Schools. *Journal of Research on Computing in Education*, 35(2), 176–185
40. Dagiene, V. (2008, November). Distance Reflective Learning in Lithuanian Young Programmers School. In *Proceedings of the 7th European conference on e-learning: ECEL2008* (p. 264–271), Agia Napa, Cyprus.
41. Dagys, V. (1994). The work principles of Lithuanian Young Programmers School by correspondence. In *Human Resources, Human Potential, Human Development: the Role of Distance Education. In Proceedings of the European Distance Education Network (EDEN) Conference. Tallinn* (pp. 182–184).
42. Dagys, V., & Klupšaitė, A. (1993). Distance teaching of programming and possibilities of e-mail. *Informatica*, 4(3–4), 303–311.
43. Dagys, V., Dagiene, V., & Grigas, G. (2006, November). Teaching algorithms and programming by distance: quarter century's activity in Lithuania. In: P*roceedings of Conference on Informatics in Secondary Schools: Evolution and Perspectives* (pp. 402–412).
44. Dagienė V., Grigas G. & Petrauskienė A. (1983). Transliator jazyka Pascal, prednaznačenyj dlia učebnych celej. *Programirovanije*, 2, 87–89.
45. Ershov, A. P. (1981). Programming, the second literacy. *Microprocessing and Microprogramming*, 8(1), 1–9.
46. Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books.
47. Dagienė, V. (2001, August). Algorithmic thinking: what set of Logo constructions should be used to develop it? *Proceedings of the 8th European Logo Conference* (pp. 245–252), Linz, Austria.
48. Dagienė, V. (2003, August). A set of Logo problems for teaching algorithms. *Proceedings of the 9th European Logo Conference* (pp. 168–177), Porto, Portugal.
49. Boytchev, P. (2007). *Logo tree project*. http://www.burcsi.hu/P-JsLogo/doku/LogoTreeProject.pdf
50. Dagienė V., & Grigas G. (1991). *Informatika*. XI-XII klasei. Kaunas: Šviesa.
51. Dagienė, V. (2005, March). Teaching information technology in general education: Challenges and perspectives. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (pp. 53–64). Springer, Berlin, Heidelberg.
52. General Curriculum and Education Standards (2003). Pre-school, Primary, and Basic Education. Vilnius: Ministry of Education and Science.
53. Benaya, T., Dagiene, V., & Gal-Ezer, J. (2015, July). CS High School Curriculum–A Tale of Two Countries. In *Proceedings Intlernational Conference IFIP* (pp. 17–28).

54. Bell, T., Curzon, P., Cutts, Q., Dagiene, V., & Haberman, B. (2011, June). Introducing students to computer science with programmes that don't emphasise programming. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (pp. 391–391).
55. Schulte, C., Hornung, M., Sentance, S., Dagiene, V., Jevsikova, T., Thota, N., Eckerdal, A., & Peters, A. K. (2012, November). Computer science at school/CS teacher education: Koli working-group report on CS at school. In *Proceedings of the 12th Koli Calling international conference on computing education research* (pp. 29–38).
56. Černiauskas, V., Dagienė, V., Kligienė, N., & Sapagovas, M. (2002). Some aspects of integration of information technologies into education. *Informatics in education*, 1, 43–60.
57. Dagienė, V., Dzemyda, G., & Sapagovas, M. (2006, November). Evolution of the cultural-based paradigm for informatics education in secondary schools–two decades of Lithuanian experience. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (pp. 1–12). Springer, Berlin, Heidelberg.
58. Hubwieser P., Armoni M., Brinda T., Dagiene V., Diethelm I., Gianakos M., Knobelsdorf M., Magenheim J., Mittermeir R., & Schubert S. (2011, June). Computer science/informatics in secondary education. In *Proceedings of the 16th annual conference reports on Innovation and technology in computer science education - working group reports* (pp. 19–38).
59. Education development center (2019). Lithuanian Informatics curriculum outline. Preschool and primary education. https://informatika.ugdome.lt/lt/biblioteka/dokumentai/
60. Dagienė, V. (1995). The first Informatics matura exam. *Informatika*, 24, 7–23.
61. Blonskis, J., & Dagienė, V. (2006, November). Evolution of informatics maturity exams and challenge for learning programming. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (pp. 220–229). Springer, Berlin, Heidelberg.
62. Blonskis, J., & Dagienė, V. (2008, July). Analysis of students' developed programs at the maturity exams in information technologies. In *International Conference on Informatics in Secondary Schools-Evolution and Perspectives* (pp. 204–215). Springer, Berlin, Heidelberg.
63. Dagiene, V., & Skupas, B. (2011, June). Semi-automatic testing of program codes in the high school student maturity exam. In *Proceedings of the 12th International Conference on Computer Systems and Technologies* (pp. 564–569).
64. Dagienė, V., Gulbahar, Y., Grgurina, N., López-Pernas, Saqr, M., Apiola, M., & Stupurienė, G. (2022). CER in Schools. In: *Past, Present and Future of Computing Education Research* (in-press). Springer.
65. Harel, I. E., & Papert, S. E. (1991). *Constructionism*. Ablex Publishing.
66. Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V., & Stupurienė, G. (2022). *Reviewing Computational Thinking in Compulsory Education - State of Play and Practices from Computing Education*.

# Computing Education Research in the Global South

**Friday Joseph Agbo, Maria Ntinda, Sonsoles López-Pernas, Mohammed Saqr, and Mikko Apiola**

## 1 Introduction

Ever since the birth of modern computing, there has been a need to arrange organised education in computing. Throughout the years, various initiatives, efforts, courses, workshops, curricula, and programs have been launched to teach computing [50]. Numerous systematic efforts to conduct research on how to teach and learn computing have taken place. Over the decades, computing education research (CER) has emerged as a respectable research specialisation of its own [43]. In the early days, publications on computing education were mostly experience reports or course descriptions. Today, CER has matured to a point where it has its own established publication venues, conceptual frameworks and theories, methodological recommendations, communities, professorships, and seminal publications [50].

From the overarching aim of CER, which includes education in computing context and broadening of knowledge on different aspects of computing in K-12, higher education, and beyond, it is important to understand how CER manifests

F. J. Agbo (✉)
University of Eastern Finland, Joensuu, Finland

Willamette University, Salem, OR, USA
e-mail: friday.agbo@uef.fi; fjagbo@willamette.edu

M. Ntinda
University of Namibia, Windhoek, Namibia

University of Turku, Turku, Finland
e-mail: nntinda@unam.na

S. López-Pernas · M. Saqr · M. Apiola
University of Eastern Finland, Joensuu, Finland
e-mail: sonsoles.lopez@uef.fi; mohammed.saqr@uef.fi; mikko.apiola@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_15

at global and regional levels. There is no doubt that CER has a huge influence on computing education globally through several channels, including developing curricula to support the uptake and integration of computing in schools. However, it is evident that CER, as a professional community, has had a strong grip in the West in terms of computing education in general [15, 54], whereas the Global South (GS) receives fewer contributions, as revealed by previous studies [15, 40]. Hence, there is a need to investigate the CER landscape of GS, in line with the identified gap, through the lens of scientometric analysis.

It is general knowledge that the concept of GS mainly refers to developing nations who may be politically, culturally, and economically marginalized [22]. This definition aligns with the United Nations global classification, but in the context of this paper, we extend the concept of GS to also include educational and scientific underachievement for development [30]. Within the CER community, for example, experts are raising concerns about the current situation of underrepresentation of participation, particularly from the GS [42]. This calls for further research to unravel the state of the art of scholarly contributions and progression in CER, not just from the global perspective, but with a specific focus on the GS context.

Over the years, many reviews, meta-reviews, and scientometric analyses of CER publications have been conducted [10, 40, 54]. Previous reviews have classified CER publications from many perspectives; analyses have focused on specific publication venues of CER [8, 10, 43], several reviews have targeted computer programming as a topic to teach [2, 28], while other research has investigated theory use, research design and methodology use in CER [4, 45, 49]. Modern scientometric studies have analysed collaboration networks, geographical diversity, and keyword trends in CER publications [9, 10, 15, 42, 54]. Many previous reviews show, e.g., the dominance of computer programming as a topic of research in CER [28]. Also, many previous analyses show that the field of CER, as measured by the origin of contributions and contributors, has been heavily dominated by high-income countries, especially the US [2, 9, 10, 42, 54]. A significantly smaller share of papers have originated from or have addressed challenges of CER in the GS [44]. For example, while computing is having massive influence in Africa, with a population of over 1.2 billion people, the needs for organised and high-quality education in computing have not been adequately met, and CER with origins in Africa is still only starting to emerge.

While the mainstream of CER has always focused on research done in the West, in this article we turn our focus towards the GS. As CER is constantly expanding its publication profile and diversifying its communities, it has become relevant to investigate those communities and their publication and citation habits with more depth. In this paper, we present a scientometric perspective on the evolution of CER, from the viewpoint of the GS. The modern methods of scientometrics offer the possibility to go beyond simple counts, and provide more mature and nuanced overviews of the evolution of science [25]. In this chapter, we give answer to the following research questions:

**RQ1:** How has CER in GS evolved over the years in terms of publication profile, keyword trends, and keyword clusters?

**RQ2:** What is the nature of authors' collaborations and who are the scholars of CER from the GS?
**RQ3:** How do international collaborations influence CER in the GS?

## 2 CER in Selected GS Countries

This section is dedicated to highlights of computing in general and CER in particular from selected GS countries, mainly from Africa and Asia. The intention is to provide a pinch of understanding, perhaps, from the historical perspective and even scientific contributions impacting CER. This overview may provide intrinsic insight into the development of computing in the region and, possibly, a justification for this study. It is interesting to note that, as far back as 1970s, when CER was still maturing [18], experts from the GS were contributing to the community by publishing papers in SIGCSE Technical Symposium [6, 16, 20]. But these earlier studies do not seem to receive a significant number of citations, which leaves one to wonder how CER in the context of GS has fared in the last four decades.

From the Asian continent, China, for example, has transitioned from "Electronic Computing" [52]—dedicated to designing hardware and software back in the 1960s—to "Computer Science and Technology", an encompassing name for computing majors at the higher education institutions. In the early 2000s, over 500 universities in China already offered Computer Science as a major. However, there was a lack of resources and standardized curricula for computing education across the universities as students enrolment rose. Notwithstanding, we can argue that the massive number of students enrolled into a CS degree, coupled with the higher percentage of institutions that offered CS courses in the early 2000, led to huge exploits in the software and hardware industries in China which, in turn, impacted their national economy greatly. Among the GS countries, China strives toward becoming one of the leading nations in technology advancements. To this end, their government developed an initiative to lead in the area of robotics, aviation, and advanced information technology by 2025 [53]. With efforts from different stakeholders, numerous developments and activities of computing are indeed taking place in China.

From the West African perspective, although the adoption of Western CS syllabi is widely evident [27], scholarly participation in CER from the Western part of the region is scarce or grossly under-represented in the globally acclaimed publishing outlets. For example, Nigeria and Ghana are advancing the field of CER from the Western Africa region [1, 3, 5, 11, 32, 33]. In Nigeria, the inception of computing education was traced to the establishment of the IBM African Education Centre in 1963 on the campus of the University of Ibadan. As Anyanwu [6] revealed, computing education in Nigeria gave rise to CER, which he and other colleagues started in the 1970s. Consequently, one could imagine that CER in Nigeria alone should have grown substantially as over 170 higher education institutions including universities are currently offering computer science degrees.

Furthermore, the southern African region is even doing a bit more in terms of CER compared to the rest of the continent [36, 41]. For example, one of the earlier works by Salt [39]—although affiliated to a university in the global north—set the pace for breeding indigenous programmers in southern Africa. From the eastern part of Africa, influential scholars from the West have built human capacities in computing over years [47, 48] and currently establishing pipelines of collaboration between universities in the West and GS in several ways to advance computing in that region [48]. Regarding the northern region of Africa, scanty studies focusing on CER are also found [9], a situation that points to the fact that there is low participation of computing educators in conferences for special interest groups such as the Technical Symposium on Computer Science Education (SIGSCE) and the ACM Conference on International Computing Education Research (ICER).

In general, there is a huge gap in terms of CER visibility from Africa compared to other global contexts. In addition, a recent study by Tshukudu et al. [51] investigated the state of computing education in selected African countries including Botswana, Kenya, Nigeria and Uganda and found that both lack of access to resources and limited professional development strategies were bedeviling the growth in the region. Consequently, the need for more efforts in advancing CER in the GS cannot be overemphasized since evidence points to potential human and infrastructural resources that could supports CS education from those regions.

## *2.1 Methods and Data*

In this research, we use the metadata of CER research published in Scopus. For a detailed explanation of the data retrieval, processing and analysis procedure, refer to the methods chapter in this book [12]. In order to explore the subset of CER data with an origin in the GS, we used the United Nations' Finance Center for South-South Cooperation's list,[1] which, as of early 2022, includes 78 countries, referred to as "Group of 77 and China." The subset of CER publications used in this chapter contains a total of $N = 1302$ articles in which at least one of the authors was affiliated with a GS institution at the time of publication. Modern scientometric methods were used to analyze this dataset [12, 40, 54]. More specifically, we analyze GS countries' impact and productivity over the years, the main research themes according to authors' keywords, the most cited papers, the most prolific authors, the collaboration among countries both from the GS and others, and the venues where most works have been published.

[1] https://worldpopulationreview.com/country-rankings/global-south-countries.

# 3 Results and Discussion

## *3.1 Evolution of CER in GS*

The most productive countries in GS in our dataset include China, Brazil, South Africa, India, Turkey, Saudi Arabia, Chile and Malaysia as shown in Table 1. USA and UK are among the top countries in the list since they and other Western countries were involved in collaboration with GS countries. It must be noted that if a paper had more than one author from the same country, this was only counted as one for that country, regardless of the number of authors.

The mean citation count per article from the GS was 4.36, compared to 6.7 in mixed articles (GS and others) and 8.04 to Non-GS. As presented in Fig. 1, the main difference was statistically significant and very small ($F(2, 16{,}860) = 13.76$, $p < 0.001$; $Eta^2 = 1.63e{-}03$, 95% CI [7.19e−04, 1.00]). One possible reason why the mean citation count per article for non-GS is almost doubled as compared to citation count to articles from GS is that perhaps CER educators and researchers from GS may have challenges in publishing in reputable journals and conferences with high visibility. Specific reasons may be related to funds, lacks in research culture and

**Table 1** Twenty most productive counties in GS CER

| Country | *n* | Pctg. | MCP | Total cit. | Mean cit./article | Mean cit./article/year |
|---|---|---|---|---|---|---|
| China | 255 | 19.59% | 26.27% | 1375 | 5.39 | 0.80 |
| Brazil | 201 | 15.44% | 19.4% | 1101 | 5.48 | 1.04 |
| USA | 201 | 15.44% | 100% | 1568 | 7.80 | 1.29 |
| South Africa | 123 | 9.45% | 17.89% | 658 | 5.35 | 0.58 |
| India | 111 | 8.53% | 28.83% | 529 | 4.77 | 0.88 |
| Turkey | 56 | 4.3% | 33.93% | 223 | 3.98 | 0.81 |
| United Kingdom | 55 | 4.22% | 100% | 231 | 4.20 | 0.94 |
| Saudi Arabia | 50 | 3.84% | 46% | 180 | 3.60 | 0.70 |
| Chile | 43 | 3.3% | 39.53% | 300 | 6.98 | 1.28 |
| Malaysia | 42 | 3.23% | 14.29% | 110 | 2.62 | 0.50 |
| Mexico | 38 | 2.92% | 42.11% | 350 | 9.21 | 0.95 |
| Colombia | 36 | 2.76% | 52.78% | 225 | 6.25 | 1.19 |
| Indonesia | 36 | 2.76% | 27.78% | 101 | 2.81 | 1.02 |
| Peru | 36 | 2.76% | 61.11% | 131 | 3.64 | 0.73 |
| Australia | 34 | 2.61% | 100% | 136 | 4.00 | 0.99 |
| Canada | 32 | 2.46% | 100% | 188 | 5.88 | 1.32 |
| Qatar | 29 | 2.23% | 55.17% | 136 | 4.69 | 0.63 |
| Thailand | 29 | 2.23% | 24.14% | 96 | 3.31 | 0.57 |
| New Zealand | 25 | 1.92% | 100% | 130 | 5.20 | 1.45 |
| Spain | 25 | 1.92% | 100% | 192 | 7.68 | 1.81 |

*n* Number of articles, $Pctg$ Percentage of articles, $MCP$ Percentage of articles in collaboration with other countries, $Cit.$ Citations

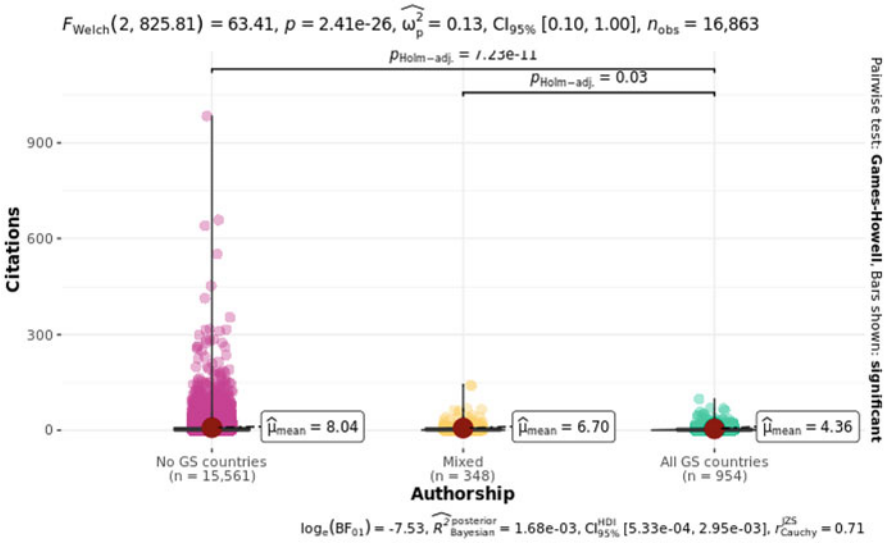


**Fig. 1** Comparison of the number of citations among articles with all authors from the GS, no authors from the GS, and mixed authors

researcher training to produce research and reports that are conducted with rigor, and formulated according to guidelines required by publication outlets. Lack of researcher training means a lack of research rigor, which also means that even highly novel contributions might not end up being published and therefore recognised. The situation might partly be mitigated through international collaboration.

Figure 2 shows the shares of articles over time in three categories: publications with no authors in GS countries (pink), publications with all authors in GS-countries (green), and publications with authors from both non-GS and GS countries (yellow). The total share of articles with origins in the GS is 1302 (7.7%) of which 954 (5.7%) were exclusively GS and 348 (2%) were in collaboration with non-GS countries. This result in itself revealed several issues related to the dominance of CER from the West. Even when collaboration happens as shown in the yellow color code, the result shows that it is very minimal, which may have an insignificant impact on CER from the GS. However, it is encouraging to see that collaboration between authors in GS countries is gaining momentum recently, although there is a dire need for improvement in this regard.

Figure 3 shows the distribution of CER within the GS countries. In line with Table 1, it can be seen that most publications in CER come from China and Brazil, followed by South Africa and India, whereas Peru, Mexico, Saudi Arabia, and Turkey are also making substantial contributions. Notwithstanding, there is generally minimal contributions in terms of CER article production emerging from Africa, Asia, and Southern America.

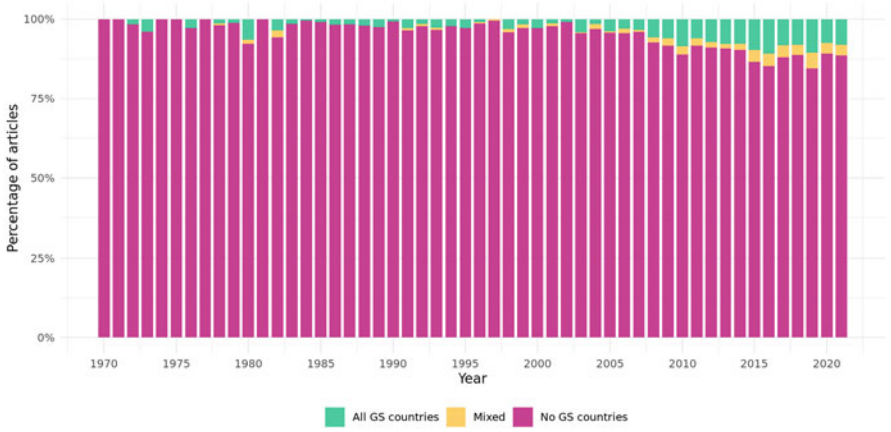


**Fig. 2** Evolution of CER in GS/percentages

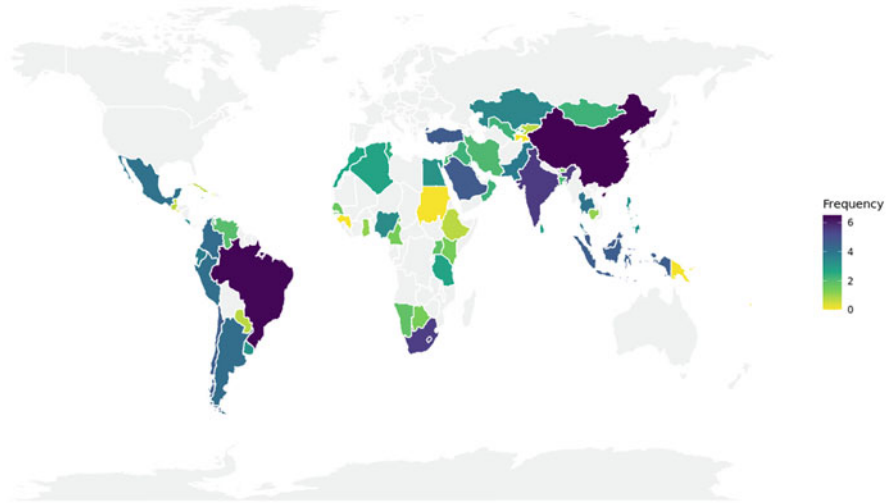


**Fig. 3** Map of CER distribution in GS

## *3.2 Top Keywords of CER from GS*

The top keywords, in this particular order (with search keywords removed) are: **programming education**, **CS1/CS2**, **computational thinking**, **K-12**, **curriculum**, **software engineering education**, **computing curriculum**, **e-learning**, **education**, **teaching**, **game-based learning**, **assessment**, **collaborative learning**, **novice programmers**, **active learning**, **gender and diversity**, **higher education** and **PBL** (problem-based learning). The list of top keywords raises several points for discussion. The list of top keywords has many similarities with the top keywords in CER

globally. A recent analysis of top keywords in CER [7, 9] shows that programming education and CS1/CS2, referring to research on first courses, computational thinking and K-12 are the top keywords, and many of the top other keywords, too, are similar. This essentially means that the terminology used to describe projects with regards to the top keywords is relatively similar. Surely, it does not mean that the projects are similar. For example, projects in the context of PBL (problem-based learning) may differ in numerous ways with regard to educational aims, context and its strengths and limitations, available resources, and practical implementation. Such differences are beyond scientometric analysis and further reviews or meta-reviews are needed in order to capture such differences.

For example, e-learning of educational technology projects may differ fundamentally between contexts. As such, projects take place within complex networks of actors: understanding such networks is important [46]. Poor socio-technical, socio-cultural or economic understanding of an educational context may result in failure of otherwise well-functioning technical solutions and pedagogically sound ideas [46]. Also, while, e.g., software engineering education is a hot topic both in the Global North and the GS, the nature of the projects may have fundamental differences. For example, while some demands of software industries may be global, e.g., good social and communication skills [24], the challenges in different educational contexts may be fundamentally different. For example, while employers in Norway evaluated software engineering graduates' skills to be adequate [29], research in Southern Africa identified severe mismatches between university graduates' skills and industries' expectations [31]. Each educational environment has its unique strengths, challenges, and characteristics, making projects on seemingly similar topics (such as educational technology, software engineering education, computational thinking or problem-based learning) unique in each context.

As presented in Table 2, aside from programming education (CS1/CS2) that remains one of the top in the list of authors' keywords, abstract trigrams, and bigrams, computational thinking or computational thinking skills is also a hot topic in CER from GS. Keywords such as educational data mining and learning analytics, which are typically emerging and growing fields in the global CER as shown in [9], were absent in the list of keywords in CER from GS. Figure 4 shows constellations of keywords most commonly found together, and identifies clusters of keywords. Four clear clusters are formed from the keywords. First, the *yellow cluster* centers around programming and related issues such as programming languages (e.g., Python), recursion, related pedagogical approaches such as problem-based learning, and gamification. The *pink cluster* is oriented towards computational thinking and K-12 education, related tools and educational technologies such as Scratch, Alice, block-based programming, informal education, and *constructionism*, a current more oriented towards making practical artefacts than its relative concept *constructivism* in the yellow cluster. The *green cluster* is oriented in educational technologies, pedagogy, capstone projects in software engineering, collaborative learning, mobile learning, and other educational concepts. The *gold cluster* centers around human-computer interaction, novice programmers, design guidelines, and programming concepts such as compilers and error messages.

**Table 2** Twenty top keywords of CER from GS

| Author keywords | Freq. | Abstract trigrams | Freq | Abstract bigrams | Freq |
|---|---|---|---|---|---|
| CS EDUCATION | 349 | CS EDUCATION | 178 | CS | 1026 |
| PROGRAMMING EDUCATION | 123 | CS CURRICULUM | 66 | SCIENCE EDUCATION | 189 |
| COMPUTING EDUCATION | 98 | CS STUDENTS | 50 | COMPUTATIONAL THINKING | 161 |
| CS1/CS2 | 71 | CS COURSES | 46 | COMPUTING EDUCATION | 155 |
| COMPUTATIONAL THINKING | 61 | CS CURRICULA | 36 | SOFTWARE ENG. | 147 |
| K-12 | 60 | LEARNING CS | 35 | COMPUTER PROGRAMMING | 89 |
| CURRICULUM | 48 | CS TEACHERS | 32 | INTRO. PROGRAMMING | 87 |
| SOFTWARE ENG. EDUCATION | 47 | TEACHING CS | 30 | INFORMATION TECHNOLOGY | 71 |
| COMPUTING CURRICULUM | 46 | INTRO. PROGRAMMING COURSES | 27 | LEARNING PROCESS | 71 |
| E-LEARNING | 44 | CS PROGRAMS | 26 | PROGRAMMING LANGUAGE | 71 |
| EDUCATION | 33 | INTRO. CS | 26 | SCIENCE CURRICULUM | 68 |
| TEACHING | 32 | UNDERGRADUATE CS | 24 | STUDENTS LEARNING | 68 |
| GAME-BASED LEARNING | 29 | COMPUTATIONAL THINKING SKILLS | 23 | LEARNING ENVIRONMENT | 65 |
| ASSESSMENT | 25 | CS CONCEPTS | 21 | PROGRAMMING COURSES | 64 |
| COLLABORATIVE LEARNING | 24 | PRIOR PROGRAMMING EXPERIENCE | 14 | SOURCE CODE | 62 |
| NOVICE PROGRAMMERS | 23 | CS TEACHING | 13 | DATA STRUCTURES | 59 |
| ACTIVE LEARNING | 22 | SOURCE CODE PLAGIARISM | 13 | SOFTWARE DEVELOPMENT | 55 |
| GENDER AND DIVERSITY | 22 | CS MAJORS | 12 | SCIENCE STUDENTS | 53 |
| HIGHER EDUCATION | 22 | CS PROGRAM | 12 | PROGRAMMING CONCEPTS | 52 |
| PBL | 21 | INTELLIGENT TUTORING SYSTEM | 12 | UNDERGRADUATE STUDENTS | 49 |

*CS* Computer Science, *PBL* Project Based Learning

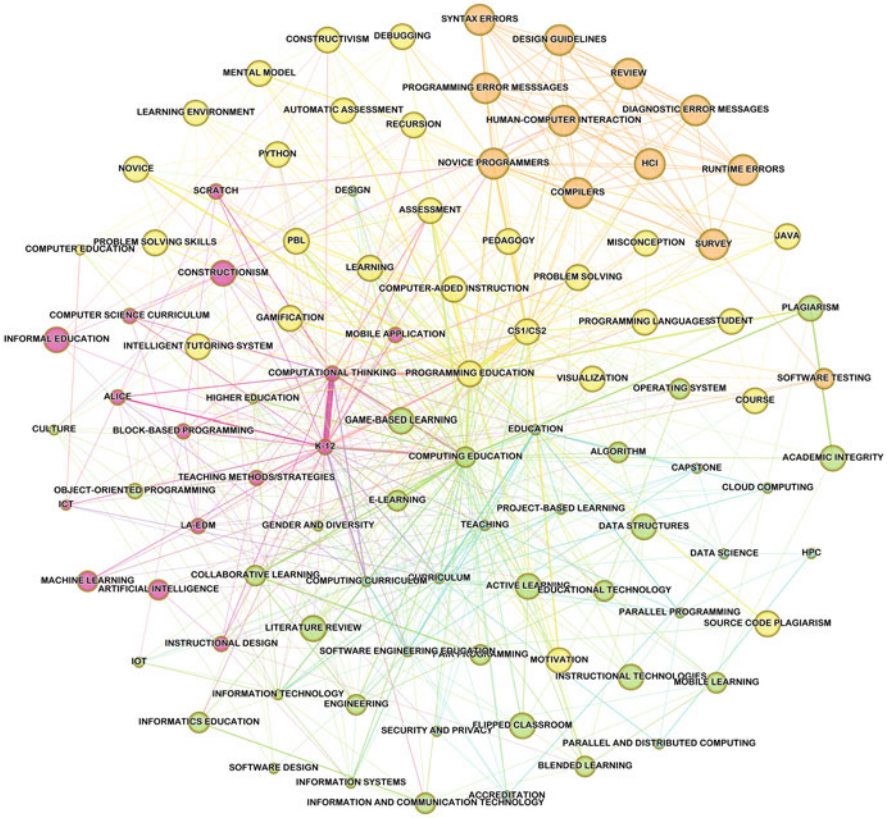


**Fig. 4** Top keywords

Figure 5 provides the usage of keywords in the last decade. This visualization showcases the trend and thematic focus of CER in GS over the years. The analysis simply tells that *computer science education* in GS was at its peak in 2019. However, from 2020, there is a gradual fall in the use of this keyword. One could think that this fall may be connected to the general closure of universities and other educational institutions during the COVID-19 pandemic, which in that case, could be mitigated by following the professional advice provided by Barocas [13], according to which collaboration with other experts could be one of the alternative plans to promote CER during pandemic.

On the other hand, we can see that keywords such as *programming education* and *computing education*—although they had a sharp fall in 2020, which might be due to the reasons already highlighted—began to rise from 2021 unlike the keyword *computer science education*. In recent times, the keyword *computing education* is more common in CER papers compared to *computer science education*. Furthermore, recent trending keywords in CER from GS show *K-12* and *computational*

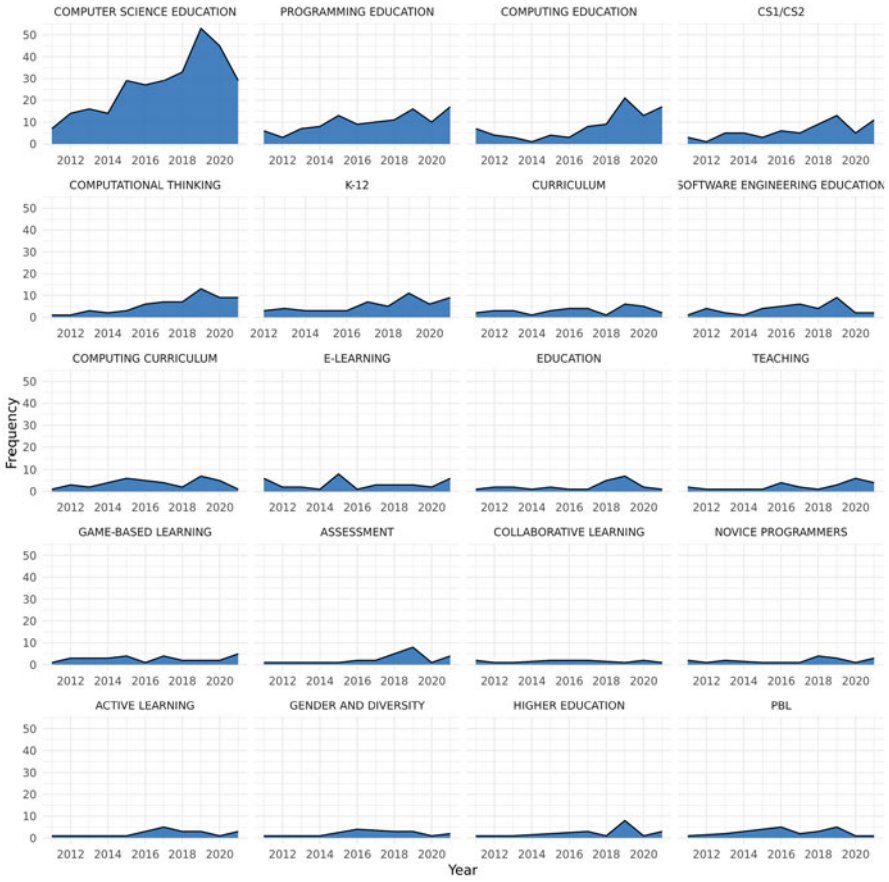


**Fig. 5** Keyword usage trends in the last decade

*thinking*. This finding is consistent with the trending keywords of CER from the global perspective as computational thinking in K-12 is seen to foster broadening participation in computing education.

## *3.3 Top Cited Papers*

Table 3 shows the most cited papers within the dataset. First, it must be noted that some papers in the dataset may report research done, e.g., in the USA while one or more of the authors have an affiliation with a GS institution. In the context of the top cited papers, we have excluded such works. The top cited work in the dataset was [19], a review of computational thinking (CT) with a focus on computer

**Table 3** Top 18 most cited articles

| Title | Authors | Country | Year | Citations |
|---|---|---|---|---|
| Changing a generation's way of thinking: teaching computational thinking through programming | Buitrago Flórez F., Casallas R., Hernández M., Reyes A., Restrepo S., Danies G | Colombia | 2017 | 98 |
| Computational thinking in educational activities: an evaluation of the educational game light-bot | Gouws L., Bradshaw K., Wentworth P | South Africa | 2013 | 69 |
| A global snapshot of computer science education in K-12 schools | Hubwieser P., Giannakos Mn., Berges M., Brinda T., Diethelm I., Magenheim J., Pal Y., Jackova J., Jasute E | Germany, India, Lithuania, Norway, Slovakia | 2015 | 68 |
| How games for computing education are evaluated? A systematic literature review | Petri G., Von Wangenheim Cg | Brazil | 2017 | 68 |
| Affective and behavioral predictors of novice programmer achievement | Rodrigo Mmt., Baker Rs., Jadud Mc., Amarra Acm., Dy T., Espejo-Lahoz Mbv., Lim Sal., Pascua Sams., Sugay Jo., Tabanao Es | Philippines, USA | 2009 | 65 |
| Coarse-grained detection of student frustration in an introductory programming course | Rodrigo Mmt., Baker Rs | Philippines, USA | 2009 | 63 |
| Effect of think-pair-share in a large CS1 class: 83% Sustained Engagement | Kothiyal A., Majumdar R., Murthy S., Iyer S | India | 2013 | 62 |
| Predicting at-risk novice java programmers through the analysis of online protocols | Tabanao Es., Rodrigo Mmt., Jadud Mc | Philippines, USA | 2011 | 61 |

| | | | | |
|---|---|---|---|---|
| Virtual machines—an idea whose time has returned: application to network, security, and database courses | Bullers Jr Wi., Burd S., Seazzu Af | Mexico | 2007 | 57 |
| Design, development, and evaluation of a mobile learning application for computing education | Oyelere Ss., Suhonen J., Wajiga Gm., Sutinen E | Finland, Nigeria | 2018 | 55 |
| Integrating the teaching of computer organization and architecture with digital hardware design early in undergraduate courses | Calazans Nlv., Moraes Fg | Brazil | 2001 | 52 |
| Mental models of recursion | Götschi T., Sanders I., Galpin V | South Africa | 2003 | 51 |
| Development of computational thinking skills through unplugged activities in primary school | Brackmann Cp., Moreno-León J., Román-González M., Casali A., Robles G., Barone D | Argentina, Brazil, Spain | 2017 | 48 |
| Applying web-enabled problem-based learning and self-regulated learning to add value to computing education in taiwan's vocational schools | Lee T-H., Shen P-D., Tsai C-W | China | 2008 | 47 |
| Do students need teacher's initiation in online collaborative learning? | Tsai C-W | China | 2010 | 46 |
| Combining challenge-based learning and scrum framework for mobile application development | Santos Ar., Sales A., Fernandes P., Nichols M | Brazil, USA | 2015 | 43 |
| Conceptual models and cognitive learning styles in teaching recursion | Wu C-C., Dale N., Bethel Lj | China, USA | 1998 | 42 |
| Developing a Nfc-equipped smart classroom: effects on attitudes toward computer science | Shen C-W., Wu Y-Cj., Lee T-C | China., China | 2014 | 41 |

programming and tools, authored by an all-Colombian author team. The study contextualises the findings in Colombia, in describing, discussing, and challenging approaches of exposing learners to mere usage of computer applications, without a clearly defined set of skills to learn, and describing case examples of positive changes taking place in a limited number of Colombian specific schools, raising the crucial importance of teaching algorithmic thinking, problem solving, debugging, and other related thinking skills, and related curricular reformations that are needed [19]. The second most cited work in the results set was a research by authors-team from South Africa on the topic of computational thinking, reporting on the development of a contextualised computational thinking framework, with a concrete application in the context of an educational game focusing on programming [21].

The third most cited paper in the dataset is a categorisation of K-12 situations in 12 countries, most of which are in the Global North, and including India, mapping the intended goals and competencies, taught content, applied programming language and tools, presenting a snapshot of K-12 computing education in varying contexts [23]. The framework in the research could be applied to cover also GS countries other than India. In the fourth most cited work in the dataset, researchers from Brazil [35] conducted a systematic literature review about evaluating games in computing education, showing limited consensus on evaluation metrics, prevalence of ad-hoc evaluations and lack of rigor in evaluation. The fifth most cited paper was conducted in the Philippines, on the topic of affective states and behaviours and their relationship with achievement in CS1 programming, showing, e.g., associations between confusion, boredom and low achievement [38]. Certain patterns of student performance in learning tasks were also found to be good predictors of exam performance, and therefore useful in detecting students at risk [38]. The rest of the top cited papers in the dataset include research from Philippines, and India, on the topics of introductory programming courses and predicting at-risk students. In one paper, a contextually relevant pedagogical method called Think-Pair-Share (TPS) is built, in which students work on programming problems first individually, then in pairs, and finally by engaging in class-wide discussions, applied in a large CS1 class [26].

### *3.4 Prolific Authors and Collaborations*

Figure 6 shows the top 10 authors who appear in papers who have at least one author affiliated with an institution in a GS-country. The list includes **John Impagliazzo** ($n = 26$), a top contributing CER author, **Oscar Karnalim** ($n = 22$) from Maranatha Christian University in Indonesia, **Ian Sanders** ($n = 19$) from University of the Witwatersrand in South Africa, **Sridhar Iyer** from Indian Institute of Technology ($n = 15$), **Christiane Gresse von Wangenheim** from Federal

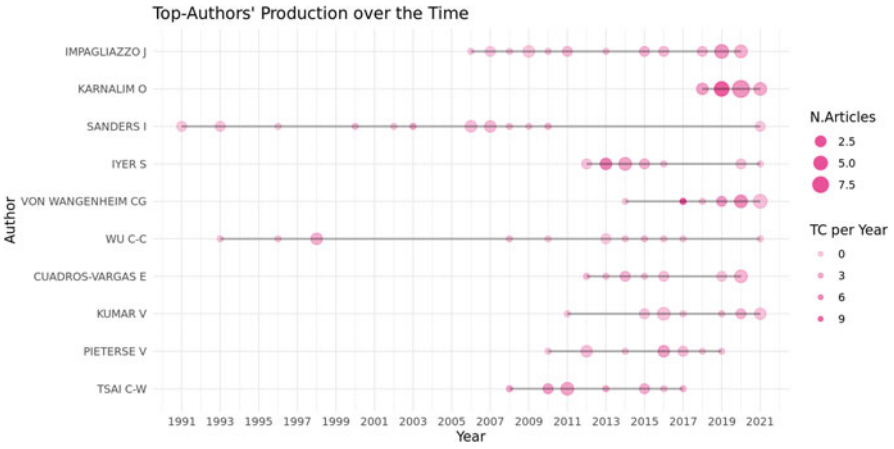


**Fig. 6** Top GS authors' production

University of Santa Catarina in Brazil ($n = 14$), **Cheng-Chih Wu** from National Taiwan Normal University ($n = 14$), **Ernesto Cuadros-Vargas** from Universidad Católica San Pablo in Peru ($n = 13$), among others. This list is not a ranking list but rather a sample of influential researchers, who are based in GS countries, or who have worked in collaboration between the so-called Global North (West) and GS in computing education.

A network of collaborations between GS and non-GS countries is visualized in Fig. 7. The USA has the strongest connection in terms of authors' collaboration with GS countries such as China, Brazil, India, Egypt, Peru, Mexico, and other countries. Some of the non-GS countries have only little collaborations with GS-countries, while others have more. For example, Finland has established collaborations with Tanzania, Nigeria, South Africa, and Morocco. In addition, the network also shows how authors from Spain have collaborated with other colleagues from GS countries such as Ecuador, Chile, and Mexico. On the contrary, there are a few strong collaborations between authors who are based in GS. For example, South African scholars collaborate with their colleagues from Kenya, Tanzania, and Ethiopia; China collaborates with other scholars from Thailand, Saudi Arabia, Indonesia, and Mongolia; whereas, in South America, there is a weak collaboration between scholars from Brazil, Colombia, Chile, and Argentina. In general, a strong collaboration was seen among authors from non-GS and GS, which may explain why many of the citations were found to come from the West. While these influential relationships can be perceived to bring some limitations on CER from GS [15], it can also mean that GS authors can gain good practices through these collaborations into the classrooms. One must, however, keep in mind that in order to become effective, curricula almost always requires contextualisation rather than plain adoption.

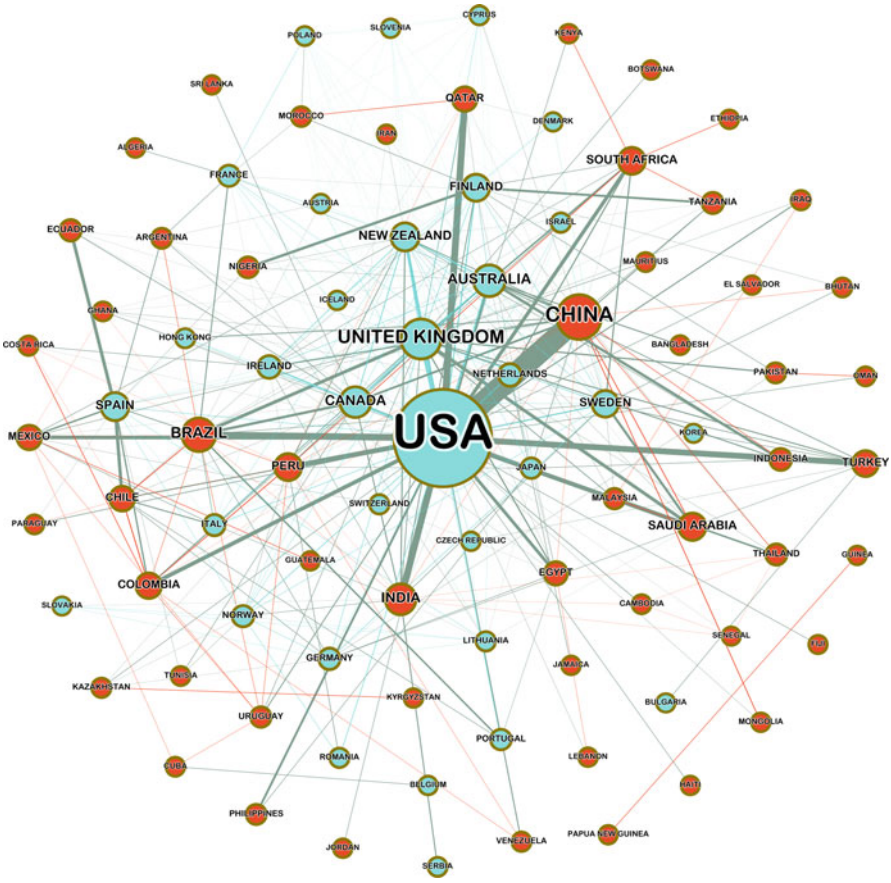


**Fig. 7** Network of collaborations: GS-countries are colored as red, while non-GS countries are colored as blue

## *3.5 Institutions and Venues*

Table 4 lists the top venues where research in this dataset has been published. All the venues in the list are well-known venues where CER is typically published. It must be noted as a limitation that researchers in the GS may publish CER in many such venues that have fallen outside of our search. Such venues may include local venues that are not indexed in Scopus, or venues that publish in languages other than English.

It should be noted that some publications in the dataset may report research done, e.g., in the USA, while one or more of the authors have an affiliation with a GS institution. In the context of the top cited papers, we have excluded such research. Moreover, many researchers from the GS pursue a PhD and even their whole career

**Table 4** Top venues with the highest number of papers about CER from GS authors based on metadata in Scopus

| Venue | No. articles | Percent |
|---|---|---|
| Innovation and Technology in Computer Science, ITiCSE | 192 | 14.75 |
| ACM Technical Symposium on Computer Science Education, SIGCSE | 118 | 9.06 |
| Frontiers in Education Conference, FIE | 67 | 5.15 |
| ACM SIGCSE Bulletin | 52 | 3.99 |
| Computer Science Education | 23 | 1.77 |
| Computer Science Education Research Conference, CSERC | 21 | 1.61 |
| International Conference on Educational Research, ICER | 20 | 1.54 |
| International Conference on Learning and Teaching in Computing and Engineering, LATICE | 18 | 1.38 |
| Koli Calling International Conference on Computing Education Research | 18 | 1.38 |
| Computer Applications in Engineering Education | 15 | 1.15 |

outside of the GS. Therefore, despite their origins, their publications with a non-GS affiliation are not included in our dataset.

## 4 Reflection and Conclusions

This study investigates the status of CER in GS through a scientometric lens. Examining the CER landscape from the perspective of GS is extremely important to undertake measures that will prevent underdevelopment of the field in those regions. Recent studies have revealed how CER in the West is tremendously advancing whereas the GS is far behind in terms of article production, citations, and research impact [9, 40]. In addition, the campaign for inclusion and broadening of participation in CER across all geographical locations [15] can be supported if there is evidence that showcases the status of the field from different regions. To provide insights into how CER has progressed in the context of the GS, research questions were formulated to guide this study, and results and discussion are presented inline with these research questions in order to address them.

*RQ1: How has CER in GS evolved over the years in terms of publication profile, keyword trends, and keyword clusters?*

This study shows that China, Brazil, South Africa, India, Turkey, Saudi Arabia, Chile, and Malaysia are among the top contributing countries publishing CER articles from the GS. Despite the amount of CER articles from the GS, the citation count is half the mean average compared to citation count of articles from the West. Several reasons could account for this disparity, including the GS researchers not able to publish in reputable journals and conferences with high visibility, lack of funds, and limited research training to produce research and reports. In addition, topics such as contextualising curriculum or pedagogies for GS contexts may be

such niche areas of research that they do not attract citations simply because not many scholars are working on such topics. This highlights a fundamental challenge with citation-based rankings: the amount of citations is a very weak measure of importance or quality of research, since it only reflects what researchers at a given moment are mostly interested at.

It was found that authors common keywords in CER from the GS are mainly programming education, CS1/CS2, computational thinking, K-12, curriculum, software engineering education, computing curriculum, e-learning, education, teaching, game-based learning, assessment, collaborative learning, novice programmers, active learning, gender and diversity, higher education, and PBL (problem-based learning). These keywords aligned with CER keywords from the West [9]. However, it was revealed in this study that keywords such as educational data mining and learning analytics, which are typically emerging and growing fields in the global space of CER as shown in [9], were absent in the list of keywords in CER from the GS. Regarding trending keywords of CER from the GS, our finding shows K-12 and computational thinking as the dominant keywords.

*RQ2: What is the nature of authors' collaborations and who are the scholars of CER from the GS?*

The analysis of influential researchers who are based in GS countries, or who have collaborated with authors from the West in CER, shows several names including John Impagliazzo, Oscar Karnalim from Maranatha Christian University in Indonesia, Ian Sanders from University of the Witwatersrand in South Africa (having longer years of CER experience), Sridhar Iyer from Indian Institute of Technology, Christiane Gresse von Wangenheim from Federal University of Santa Catarina in Brazil, Cheng-Chih Wu from National Taiwan Normal University, Ernesto Cuadros-Vargas from Universidad Católica San Pablo in Peru, among others. As revealed in the analysis, the strongest collaboration occurs between researchers in China and USA. In addition, this study shows that there are a few strong collaborations between authors who are based in GS, which is a good indicator of synergy among the GS authors which could foster regional CER development.

*RQ3: How do international collaborations influence CER in the GS?*

We argue that collaborations that exist between researchers from the West and GS could foster advancement in both contexts in several ways. For example, as revealed in this study, Finland has established collaborations with Tanzania, Nigeria, South Africa, and Morocco. In some of these African countries such as Tanzania, Finnish professors have accomplished capacity building for educators across education institutions in the recent past [48]. Similarly, researchers from the West can also learn from the GS counterparts as they can gain through experience how different contexts engage in and react to their professional commitment towards addressing inclusion and diversity of computing education.

Generally, the finding of this study show that CER in the GS is still maturing and there are few but important studies conducted to address some of the challenges in teaching and learning of computing at different educational levels. The evolution has recorded substantial progress in the volume of publications starting from 2015.

Most of the publications are from China in Asia, Brazil in South America, and South Africa in Africa. In addition, other countries such as India, Saudi Arabia, Peru, Nigeria, and Tanzania are making significant contributions in terms of volume of publications. However, there are quite a few publications that are co-authored between authors from the West and GS. This kind of collaboration can create opportunities for maturing CER and computing in general [16] by exposing authors from the GS to good practices in CS by simply co-authoring with experts from the West. These collaborations may also teach researchers from the West many lessons, which might otherwise be difficult to learn, and, importantly, help to reshape and address some of the crucial challenges that CER in the West is facing.

Regarding curriculum, most of the universities in the GS are yet to key into the current view of computing curriculum [17]. Besides, adoption of international curricula is a common practice in many African universities, with many strengths but also limitations [14]. While the commonly used curricula, such as those made by ACM (Association for Computing Machinery) or IEEE (Institute of Electrical and Electronics Engineers) were designed primarily for Western rather than African realities, successful implementation, and adaptation, require special efforts [14]. For example, adoption of international curricula within Africa may face particular challenges for two reasons: (1) under-investment and under-resourcing of higher education, and (2) contextual issues, i.e., curricula designed primarily for the US, or at least Western nations that may not fit into Africa context.

With all these existing realities, experts from the West, particularly in Europe, have been contributing to advancing computing education in the GS context. For example, Namibia, Tanzania, Eritrea, and other parts of sub-Saharan Africa have benefited from projects related to capacity building in Computing education [5, 34, 48]. Furthermore, a strong collaboration exists between countries in the West and some countries from the GS. For example, it was observed that USA has a strong connection with China, which may explain why most of the publications of CER from GS emerged from China, and perhaps may be attributed to PhD students from China studying in the US. Other prominent collaborations involve Western, predominately English speaking countries, and less of Spanish speaking countries.

Some limitations are inherent in this study that are worth mentioning. Here we looked only at typical publication venues of CER within the Scopus database. Despite its massive volume of indexed metadata, Scopus has a vetting mechanism that was heavily criticized for its lack of inclusiveness and bias in several ways [37], for example, against non-English speaking countries. In fact, most major databases e.g., Microsoft Academic, Dimensions, and Web of Science, are neither complete nor inclusive and effectively exclude most scholarly work from the GS. It is also worth mentioning that CS publishes a significant number of papers in conferences which are inaccessible to most researchers in GS due to visa problems, conference fees, and travel expenses. In other words, the current publication system in CS—and science at large—hinders the contribution of GS researchers. Consequently, GS researchers have their own venues of publication that are either local venues, non-English speaking, or local events that may be non-indexed. Therefore, our data is inherently biased, non-representative and skewed. Nevertheless, it is far from feasible to collect all local journals and events.

As a future study, efforts maybe needed on a global scale to create more inclusive and representative database of current knowledge. In the same token, researchers in the GS have difficulties accessing the global body of knowledge that is pay-walled with high subscription fees. There are serious trials for connection with the GS from researchers, institutions and funding agencies. Yet, there are also barriers that contribute to a global divide, namely, the current difficulties in publishing and travelling that GS researchers face, which minimizes their opportunities in connecting with other researchers. This chapter is our initial exploration of CER in the GS. We think of our work as revealing the difficulties, the gaps, and the future directions rather than a reflection of the current realities. Future research must be directed at curating a more representative data which while an ardours process, it is still worthwhile.

From this study, we can reflect that under-investment and under-resourcing of education, mismanagement, and limited capacity building strategy are part of the issues causing under-representation of GS in the global map of CER community. In this twenty-first century and onwards, donor institutions especially in the West should recognise, promote and push higher education in the GS as part of the development agenda. In addition, not only an increase of resourcing is required to boost CER in the GS but also development of new curricula, especially STEM. In addition, it is expected that CER scholars and computing educators from the West understudy how curricula they have developed also integrate into the GS classrooms or whether they require some contextualization. It is until this understanding is reached, generalization of computing curricula across the globe may be difficult.

## References

1. Agbo, F.J.: Co-designing a smart learning environment to facilitate computational thinking education in the nigerian context. Ph.D. thesis, Itä-Suomen yliopisto (2022)
2. Agbo, F.J., Oyelere, S.S., Suhonen, J., Adewumi, S.: A systematic review of computational thinking approach for programming education in higher education institutions. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research, pp. 1–10 (2019)
3. Agbo, F.J., Oyelere, S.S., Suhonen, J., Laine, T.H.: Co-design of mini games for learning computational thinking in an online environment. Education and information technologies **26**(5), 5815–5849 (2021)
4. Agbo, F.J., Yigzaw, S.T., Sanusi, I.T., Oyelere, S.S., Mare, A.H.: Examining theoretical and pedagogical foundations of computational thinking in the context of higher education. In: 2021 IEEE Frontiers in Education Conference (FIE), pp. 1–8. IEEE (2021)
5. Anohah, E., Suhonen, J.: Measuring effect of culturally responsive learning environment for computing education in african context. Problems of Education in the 21st Century **73**, 6 (2016)
6. Anyanwu, J.: Computer science education in a developing nation. In: Papers of the SIGCSE/CSA technical symposium on Computer science education, pp. 37–40 (1978)
7. Apiola, M., López-Pernas, S., Saqr, M.: The evolving themes of computing education research: Trends, topic models, and emerging research. In: M. Apiola, S. López-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research. Springer (2023)

8. Apiola, M., López-Pernas, S., Saqr, M., Pears, A., Daniels, M., Malmi, L., Tedre, M.: From a national meeting to an international conference: A scientometric case study of a finnish computing education conference. IEEE Access **10**, 66576–66588 (2022). DOI https://doi.org/10.1109/ACCESS.2022.3184718
9. Apiola, M., Saqr, M., López-Pernas, S., Tedre, M.: Computing education research compiled: Keyword trends, building blocks, creators, and dissemination. IEEE Access **10**, 27041–27068 (2022). DOI https://doi.org/10.1109/ACCESS.2022.3157609
10. Apiola, M., Tedre, M., López-Pernas, S., Saqr, M., Daniels, M., Pears, A.: A scientometric journey through the fie bookshelf: 1982–2020. In: 2021 IEEE Frontiers in Education Conference (FIE), pp. 1–9 (2021). DOI https://doi.org/10.1109/FIE49875.2021.9637209
11. Asabere, N.Y., Torgby, W.K., KwameGyamfi, N.: Towards a perception of computing related programmes offered by public tertiary institutions in Ghana. International Journal of Computer Applications **975**, 8887 (2013)
12. Lépez-Pernas, S., Saqr, M., Apiola, M.: Scientometrics: a concise introduction and a detailed methodology for the mapping of the scientific field of computing education research. In: M. Apiola, S. Lépez-Pernas, M. Saqr (eds.) Past, Present and Future of Computing Education Research, Springer (2023)
13. Barocas, S., Biega, A.J., Boyarskaya, M., Crawford, K., Iii, H.D., Dudík, M., Fish, B., Gray, M.L., Hecht, B., Olteanu, A., et al.: Responsible computing during covid-19 and beyond. Communications of the ACM **64**(7), 30–32 (2021)
14. Bass, J.M., Heeks, R.: Changing computing curricula in african universities: Evaluating progress and challenges via design-reality gap analysis. The Electronic Journal of Information Systems in Developing Countries **48**(1), 1–39 (2011)
15. Becker, B.A., Settle, A., Luxton-Reilly, A., Morrison, B.B., Laxer, C.: Expanding opportunities: Assessing and addressing geographic diversity at the sigcse technical symposium. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, pp. 281–287. Association for Computing Machinery, New York, NY, USA (2021). URL https://doi.org/10.1145/3408877.3432448
16. Chvalovsky, V.: Computer science education at universities: the case of developing countries. In: Papers of the SIGCSE/CSA technical symposium on Computer science education, pp. 41–47 (1978)
17. Clear, A., Parrish, A.S., Impagliazzo, J., Zhang, M.: Computing curricula 2020: introduction and community engagement. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, pp. 653–654 (2019)
18. Fincher, S.A., Robins, A.V.: The Cambridge handbook of computing education research. Cambridge University Press (2019)
19. Flórez, F.B., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., Danies, G.: Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. Review of Educational Research **87**(4), 834–860 (2017). URL https://doi.org/10.3102/0034654317710096
20. Gonzales, C.: A computer engineering degree in mexico. In: Papers of the SIGCSE/CSA technical symposium on Computer science education, pp. 48–52 (1978)
21. Gouws, L.A., Bradshaw, K., Wentworth, P.: Computational thinking in educational activities: An evaluation of the educational game light-bot. In: Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '13, pp. 10–15. Association for Computing Machinery, New York, NY, USA (2013). URL https://doi.org/10.1145/2462476.2466518
22. Hollington, A., Salverda, T., Schwarz, T., Tappe, O.: Concepts of the global south (2015)
23. Hubwieser, P., Giannakos, M.N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., Pal, Y., Jackova, J., Jasute, E.: A global snapshot of computer science education in k-12 schools. In: Proceedings of the 2015 ITiCSE on Working Group Reports, ITICSE-WGR '15, pp. 65–83. Association for Computing Machinery, New York, NY, USA (2015). URL https://doi.org/10.1145/2858796.2858799

24. Isomöttönen, V., Daniels, M., Cajander, A., Pears, A., Mcdermott, R.: Searching for global employability: Can students capitalize on enabling learning environments? ACM Trans. Comput. Educ. **19**(2) (2019). URL https://doi.org/10.1145/3277568
25. Ivancheva, L.: Scientometrics today: A methodological overview. Collnet Journal of Scientometrics and Information Management **2**(2), 47–56 (2008)
26. Kothiyal, A., Majumdar, R., Murthy, S., Iyer, S.: Effect of think-pair-share in a large cs1 class: 83% sustained engagement. In: Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research, ICER '13, pp. 137–144. Association for Computing Machinery, New York, NY, USA (2013). URL https://doi.org/10.1145/2493394.2493408
27. Kroeze, J.H., Prinsloo, P., Ponelis, S., Venter, I., Pretorius, P.: Graduateness of computing students in a Sub-Saharan African context. In: Proceedings of the Americas' Conference on Information Systems (AMCIS) (2012). URL https://aisel.aisnet.org/amcis2012/proceedings/Posters/11
28. Lee, S.J., Francom, G.M., Nuatomue, J.: Computer science education and k-12 students' computational thinking: A systematic review. International Journal of Educational Research **114**, 102008 (2022)
29. Lundberg, G.M., Krogstie, B.R., Krogstie, J.: Becoming Fully Operational: Employability and the Need for Training of Computer Science Graduates. In: 2020 IEEE Global Engineering Education Conference (EDUCON), pp. 644–651 (2020). DOI https://doi.org/10.1109/EDUCON45650.2020.9125188
30. Medie, P.A., Kang, A.: Global south scholars are missing from european and us journals. what can be done about it. University of Nebraska - Lincoln Faculty Publications: Political Science 100, 1–4 (2018). https://digitalcommons.unl.edu/poliscifacpub/100
31. Ntinda, M., Apiola, M., Sutinen, E.: Mind the Gap: Aligning Software Engineering Education and Industry in Namibia. In: Proceedings of IST-Africa 2021 Conference (2021)
32. Oyelere, S.S., Agbo, F.J., Sanusi, I.T., Yunusa, A.A., Sunday, K.: Impact of puzzle-based learning technique for programming education in nigeria context. In: 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT), vol. 2161, pp. 239–241. IEEE (2019)
33. Oyelere, S.S., Suhonen, J.: Design and implementation of mobileedu m-learning application for computing education in nigeria: A design research approach. In: 2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE), pp. 27–31. IEEE (2016)
34. Oyelere, S.S., Suhonen, J., Wajiga, G.M., Sutinen, E.: Design, development, and evaluation of a mobile learning application for computing education. Education and Information Technologies **23**(1), 467–495 (2018). URL https://doi.org/10.1007/s10639-017-9613-2
35. Petri, G., Gresse von Wangenheim, C.: How games for computing education are evaluated? a systematic literature review. Computers & Education **107**, 68–90 (2017). DOI https://doi.org/https://doi.org/10.1016/j.compedu.2017.01.004. URL https://www.sciencedirect.com/science/article/pii/S0360131517300040
36. van der Poll, A., van Zyl, I., Kroeze PhD IT, J.H.: Towards decolonizing and africanizing computing education in south africa. Communications of the Association for Information Systems **47**(1), 7 (2020)
37. Pranckutė, R.: Web of science (wos) and scopus: The titans of bibliographic information in today's academic world. Publications **9**(1), 12 (2021)
38. Rodrigo, M.M.T., Baker, R.S., Jadud, M.C., Amarra, A.C.M., Dy, T., Espejo-Lahoz, M.B.V., Lim, S.A.L., Pascua, S.A., Sugay, J.O., Tabanao, E.S.: Affective and behavioral predictors of novice programmer achievement. In: Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE '09, pp. 156–160. Association for Computing Machinery, New York, NY, USA (2009). URL https://doi.org/10.1145/1562877.1562929
39. Salt, N.F.: 70's programming style for a developing country programming. In: Proceedings of the 1979 Annual Conference, ACM '79, p. 128–134. Association for Computing Machinery, New York, NY, USA (1979). URL https://doi.org/10.1145/800177.810048

40. Saqr, M., Ng, K., Oyelere, S.S., Tedre, M.: People, ideas, milestones: a scientometric study of computational thinking. ACM Transactions on Computing Education (TOCE) **21**(3), 1–17 (2021)
41. Scholtz, B.M.: An investigation into the learnability of object-oriented case tools for computing education. Ph.D. thesis (2008)
42. Settle, A., Becker, B.A., Duran, R., Kumar, V., Luxton-Reilly, A.: Improving Global Participation in the SIGCSE Technical Symposium: Panel. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20, pp. 483–484. Association for Computing Machinery, New York, NY, USA (2020). URL https://doi.org/10.1145/3328778.3366979
43. Simon: Emergence of computing education as a research discipline. Ph.D. thesis, Aalto University School of Science (2015)
44. Sutinen, E.: Koli calling: From the ten past years to the future - a developing country's perspective. In: ACM Koli Calling: Conference Proceedings (2010)
45. Tedre, M.: Methodology education in computing: Towards a congruent design approach. In: Proceeding of the 44th ACM technical symposium on Computer science education, pp. 159–164 (2013)
46. Tedre, M., Apiola, M., Cronjé, J.: Towards a systemic view of educational technology in developing regions. In: Proceedings of IEEE Africon 2011 Conference (2011)
47. Tedre, M., Apiola, M., Oroma, J.O.: Developing it education in tanzania: Empowering students. In: 2011 Frontiers in Education Conference (FIE), pp. T3E–1–T3E–6 (2011)
48. Tedre, M., Bangu, N., Nyagava, S.I.: Contextualized IT education in Tanzania: Beyond standard IT curricula. Journal of Information Technology Education: Research **8**(1), 101–124 (2009)
49. Tedre, M., Brash, D., Männikkö-Barbutiu, S., Cronjé, J.: Towards identification and classification of core and threshold concepts in methodology education in computing. In: Proceedings of the 2014 conference on Innovation & technology in computer science education, pp. 237–242 (2014)
50. Tedre, M., Simon, Malmi, L.: Changing aims of computing education: a historical survey. Computer Science Education **28**(2), 158–186 (2018). URL https://doi.org/10.1080/08993408.2018.1486624
51. Tshukudu, E., Sentance, S., Adelakun-Adeyemo, O., Nyaringita, B., Quille, K., Zhong, Z.: Investigating k-12 computing education in four african countries (Botswana, Kenya, Nigeria and Uganda). ACM Transactions on Computing Education (TOCE) (2022)
52. Xiaoming, L., Lunt, B.M.: Undergraduate computing education in China: A brief status and perspective. In: Proceedings of the 7th conference on Information technology education, pp. 35–38 (2006)
53. Zaagman, E.: China's computing ambitions. Communications of the ACM **61**(11), 40–41 (2018)
54. Zhang, J., Luxton-Reilly, A., Denny, P., Whalley, J.: Scientific collaboration network analysis for computing education conferences. In: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE '21, pp. 582–588. Association for Computing Machinery, New York, NY, USA (2021). URL https://doi.org/10.1145/3430665.3456385

# Computing Education Research in Finland


**Lauri Malmi, Arto Hellas, Petri Ihantola, Ville Isomöttönen, Ilkka Jormanainen, Terhi Kilamo, Antti Knutas, Ari Korhonen, Mikko-Jussi Laakso, Sonsoles López-Pernas, Timo Poranen, Tapio Salakoski, and Jarkko Suhonen**

## 1 Introduction

In this chapter, we present first an overview of the educational system and computer science education in Finland, followed by scientometric analysis of CER publications with Finnish authors. Thereafter we present briefly work carried out in Finnish research groups and finally reflect on the factors behind the intensive work in CER in Finland.

L. Malmi (✉) · A. Hellas · A. Korhonen
Aalto University, Espoo, Finland
e-mail: Lauri.Malmi@aalto.fi; Arto.Hellas@aalto.fi; Ari.Korhonen@aalto.fi

P. Ihantola
University of Helsinki, Helsinki, Finland
e-mail: petri.ihantola@helsinki.fi

V. Isomöttönen
University of Jyväskylä, Jyväskylä, Finland
e-mail: ville.isomottonen@jyu.fi

I. Jormanainen · J. Suhonen · S. López-Pernas
University of Eastern Finland, Joensuu, Finland
e-mail: ilkka.jormanainen@uef.fi; sonsoles.lopez@uef.fi; jarkko.suhonen@uef.fi

T. Kilamo · T. Poranen
Tampere University, Tampere, Finland
e-mail: terhi.kilamo@tuni.fi; timo.poranen@tuni.fi

A. Knutas
LUT University, Lappeenranta, Finland
e-mail: antti.knutas@lut.fi

M.-J. Laakso · T. Salakoski
University of Turku, Turku, Finland
e-mail: milaak@utu.fi; tapio.salakoski@utu.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_16

## *1.1 Finnish Educational System*

Finland is located in northern Europe and has a population of 5.5 million people. It is a member of the European Union (EU) and associated with Nordic countries together with Sweden, Norway, Denmark and Iceland.

Finnish educational system includes pre-school education for 6-years old children and comprehensive school (grades 1–9), followed by either high school (grades 10–12) or vocational education. Currently, students are requested to continue their studies in secondary education after comprehensive school until 18 years of age and the goal is that everyone should get either a high school or vocational school degree. Tertiary education covers two branches. Research universities provide Bachelor's, Master's and Doctoral education, with target studying time (3 + 2 + 4 years) correspondingly. Universities of applied sciences provide more practically oriented degrees in a large number of different professions (4 years) which roughly correspond to Bachelor's level degrees in research universities.

All teacher education is given in research universities. Pre-school teachers must have at least a Bachelor's degree in educational sciences, while primary school teachers responsible for a class of pupils must have at least a Master's degree in educational sciences. Subject teachers in primary school (grades 1–6), i.e., teachers responsible for teaching a particular subject such as mathematics or arts, must have at least 60 ECTS[1] worth of studies on the subject they are teaching. In lower secondary school (grades 7–9), subject teachers are required to have a master's degree (not necessarily from educational sciences) including at least 120 ECTS worth of studies in their main teaching subject and at least 60 ECTS worth of studies from other subjects they teach. They also must have completed at least 60 ECTS worth of pedagogical studies in their teacher education specialization track. Similar pedagogical studies are also required for teachers in universities of applied sciences but not in research universities, where requirements for pedagogical studies vary but typically are much smaller.

Students are admitted to tertiary education based on their national level matriculation exam results (high school track) or vocational degree or based on a field-specific entrance examination or a combination of the previous ones. Students from universities of applied sciences can continue to master's level studies in research universities within a competitive admission process.

Bachelor level education in universities is widely given in Finnish and partially in Swedish, the other official language of Finland. There are few bachelor level programs where education is given fully in English and they are targeted to international students and immigrants with no sufficient command of Finnish or Swedish. On the other hand, on master's level education, programs provided in English are much more common. Many universities of applied sciences also provide targeted programs in English to recruit good international students.

[1] European Credit Transfer System. One ECTS means roughly 26 h of work.

One leading principle in the Finnish educational system has been that education is free. There are no tuition fees. Only quite recently international students coming from non-EU countries have been requested to pay tuition fees. Generally, the programs also provide scholarships options to waive the fees partially or even wholly.

## *1.2 Computer Science Education in Finland*

Computer science education in Finnish universities began in the 1960s when the first professorships were established. Currently, computer science and/or information systems programs are available in almost all universities. While learning programming has been a natural part of computer science programs, programming courses have also been widely taught for CS minors and as service courses. As a consequence, for a very long time, teachers have faced the challenge that the introductory programming courses are large ranging from several dozens of students to courses with 1000+ students. While the course sizes naturally vary among the universities, a common challenge has been the very limited number of faculty members as teachers. The main approaches to address this challenge have been using large numbers of teaching assistants, mostly BSc level students, to instruct younger students, and building in-house tools to support programming education, or adopting such software from other universities. Commercial solutions from companies either in Finland or elsewhere have been used on a very limited scale.

Development of in-house learning tools has generally been initiated by active teachers of large courses either as their own work, based on student projects or funded by small educational development grants provided by computer science departments or universities. This development work started actively in University of Helsinki, Helsinki University of Technology and University of Joensuu in the 1990s and a few years later in several other universities independently from each other. The tools were tailored for addressing local educational challenges in basic programming and data structures and algorithms courses. The corresponding pedagogical reforms were carried out from the same perspective.

This extensive effort in developing education was the seed for initiating research in several sites. Already in the 1990s, the first experience reports were published in educational development conferences organized in Finland, such as Hypermedia in Vaasa 1993 and 1994. The international perspective was adopted by the pioneers when ACM Innovations and Technology in Computer Science Education Conference (ITiCSE) was organized in Uppsala, Sweden in 1997 and in Helsinki in 2000 (chaired by prof. Jorma Tarhio). Moreover, the first Program Visualization Workshop was organized in 2000 by professor Erkki Sutinen in Porvoo. The pioneering professors launched the Koli Calling conference in 2001 as a swap meeting for Finnish computer science teachers. A few years later, the conference took steps towards an international research conference.

Concerning education in the K-12 level, however, Finland has not been among the pioneers globally, and computer science has never been an independent school subject. Information and Communication Technology (ICT) skills was introduced as a voluntary subject to the 8th and 9th grade of comprehensive school in the late 1980s. Pupils who chose the subject –depending on the teacher– also had the opportunity to learn programming (e.g. with Pascal). In the early 1990s, ICT was to be integrated in other subjects, which was further emphasized by the Ministry of Education in the early 2000s. This effort did not fully succeed [18]. However, due to the relative freedom of school teachers to organize additional voluntary courses, some pupils had the opportunity to learn programming despite the integration efforts [75]. Only in 2016, the school curriculum was finally revised to include computational thinking and basics of programming. These are most often implemented in the context of mathematics education.

Computer science teacher training has been organized at many universities as part of their teacher training programs. However, this sub area has not been very popular due to the availability of very few teacher positions in schools in computer science. Therefore, some programs have even been discontinued.

## 2 Finnish CER Community: Scientometric Analysis

In this section, we present findings from our scientometric analysis of CER in Finland. A subset of the dataset described in chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" of this book [92] (containing CER worldwide) has been created by including only those papers in which at least one of the authors had a Finnish affiliation at the moment of publication. We have interpreted the affiliations as Finnish if the author has self-given an affiliation that matches a Finnish university or other institute. Thus, Finnish authors visiting foreign institutes and using their affiliation there are excluded. Correspondingly, foreign visitors in Finnish universities are excluded if they are using their home affiliation. In some cases, authors have not given a clearcut affiliation. These are excluded. The total number of articles included for analysis is 535.

We must also emphasize, as explained in chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research", that the metadata collection of CER articles was performed in Scopus. Scopus does not include all publication years of venues where CER papers are regularly published. Moreover, the keyword search used for finding CER papers in other publication venues is limited to the keywords we used. Therefore, it is understandable that the total publication and citation counts of specific authors are lower than what one could find, for example, from their Google Scholar page. We, however, believe that our data from Scopus corresponds well enough to theoretically complete data, if such were available, because we report

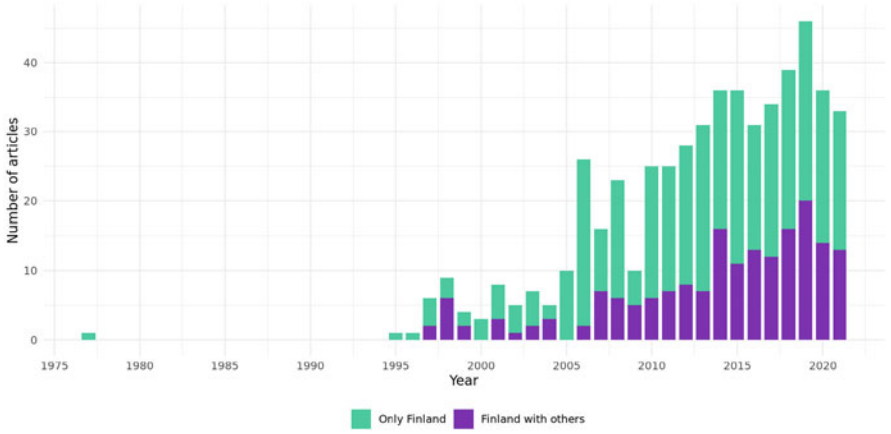


**Fig. 1** Evolution of publications with Finnish affiliation. Green color indicates papers with Finnish affiliation only, and cyan color indicates papers with also international authors

mainly data concerning the most active researchers. They have been working in CER for many years, and their collaboration networks have evolved over the years.

Figure 1 presents the growth of the number of papers with Finnish affiliation in our data pool. The oldest paper [49] from 1977 discusses education from a systems' approach, presenting it as a data communication process. Thereafter, there is a long pause and only in the late 1990s a continuous stream of papers begins to appear with rapid growth reaching the level of 30–40 papers annually around 2015. It is notable that international collaboration emerged very early, and it still has a very strong role. In most years, roughly 25–40% of papers also have international authors. Note that the data pool also includes papers where the main work, including data collection and analysis, has been carried out outside of Finland, but some Finnish researchers have been participating in them as co-authors.

The ten most productive authors and their publication history are shown in Fig. 2. From those in the list, pioneers in the field are Erkki Sutinen (published since 1997), Lauri Malmi, Ari Korhonen (since 2000) and Jarkko Suhonen (since 2001). Only Malmi and Sutinen in the list are professors whose own PhD was from another area of computer science. All others are PhD graduates in CER. In total, at least 50 PhD theses have been completed in Finnish research teams during the last 20 years.[2]

Finnish authors have built their own collaboration networks, which are shown in Fig. 3. The size of the circle indicates the total publication activity and the width of connecting lines indicates the number of joint publications between the authors calculated using fractional counting. In fractional counting, instead of

[2] Many more have been completed in the groups in other, closely related areas, such as engineering education research, educational technology or ICT4D. We counted only those ones in CER. We, however, acknowledge that the borderline of what is included in CER or not is not always obvious.

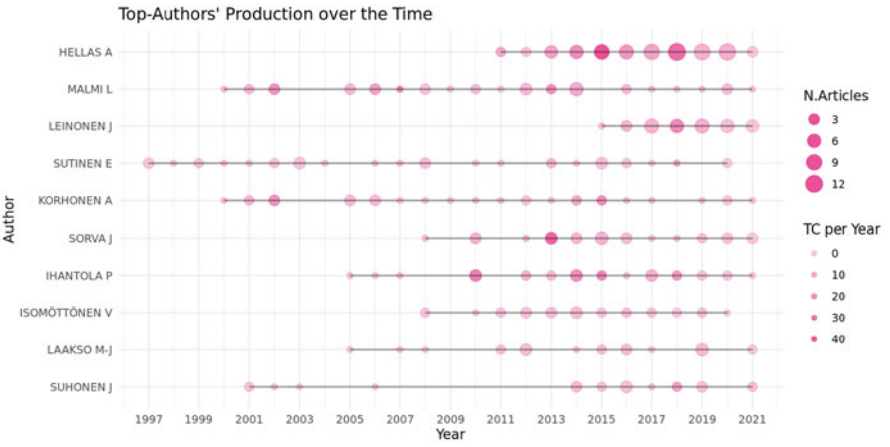


**Fig. 2** Most productive authors with Finnish affiliation. The sizes of circles indicate the number of papers published by the author in a specific year. The color indicates the citations for the authors in a specific year. Data is from Scopus

counting each publication as "one" between each pair of co-authors, the count is divided by the number of co-authors in the paper (see chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" for a more detailed explanation). Many of the stronger links reflect the supervisor-PhD student/postdoc relation, but this is no general rule. Colors indicate communities who have done more work together in terms of joint papers. Note that the communities are identified by an algorithm, and they are not disjoint; thus people can be a part of several communities—coloring cannot fully visualize this.

For example, Hellas and Leinonen have had very strong collaboration, as well as Malmi and Korhonen, Korhonen and Karavirta, Sutinen and Suhonen, Laakso and Apiola. Due to overlapping edges and nodes in the graph layout, some collaboration is not visible, or might give a somewhat misleading image. For example, strong collaboration between Ihantola and Karavirta is partially hidden behind the edge between Karavirta and Korhonen. Moreover, Sorva and Sirkiä have much collaboration, but this is not connected with Sheard.

On the top, in pink and orange, we can see the University of Eastern Finland (UEF) team with Sutinen, Suhonen, Tedre, Jormanainen, Toivonen and Oyrlele as the key people. On the top right, there is the University of Turku (UTU) team with Laakso, Apiola and Salakoski as the main people. On the other hand, Sutinen has moved from UEF to UTU building new collaborations there. In the center, there is the Aalto University team in gray and pink with Malmi, Sorva, Korhonen and Kinnunen (as main Finnish authors). Below this, there is the wide circle of Hellas who has a very large network with Leinonen, Luukkainen, and Ihantola forming the core of researchers at University of Helsinki. Low left in yellow is Lappeenranta University of Technology group (Knutas, Ikonen, Kasurinen, et al.) and on the left,

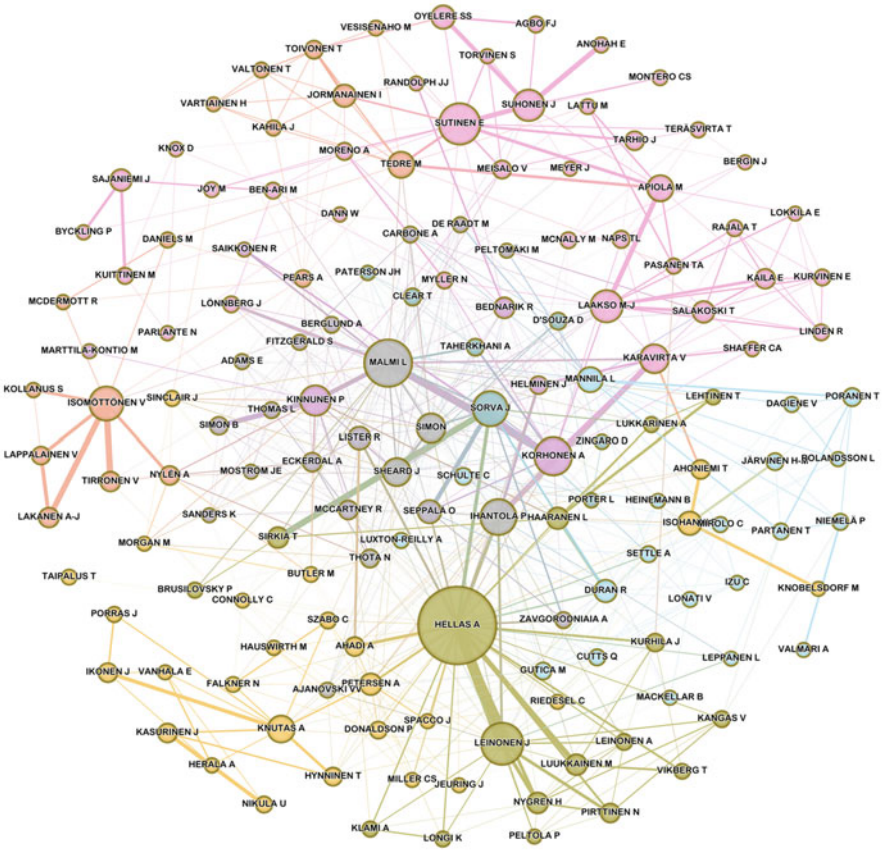


**Fig. 3** Collaboration network of authors with Finnish affiliation

in light orange, there is University of Jyväskylä team with Isomöttönen as the core person.

There is a large number of foreign authors in the network, which partially confuses the picture, but at the same time demonstrates the international collaboration network among Finnish authors. It is also natural that the communities evolve, as some people change their affiliation. For example, Sutinen worked a long time in UEF and then continued his career in UTU; Ihantola has worked at Aalto University, University of Tampere and finally at University of Helsinki; Kinnunen has moved from Aalto University to University of Helsinki, and Hellas has moved from University of Helsinki to Aalto.

When considering the most popular publication venues among authors with Finnish affiliation, the two clear top venues are Koli Calling and ITiCSE, followed by ICER and SIGCSE. Table 1 presents the ten most popular venues. In total, papers

**Table 1** Most popular publication venues among Finnish authors (in Scopus)

| Venue | Papers |
|---|---|
| Koli Calling International Conference on Computing Education Research | 158 |
| Innovation and Technology in Computer Science Education, ITiCSE | 124 |
| International Conference on Computing Education Research, ICER | 36 |
| ACM Technical Symposium on Computer Science Education, SIGCSE | 32 |
| Computer Science Education | 20 |
| ACM Transactions On Computing Education | 16 |
| Frontiers in Education Conference, FIE | 14 |
| ACM SIGCSE Bulletin | 11 |
| Australasian Computing Education Conference, ACE | 7 |
| International Conference on Computer Supported Education, CSEDU | 7 |

had been published in 90 different conferences and journals, including 59 venues with only a single paper.

Despite the fact that Koli Calling is always organized in Finland, it is a very international conference with participants and submissions coming from numerous countries globally. We discuss its history and character more below.

## 3 Koli Calling Conference

One of the landmarks in Finnish CER was launching the Koli Calling conference in 2001, with professors Erkki Sutinen and Tapio Salakoski being the initiators. Lauri Malmi soon joined the team. They were conference chairs for the first 5 years.

Koli is a high hill in Eastern Finland within a national park with a wonderful view to Lake Pielinen. The initial name of the conference in Finnish was Kolin Kolistelut, where the latter word means a rattling noise. The selected name indicated that the purpose was to shake existing practices of teaching computing and to invent something new. Indeed, for the first 3 years, the conference was a swap meeting for Finnish computer science teachers and only few foreign people attended it. Even the language of discussion changed between Finnish and English depending on whether foreign people were present in the session or not.

In 2004, the program committee agreed that the conference should take a different profile seeking to solicit research papers internationally. The program committee was extended with more international scholars, and the call for papers revised to solicit papers on two tracks: Research papers and Discussion papers. The latter were shorter and targeted to present novel educational innovations for the conference audience. In the following years, the call for papers was further elaborated to better respect the richness of work carried out in the field. Thus, new submission types were added, including system papers for describing novel educational software tools and theoretical papers for theoretical discourse. Moreover, in some years, a separate

Call for Tools was published with the idea that the submission should also include relevant software which could be evaluated, too, and not just the paper describing it. All these activities reflect the nature of the conference as a versatile venue for presenting research and discussing new developments. Over the years, a large share of Koli Calling participants have been PhD students who have presented their early work first as posters, demonstrations or discussion papers and later on presented solid research papers at Koli.

One of the basic characteristics of Koli Calling has been its location, in the middle of a national park. Staying in an isolated hotel and the small size of the conference (around 50 participants) has created excellent opportunities for networking. The conference begins with Thursday evening dinner, followed by two full days of presentations and discussions, with typically the closing session on Sunday morning. On Saturday afternoon, there is a break and the Koli Walk for visiting the national park (there is often snow on ground which is quite spectacular for many foreign visitors). In the evenings, there is an opportunity for attending the Koli sauna session or visiting Koli Spa. All these activities provide ample opportunities for meeting colleagues informally. Moreover, as nobody leaves for visiting elsewhere for restaurants and sightseeing, it is practically possible to discuss with everyone during the conference.

Koli has gained a reputation of one of the leading conferences in CER, among SIGCSE, ITiCSE, ICER, and ACE. While the share of Finnish participants has naturally always been large, the majority of participants are international, especially from Europe. There are, however, frequently many participants from Australasia, and increasingly also from the US. During the pandemic in 2020 and 2021, the conference was organized only virtually, which extended its size to 100 participants, many from the US.

For more information about the conference and a scientometric analysis of its publications, see [7].

## 4 CER in Finnish Universities

In this section, we present the development and main focus areas in the major research groups in Finnish universities.

### *4.1 Aalto University*

The roots of CER at Aalto University originate from the educational development activities in 1980s and 1990s at Helsinki University of Technology, TKK.[3] The basic programming courses targeted to the whole university were very large ranging from

[3] Aalto University was launched in 2010 as a merger of Helsinki University of Technology, Helsinki School of Economics and University of Art and Design.

a few hundred to over thousand students. The courses had lectures, weekly exercises, programming projects and an exam. The weekly exercises were not graded; their model solutions were presented in large exercise groups in lecture halls, where teaching assistants also gave some guidance. One or two programming projects per course were submitted for manual grading. Exams were on paper only. In addition to such traditional teaching methods, online guidance for projects was widely used already in the 1980s, implemented with course-specific Unix newsgroups, where students could ask questions.

Lauri Malmi started to work as a lecturer in 1986, and soon became interested in improving pedagogical approaches to teaching programming. There was a burning problem: how to manage four large courses annually with roughly 2000 enrolled students with one lecturer and only a small number of BSc level teaching assistants working a few hours a week to give guidance and grade projects. While new learning resources and new pedagogical approaches were developed, grading and giving personal feedback on weekly exercises turned out to be infeasible with these human resources. Unfortunately, recruiting more teaching personnel was not an option either. Hence, the solution was to build and use software to support education.

The first educational technology project was launched as a capstone project in 1990. A student team implemented the tool called TRAKLA that automated the assessment of *algorithm simulation exercises* [39, 68] on a data structures and algorithms course. Students received the assignments and submitted their solutions in a predefined text format by email, which TRAKLA server checked. In these partially compulsory exercises, students presented in high level of abstraction how a given algorithm and a set of operations change a given data structure.

Launching the tool in spring 1991 reduced course grading workload hugely. Moreover, the final exam results also improved. Encouraged with this, a paper presenting the system was submitted to HyperMedia in Vaasa conference in 1993 [39]. In this conference, Malmi met Edmund Burke and learned about the Ceilidh tool for automatic assessment of programming submissions [16] that had been developed at University of Nottingham, UK. Based on Malmi's recommendation, the tool was adopted in the basic programming course at TKK. Ceilidh was used the first time in 1994 and made a huge change in the course. Now, it was possible to set up weekly compulsory exercises which were graded automatically, and teaching assistants' work could then be directed much more into giving guidance, instead of grading. Moreover, students could resubmit their solutions after getting feedback.

Finnish Ministry of Education launched in the mid 1990s a program to support the quality of university education. National Centers of Excellence in Education were selected every third year based on competitive applications. A team of highly devoted junior teachers and researchers who had convened regularly to discuss how CS education could be improved managed to prepare successful applications to these calls. The department's basic education section gained the national level status for 2001-2003, and this was revised for the second period 2004–2006, and after a compulsory hiatus for the next round the whole department received the status again for 2010–2012. This provided substantial funding for the team. Moreover, the

funding from the Ministry was not typical project funding tied to the project plan goals, but it was more like an award to freely improve education further.

Computer Science Education Research Group, COMPSER, was formed in 2000. In addition, Malmi was promoted to associate professor in 2001, which increased the academic independence of the group. Ari Korhonen was his first PhD student, who had already started developing TRAKLA further in his MSc thesis a few years earlier. His PhD research, completed in 2003, focused on development and evaluation of *visual algorithm simulation exercises* in TRAKLA2 [68]. Päivi Kinnunen started her PhD studies first by investigating problem-based learning in programming education and thereafter CS1 students' dropout problem [60, 61]. Several talented MSc students, who have later on gained substantial visibility in the CER field, Juha Sorva, Otto Seppälä, Petri Ihantola, and Ville Karavirta joined the team in early 2000s. They had been working earlier as teaching assistants on programming courses or summer trainees and started to work with various new software projects, and soon were involved in writing papers already when studying for their Master's degree or doing their MSc thesis. They all continued for doctoral studies after completing their master's thesis, resulting in many doctoral theses a few years later.

The same model of recruiting talented students early in master's level studies or at the latest when starting the MSc thesis project has continued and turned out to be a very successful practice, resulting in a large number of doctoral theses. The main research areas in the theses have focused on program and algorithm visualization [53, 90, 108, 125, 128], automatic assessment [33, 40, 124, 135] and games and gamification [9, 26, 29].

As part of the research, multiple software tools were developed, including several versions of the TRAKLA concept [39, 69, 96, 109], teacher's algorithm simulation tool MatrixPro [55], visualization tool for concurrent programs, Atropos [91], program simulation tool UUhistle [129], Parsons problem framework jsParsons [41], ACOS content server [127], JSVee and Kelmu visualization tools [126], and Rubyric manual grading support tool [8]. Most of them have been used for several years in large programming courses, which has enabled collecting and analyzing lots of data of their impact on students' learning results and studying process, as well as their understanding of programming concepts. Naturally, many of these software are now outdated due to being implemented in dated technologies, or as a natural result of course development when they are not needed any more. On the other hand, some tools have persisted in use. TRAKLA2 exercises have been re-implemented with Javascript library jSav [56], and the A+-learning environment [54] has been in continuous use at the department since 2013 and is now used in dozens of courses. It has also been adopted at University of Tampere. Rubyric is still being used, after 10 years, to support manual grading of project reports and submissions.

COMPSER changed its name to the Learning + Technology research group, LeTech, as some research activities extended to more general education technologies, and engineering education research. Current research themes in the group cover teaching/learning event-driven programming, automatic generation of questions from students' programs, students' misconceptions on algorithms,

motivational factors in affecting learning programming, interactive tutoring for debugging, learning analytics, as well as automatic assessment in mathematics and tools for supporting learning to write academic English.

COMPSER/LeTech members have been very active in national collaboration. Several Koli Calling program chairs have had their background in COMPSER/LeTech. Malmi and Korhonen have also coordinated important national networking projects, which are discussed more below.

International collaboration started early after being inspired by participating in ITiCSE conferences in Uppsala 1997 and Helsinki 2000. Malmi and Korhonen participated in ITiCSE working groups focusing on evaluation of the impact on algorithm visualization in 2002 and 2003, and thus built valuable contacts with international researchers working in this area. Much of this continued also in active participation of Program Visualization Workshops, a series of biannual small international workshops organized in 2000-tale, initiated by prof. Sutinen at University of Joensuu in 2000. Very many of COMPSER/LeTech members have participated in ITiCSE working groups thereafter, which has supported their own international networking. Often the working group reports were finally included among their doctoral thesis publications.

## *4.2 University of Helsinki*

The department of Computer Science at the University of Helsinki (UH) has a long tradition in developing and utilizing educational technologies and practices to improve teaching, as well as in evaluating tools and practices developed by others. This tradition has been mostly grassroot level activity, driven by individual teachers and professors. Teachers at UH—in addition to developing tools and teaching—often studied the effect of these tools and teaching on students' learning. Results of these studies and experiments have typically been shared as reports and presentations at department-level teaching days or at university-level events. Teaching has also been valued, evidenced both through teaching-related annual awards both from the department and university levels, as well as through funding based on gaining the status of a center of excellence in education from the Ministry of Education in 2001–2003.

CER has been acknowledged at the department at least since the early 1990s. However, despite the fact that the fifth ITiCSE conference was organized at UH in 2000, presenting the work at CER venues was relatively rare at the beginning, when contrasted with the amount of work that took place at the department. Such work was often published elsewhere. For example, Eliot and Jeliot systems that piloted pedagogical algorithm animations were published at conferences related to Computer Graphics and Visualization [80] and Visual Languages [25]. The similar observation holds e.g. for intelligent tutoring systems, intelligent learning materials, and systems with social navigation support [72, 73]. The emergence of the Koli

Calling conference provided a home for some of the work in the early 2000s [58, 59] and only later on CER papers were published more in classic CER venues.

The first doctoral dissertation in CER from the Department of Computer Science (2003) was "Considering Individual Differences in Computer-Supported Special and Elementary Education" [71] that focused on how interactive learning environments can adapt to special needs. It has been followed by a handful of theses focusing on aspects such as supporting creativity in teaching programming [6], pedagogies and tools for teaching programming [32], and data analytics from programming environments [85]. Beyond the theses, CER researchers have studied approaches to teaching programming [74, 148] and good software engineering practices [70, 94]. As a part of this work, researchers have also looked into approaches to increase student engagement and support peer learning [150] as well as on building automated assessment approaches that provide stepwise help to students learning programming [149]. While the previous examples focus mostly on bachelor's level education, researchers have also looked into supporting students' working on capstone projects and beyond [23, 95].

This work has also led to building open online courses in programming [75], which in turn has led to studies on using open online courses as a way to recruit students into computer science studies [87]. Furthering this work, the Department of Computer Science has also created a MOOC-platform[4] that currently has millions of users from across the world. Moreover, developing the tools and platforms for supporting online learning has enabled automatic data collection and data-driven approaches to investigate the learning processes [149].

In particular, CER researchers have looked into approaches to identifying at-risk students and their challenges [2, 88], understanding students' help-seeking strategies [103], studying how learners use online materials and whether adjustments to contents such as images or progress visualizations helps learners [27, 42], understanding how code is written and who is writing it [86, 89], understanding characteristics of students [21, 116], and more broadly modeling students' learning [37]; this, in a sense, links back to the intelligent tutoring system-related studies conducted at the department in the late 1990s and early 2000s [72, 73].

The good track record in research driven development of learning software was recognized also at the university level when the University of Helsinki MOOC center was established late 2020 to carry out research around online learning and to extend the technology and related best practices built around computing education to other disciplines. The new center was positioned in the CS department and the head of the unit, professor Petri Ihantola, was selected from the faculty of Educational Sciences, where computing education is also developed. As an example, the faculty hosts the Innokas Network[5] with focus on K-12 education and teacher education.

[4] https://www.mooc.fi.

[5] https://www.innokas.fi/en.

### *4.3 University of Jyväskylä*

University of Jyväskylä (JYU) has a long history in developing tools to support teaching and learning programming, especially by lecturer Vesa Lappalainen; however, little of this work has been published. JYU has also over a two-decade history of educating computer science subject teachers. Both these traditions contributed to a situation that a door was open for CER. Professor Kärkkäinen, who was in charge of subject teacher education, supervised two education-related dissertations [35, 45], of which Isomöttönen's thesis [45] was in CER. During this time, CER-related dialog started to grow in the faculty.

The background of the group can also be said to be based on accidents. Ville Isomöttönen who now leads the group started his PhD work with a computer science music topic. However, he changed the topic into project-based learning in software engineering after receiving an acceptance on a project-course themed conference paper. The change resulted from a collaboration invitation by a colleague Sami Kollanus to work within CER which was his side topic. The key point of this turn was the first publications in Koli Calling, ITiCSE, and CSEE&T conferences in 2008.

Over the years, the pioneers persuaded others to attend, which has led to research in multiple topics and completing several dissertations. The CER group has studied functional programming education that emphasizes students' self-direction [48, 140], interest development in programming education for K-12 outreach activities [82], as well as multidisciplinarity and students' view of industry collaboration [31]. More recently, a dissertation was completed on the topic of SQL education [136].

Examples of recent research themes on programming education include motivation, identity, creativity, and interest development during programming courses, as well as exams as a learning experience. Project-based learning is studied from the perspectives of reflective learning [112], justice [47], and status processes. Research on database education has continued [137], whereas a more general theme has addressed study difficulties and related interventions among CS students [36]. Many other themes (e.g., developing theoretical frameworks to explain the challenges of teaching a particular area, flexible delivery, infographics for reflective learning, and multi-purpose educational technology) have been recently addressed when attempting to introduce new persons to the group or to initialize shared research topics. The group has slightly emphasized qualitative approaches in research.

On the side of the research, programming-related course teachers have developed and taken into full use several novel software products, e.g., a unit testing tool that can be effortlessly integrated into introductory programming materials [83], an automatic assessment tool of ICT skills [81], and a tool for learning Haskell. The current prominent example is the TIM (The interactive Material) teaching and learning platform, which integrates a high number of functionalities—all that teachers need—into a single learning management system [46, 141]. The system

has served also in wider contexts, for example, to support national level university entrance examinations.

JYU group has built international collaboration with UpCERG group in Uppsala University, Sweden, resulting in multiple joint research articles (e.g., [46]) and visits. A major starting point for this collaboration was discussions (e.g., between Anders Berglund and Isomöttönen) during Koli Calling conferences after which collaborations at personal levels ensued. After a couple of less active Covid years, this collaboration was recently revitalized for more project-based learning studies in which critical incident technique (CIT) provides a framework for exploring reflective learning. Additionally, the group is currently collaborating with Eindhoven and Leiden universities on database education.

The CER group at JYU is currently an acknowledged research group of IT faculty, while not yet in the position of main research divisions. Thus, JYU has not initiated a professorship in CER. Doctoral theses in the group are now supervised based on docentships of the senior researchers in the group. Finally, it is worth noting that educational technology and subject teacher education lines also conduct important educational research in the faculty. However, they are geared towards other publication forums outside CER.

### *4.4 University of Joensuu/University of Eastern Finland*

The first research in the CER field at the Department of Computer Science, University of Joensuu[6] can be traced back to the end of the 1990s. Prof. Martti Penttonen supervised the doctoral dissertation of Marja Kopponen, titled "CAI in CS", in 1997, the first CER dissertation from the department [67]. Dr. Kopponen continued to publish work in CER together with Prof. Jorma Sajaniemi mainly regarding computer-aided lecturing technologies [120]. At the beginning of the 2000s, Sajaniemi's research group focused on cognitive science aspects of computing education, especially on the *roles of variables* and program animation in computing education [17, 24, 110, 121, 122]. The group also worked on eye tracking research [104], which later on extended beyond computing education to the medical field, mainly by research and development work of Associate Professor Roman Bednarik.

Research in CER was expanded when Prof. Erkki Sutinen joined the department in the late 1990s, and he was responsible for coordinating the computer science teacher education studies. He also formed the edTech research group and started several new CER initiatives. The first of these was ViSCoS (Virtual Studies of Computer Science) online studies, which offered university-level computing studies to high school students in the North-Karelia region in Finland [28, 133]. The design

[6] Later on the Department of Computer Science and Statistics at the University of Joensuu (2006-2010) and the School of Computing at the University of Eastern Finland (2010-current).

aspects of ViSCoS studies were the focus of the first doctoral dissertation in the edTech research group, by Jarkko Suhonen [132].

The second initiative was Kids' Club—a technology-rich after-school club environment. The club environment accelerated the group's research, specifically on educational robotics and programming education. The Kids' Club also participated in international robotics competitions, especially RoboCupJunior, with good success. Club activities sparked ideas for in-service teacher training with educational robotics and other state-of-art technologies, and gained support from two externally funded projects (2003–2007, European Social Fund) focusing on the development of technology and computing education at schools. The projects, RoboCupJunior activities in Finland, and in-service teacher training were some of the building blocks for a Finnish network of school teachers, later known as the Innokas network. This network also influenced on the Finnish curriculum reform for primary and secondary schools by defining what "Computing at Schools" could be in Finland. The club environment formed a basis for two doctoral dissertations [50, 151]. The third CER research initiative was contextualized computing education [57, 146]. The edTech group had many years of intensive collaboration with the Tumaini University, Tanzania to implement a locally relevant bachelor's study program in Information Technology [147]. Moreover, several individual doctoral students' research topics have been connected to contextual computing education, for example, [1, 66, 101, 113].

The fourth significant line of CER research was related to the Jeliot program visualization tool, which was originally developed at the University of Helsinki. New features and related research focused especially on collaborative visualization and conflictive program animation [14, 99, 102]. The development of Jeliot continued till the end of 2000s, until it started fading after a more than a decade of work [15]. However, thereafter it has still been a part of individual doctoral students' research work [30].

The size and impact of the edTech research group increased considerably after the mid-2000s when IMPDET[7] online doctoral studies, a joint initiative between the edTech research group and education researchers [134] was launched and gained an important role in the group's activities. While the research topics of IMPDET students have been diverse and mainly related to educational technology and ICT for development, there have also been doctoral dissertation topics connected to CER, for example, in improving assessment processes of information system studies [10].

A significant change in the edTech research group happened when Prof. Sutinen left to the University of Turku in 2015, which forced the group to renew its operations. The senior researchers, Jarkko Suhonen, Ilkka Jormanainen, and Calkin Suero Montero started to supervise doctoral students and apply for external funding independently. Prof. Markku Tukiainen took over Sutinen's existing research projects and initiatives, including IMPDET studies with almost 50 enrolled doctoral students. Prof. Matti Tedre and Dr. Mohammed Saqr joined the research group

[7] https://www.impdet.org.

in 2017 from Stockholm University, which strengthened the group's activities considerably.

The changes in the core personnel brought new research focus areas, such as maker pedagogy [142], learning analytics [93], VR/AR in computing education [1], computational thinking education [51, 84] and machine learning/artificial intelligence education [100, 138, 139]. The new research topics also intertwined with earlier research, such as exploring teachers' preconceptions of teaching machine learning in the African context [123].[8] Finally, collaboration with education researchers inside the University of Eastern Finland was re-established, especially related to machine learning/artificial intelligence education and computational thinking education [144].

The edTech research group has not been focusing purely on CER topics, but the group's research interests have been very diverse, including educational technology, ICT for development, natural language processing, business informatics, and text analysis methods. New research topics have emerged, for example, when new doctoral students and faculty members joined the group's activities. Specifically, Prof. Sutinen expanded the group's research work into new areas, instead of focusing on one or even few narrowly specified research topics. The group's openness to accept a wide range of topics also seemed to attract new people with varying backgrounds to join the group. Moreover, the wide spectrum of research interests enabled acquiring funding from various sources.

The two groups, Prof. Sajaniemi's group, and edTech have had quite different methodological profiles. The former employed mainly empirical-quantitative research approaches, while the edTech group has been using a diverse mix of research approaches, quantitative, qualitative, action research, design science research and many others. Besides pure research interests, the work at edTech has also been motivated by creating completely new study opportunities for computing education for different target groups (examples: ViSCoS, contextual bachelor's degree studies in Tanzania, Kids' Club and its spin-offs, maker movement pedagogy and robotics [130, 131]).

The diversity of research topics and approaches also have some drawbacks. Research topics of CER doctoral students have been sometimes too separated, which has led to inefficient use of available resources, and in some cases, the research efforts have not deepened beyond "proof-of-concept" type of research. Moreover, collaborative work between the two research groups could have been stronger, especially on program visualization. However, the CER research at the University of Eastern Finland is currently very active and, for example, new doctoral students with CER interests are joining the group constantly.

[8] The name of the group was also changed from edTech to Technologies for Learning and Development.

## *4.5 Tampere University*

Tampere University was created in 2019 when University of Tampere and Tampere University of Technology were merged. As their groups have a long independent history, we present them separately.

### 4.5.1 University of Tampere

The first Scandinavian computer science professorship was established in 1965 at University of Tampere [114]. This was also the start of university level computer science education in Finland. Professor Reino Kurki-Suonio was nominated to this position, and in 1980, he moved to the Tampere University of Technology. Although programming was already part of the first curriculum, it took four decades before computational thinking related research started. The seeds were sown in 1990s when activities related to the International Olympiad in Informatics (IOI) began.

Informatics Olympiad is one of the international science olympiads, such as International Olympiad in Mathematics, Chemistry and Physics. In IOI, high school students solve programming tasks that require exceptional algorithmic thinking skills. The Finnish team participated in the IOI the first time in 1992 [117] and in 1998, Finland also started participating in Baltic Olympiad in Informatics. During the first 3 years, leading the team was the responsibility of the University of Helsinki, after which it was circulated to the University of Tampere for the next 4 years. Since then, the team lead has been circulated between Universities of Helsinki, Turku, and Tampere. The University of Tampere organized the IOI contest in 2001; there were 272 contestants from 74 countries [111]. Finland's contest success has been relatively good when considering its population. If all medals (gold, silver and bronze) are counted, Finland is currently ranked 28th among all participating countries [43].

During IOI'2008 contest journey, professor Valentina Dagiené from University of Vilnius, Lithuania, proposed that Finland could also organize the Bebras challenge. Bebras is an international initiative aiming to promote Informatics and computational thinking among school students at all ages [13]. University of Tampere started to develop its own contest system, and the first national contest was organized in 2010 with 1472 participants from primary and secondary schools. The contest system has been used also in Sweden and Slovenia. The number of participants has increased since the beginning, and in 2021 it was about 4900. Finland has collaborated actively with Sweden, and this has helped Finland to organize the contest in both official languages, Finnish and Swedish. Currently Finland is using France's contest system, and there are plans to use the ViLLE system developed at University of Turku in 2022 (see below). Bebras challenge produces data on how pupils are solving tasks requiring computational thinking skill, and this data has been used in research with Lithuania and Sweden [19, 20].

Bebras contest has brought many contacts to primary and secondary school teachers, and this has yielded projects resulting in research on programming learning resources and MOOCs to primary and secondary school teachers [107, 115].

Finally, as many other universities, also University of Tampere developed its own learning management system, WEb Teaching Organizer (WETO) in the early 2000s, to help to organise mass courses with peer reviews and automatic assessment [106].

### 4.5.2 Tampere University of Technology

In Tampere University of Technology (TUT), computer science education was initially in the 1970s given under electrical engineering. The first professorship in Computing Systems was established in 1980. Professor Reino Kurki-Suonio from University of Tampere was appointed to this position which he held until his retirement in 2002. The degree program in information technology started in 1985 and finally information technology got its own department in the university in 1993.

Research in CER started gradually from the establishment of the computer science department. The first master's thesis in this area was completed in 1998 [4]. The research and development group for Programming Education, EDGE, led by professor Hannu-Matti Järvinen was established soon after the first thesis and by 2003 an EU project was running in the team.

In addition to learning programming, the main focus area at TUT has been new learning tools and how to best utilize them in computing education. The use of automatic grading, grading feedback, program visualization and peer review have been among the research topics, which has resulted in three doctoral dissertations [3, 5, 44] in addition to one in computational thinking [105]. Notably, Lahtinen et al.'s paper, "A Study Of The Difficulties Of Novice Programmers" [79] has the highest citation count in CER in Finland.

In addition to its own learning technology development, TUT has had active collaboration with Helsinki University of Technology/Aalto University. For example, the rubric-based evaluation tool Aloha, initially developed at Tampere, was further developed at Aalto under the name Rubyric. This collaboration has carried on to the present day, when Tampere University uses and co-develops the learning management system A+ [54] initially developed at Aalto.

When the two universities in Tampere were merged, a challenge emerged: how to harmonize the tool development and usage in the new Tampere University, when both partner universities had their own tools. Luckily at the same time, a national network project, The Intelligent Systems and Content Creation project, was initiated, which helped to resolve these issues (see Sect. 5 for more information).

In 2020, professor Hannu-Matti Järvinen established a new education research group which unites researchers in mathematics education and computing education. Its current main research themes include flipped learning, computational thinking and learning tools.

## *4.6 University of Turku and Åbo Akademi University*

### 4.6.1 The Dawn of CER at the University of Turku

At the University of Turku (UTU), interest and efforts in developing CS teaching were substantially increased when Jorma Boberg and Tapio Salakoski joined the CS department in the mid 1980s. In 1993, Open University CS education was started, calling for a new, multimodal and partially virtual approach taking advantage of modern educational technologies. Very soon it was discovered that contemporary digital pedagogy could not handle increasing numbers of students, and pedagogical research did not focus on scaling up teaching. Challenges dealing with students' difficulties in learning computational thinking and programming posed the first CER questions. The focus remained, however, in developing one's own teaching.

In the late 1990s, Salakoski started as a fixed-term CS professor in bioinformatics. Nevertheless, he maintained his interest also in CER and learned about similar development efforts by Erkki Sutinen at University of Joensuu and by Lauri Malmi at Helsinki University of Technology. The founding of Koli Calling Conference in 2001 by the three professors marked a shift from mere professional development of CS teaching to more serious CER.

This development was accompanied by Salakoski's initiative of setting up CS teacher education at UTU. One of the very first CS Open University students, Mikko-Jussi Laakso, was recruited as a teacher for a new CS course in digital educational technology. A course project work by students Erkki Kaila and Teemu Rajala supervised by Laakso and Salakoski resulted in the first version of ViLLE programming visualization environment in 2004 [119]. The same group later developed a new version of a new more comprehensive ViLLE collaborative education tool in 2010, aimed at supporting teachers facing growing numbers of students with digital tools [78]. The group began to study the impact of technological interventions such as automated assessment and immediate feedback in research settings. In addition to ViLLE, they used the newly developed TRAKLA2 system in collaboration with Malmi and Korhonen at TKK.

At the UTU Faculty of Education, professor Erno Lehtinen was a pioneer in educational technology and technology education. His interest in learning mathematics and computing led him to collaboration with Salakoski already in the 1990s. The collaboration started with jointly supervised MSc theses and has lasted ever since. Even today, they have a joint major 6-year research project Growing Mind funded by the Strategic Research Council at the Academy of Finland.

The first PhD in CER at UTU graduated in 2010, when Laakso received his degree under Salakoski's supervision [77]. Their continued collaboration has resulted in several other PhDs: [52, 76, 145]. The focus of the Salakoski-Laakso group has been the use of automated assessment and immediate personalized feedback in supporting the learning of programming, mathematics, and computational thinking. The ViLLE system has also expanded to a general learning platform used in teaching many subjects, and automated assessment has grown to more

comprehensive learning analytics. Methodology-wise, the role of data and machine learning in analytics has increased. In addition to providing for teachers longitudinal analysis of individual students' learning results, these methods support developing tools for knowledge management and business intelligence for decision makers.

While the majority of Salakoski's scientific work has been in bioinformatics and natural language processing according to the field of his professorship, he has also continued CER work. Another CER-related professorship in interaction design was established in 2016, when Erkki Sutinen moved to Turku. Instead of CER, his main work in Turku has now focused on interactive game design and digital humanities, especially digital theology.

### 4.6.2 The Centre for Learning Analytics

In 2019, Mikko-Jussi Laakso began as an associate professor in learning analytics as the first explicit professorship in CER. He started as the leader of the newly established Centre for Learning Analytics at the Department of Computing. The main efforts were directed towards building a data-based ecosystem for learning and teaching, especially focusing on diagnostics of learning difficulties of mathematical and computational thinking in secondary education, supporting teaching interventions, large scale e-assessment, and knowledge management.

As a result, the ViLLE system has been widely adopted in lower and upper secondary education in Finland, the current penetration being 60% of the schools on a national level. In addition, international collaboration has increased substantially in Europe, USA, Middle East, and Asia. Annually, 500 million ViLLE exercises are being submitted and assessed. The Centre and the ViLLE system have been recognized on several occasions, also globally; they recently received the UNESCO King Hamad Bin Isa Al-Khalifa Prize for the Use of ICT in Education 2021 [143].

The Centre was given the status of an independent unit and moved to the Faculty of Science in 2022. The decision was motivated by the national and international learning loss in mathematical science subjects, along with the decreasing trend in interest of the youth towards university level studies in science and science teacher education. Also the COVID-19 pandemic catalyzed digitalization at all levels of education. The work at the Centre has been funded by the Ministry of Education, Academy of Finland, several foundations and municipalities, as well as the EU. Currently, the Centre has about 30 employees. In 2018, a spin-off company Eduten Ltd was established for the valorization and internationalization of ViLLE technology. Eduten has operations in more than 50 countries. In 2022, it received the UNICEF EdTech Award as the winner of the global Extreme Tech Challenge 2022 competition for EdTech startups.

### 4.6.3 CER at Åbo Akademi University

Åbo Akademi University is another university in Turku giving education in Swedish, the other official language in Finland. There the central people for CER were Ralph-Johan Back, a professor of software engineering (now retired), and Linda Mannila née Grandell, who received her PhD focusing teaching mathematics and programming in 2009 [97]. Working in adjacent premises in Turku, Back and Salakoski created a joint research group with Mannila and Mia Peltomäki, a senior mathematics teacher, CS teaching pioneer, and a PhD student at UTU.

Their group focused on the logical thinking behind both mathematical derivations and programming. The objective was to make mathematical inference visible and explicit; a strict logical formalism enabled applying automated theorem proving for automated verification and immediate individual feedback on students' work in high-school mathematics. They used refinement calculus, a software verification formalism, for describing mathematical inference as structured derivations [11, 12]. They also studied invariant based programming (PhD Johannes Eriksson 2010 [22]) and the difficulties in learning the first programming language [98].

The flagship project of the group was EU-funded E-Math, a collaborative effort in high-school mathematics education with the cities of Turku, Stockholm, and Tallinn in 2007-2013. As a result of the project, a spin-off company Four Ferries Ltd was established, offering interactive math textbooks and other learning material for upper secondary education.

## *4.7 Lappeenranta University of Technology*

Lappeenranta University of Technology (LUT) does not have a specific education research center. Rather, several persons from the faculty contribute to the CER community as a part of their teaching development activities or PhD studies on adjacent topics.

There have been several distinct research themes. LUT participated in some of the pioneering Finnish work on hackathons (later on named Code Camps) since 2008 [118]. These events were instrumental in building industry cooperation around the technology cluster at the campus region. They have added internship opportunities and supported learning practical skills about recent technologies, as well as created opportunities for research collaborations. For example, as an offshoot from the hackathons emerged research into collaborative learning: How could students effectively collaborate in programming teams? To support these efforts, research efforts focused on teamwork analysis [63], content delivery methods [34] and designs (e.g. gamification) to support effective teamwork [65].

A more recent line of research has focused on text mining and analysis of student feedback. This has been divided into two lines of research: Processes and tools for text mining [38], and its impact on, for example, curriculum design [62].

The final line of research from LUT has been software engineering and computer science education networks [64]. LUT has coordinated the Pathways to PhDs in Software Engineering project that mapped and coordinated software engineering education at PhD level. While bachelor's and master's education has been standardized by the Bologna Process in EU, PhD level education has considerable variance.

## 5 National Level Collaborative Activities Related to Computing Education

The Virtual University of Finland (VUF) was a collaborative network of Finnish universities, which started its operations in 2001. Despite its name, it was not an actual university, but an umbrella organization. It was a collection of university discipline networks that build multidisciplinary nationwide networks of activities. The aim was to promote the use of information and communication technologies and to develop cooperation among universities in various fields. In 2001–2006, the activities were funded by the Ministry of Education. Starting in 2007 the universities were supposed to be responsible for the funding, which did not realize well and basically led to closing VUF at the end of 2010. However, several projects which started during these years still remain active and get funding from several sources including the Ministry of Education.

The Basic Programming Education Network (BPEN) was one of the virtual university networks (2006–2008) in the field of mathematics and science. The purpose of the network was to promote the dissemination of specialized tools and materials used in basic programming courses in Finnish universities and to promote the networking of teachers in this area. The aim was to establish a high-quality and economical approach to teaching and research that relies heavily on the use of information and communication technologies, which also utilizes the latest research data in the field.

The network worked in close co-operation with the Computer Science Teaching SIG (CSTSIG), a thematic group within the Finnish Society for Computer Science. The aim of the theme group is to bring together teachers in the field and researchers interested in learning technology. The purpose of the joint network was to promote, e.g., collaboration between teachers, exchanging teaching and learning materials, introducing ICT-based tools to support teaching and learning computing and taking a stand on current societal issues in the field of teaching computing. The concrete form of activities included courses, network meetings, and seminars, which were regularly attended by about 40–50 teachers, researchers or other people who were interested in the network's activities. The seminars included workshops presenting a variety of ICT-based solutions that had been developed in Finnish universities to support teaching and learning computing. At the time, it was typical to bundle the tools on a USB stick, but share and update the content online. In addition, the network organized other opportunities to meet, for example, doctoral students

whose dissertation topic focused on learning technology or teaching computer science.

BPEN also had good international relations which enabled bringing together users and tool developers from many other countries. Many scholars visited Finland during the years and gave courses, for example, on automatic assessment tools, software visualization, and how to teach programming in general. BPEN was a three year project that ended in 2008; however, the teachers and researchers from many universities continued networking, which also led to close collaboration in CER and supported many doctoral research projects.

ÄlyOppi project (The Intelligent Systems and Content Creation project, 2018–2021) had roots which are heavily tied in the aforementioned networks. It was funded by the Ministry of Education and its goals were to develop new and improve existing online learning materials and environments for university level use in computer science, mathematics and physics. In computer science, which was the largest subproject, the aim was to develop tools for automatic assessment, visualizations and simulation, as well as improve existing tailored (for computer science education) learning environments in universities, and support their integration with each other. Thus, tools developed at different institutes could be used elsewhere and interactive learning resources, based on the tools, could be used in wider settings. The project significantly strengthened the network of CER people in Finland. It also organized a series of webinars presenting the results and accomplishments of the project for a wider audience in tertiary education.

In recent years, the Ministry of Education has strongly supported extending opportunities for life-long learning. A major networked project FITech is a flagship in computing education, which seeks to open widely university level computing courses for people in working life who wish to upgrade their skills in computing.[9] Similarly, a smaller project Digital Education for All, focused on opening first and second year computing courses and learning resources as MOOCs to people who would like to learn computing, but have not enrolled to universities. Moreover, sufficiently good performance in these open courses would allow them to continue their studies in formal computing degree programs in universities without the need to pass an entrance examination. While these projects are not research oriented as such, they allow collecting much data and experiences which can be investigated in CER.

## 6 Discussion

CER is flourishing in Finland with several research groups working actively in the area, as presented above. An interesting question emerges: what may be the background behind this phenomenon.

[9] https://fitech.io/en/.

## *6.1 Pioneering Teachers*

The roots of many CER groups emerge from solving challenges in university level education. A common problem in many institutes has been the shortage of teaching resources when compared with the number of students enrolling in introductory programming courses. This challenge has often been addressed by the extensive use of BSc level students as teaching assistants or tutors (while graduate students as teaching assistants may seem a more proper solution, they are more often recruited as teaching assistants for advanced courses). Another solution has been the development of tools to help teaching, learning and assessment, e.g., ViLLE in Turku, Jeliot in UEF and TRAKLA, Ceilidh and A+ in Aalto/TKK. These software projects have been driven by enthusiastic pioneers, typically junior teachers in charge of the courses. However, from early on, many groups have built much activity in school level education, too, such as the Kids CLub in UEF, IOI and Bebras contests in Tampere, and building the ViLLE collaborative learning environment in Turku. Interestingly, these K-12 level activities are not follow-ups of developments of the national level school curricula, because computational thinking and basics of programming have been included in school curriculum only from 2016 onwards. Previously, programming education has been organized in schools only as voluntary optional subjects.

However, much of this work can be considered more of computing education development than research. A highly important factor supporting the turn towards research have been the pioneering professors, who have had a strong personal interest in developing education and building their own research area associated with it, regardless of whether their primary research area or teaching responsibility has been some other area in computing or education. While faculty members generally have more or less academic freedom to choose their research foci, professors have stronger shoulders to push their own agenda forward despite their formal research areas. For example, Salakoski in Turku has worked mainly in bioinformatics but at the same time conceived, encouraged and supported work in CER. In Tampere, Järvinen, while working in software technology research, was also very interested in supporting education and thus provided the support for junior researchers and teachers to carry out research in CER. In Jyväskylä, Kärkkäinen was associated with teacher education, and supporting CER has been an easy extension. In the same way, in UEF, Sutinen was assigned the responsibility of teacher education and soon extended his work into a wide variation of themes in computing education, educational technologies, as well as ICT4D. At TKK/Aalto, Malmi was responsible for basic programming education and found there a fruitful symbiosis of carrying out pedagogical development and researching the impact of the implemented educational innovations. Thus, he could avoid the potential tension of carrying out research in some other computer science topics while having a heavy teaching load elsewhere. At University of Helsinki, the challenge was different. While there was a lot of educational development activity, there was no professor with a similar strong interest in the area before Ihantola got a tenure track professorship in Learning

Analytics in 2018. Before that, the solution was a grass-root level approach, led by Jaakko Kurhila and later on Matti Luukkainen and Arto Hellas, where a research group was formed to legitimize the activities of both tenured teachers and aspiring researchers.

Considering the most productive authors in Figs. 2 and 3, the pioneers and their PhD graduates are well presented. Among the list of people are two pioneering professors, Sutinen and Malmi and several of their early PhD graduates: Suhonen was Sutinen's student and Ihantola, Korhonen and Sorva are Malmi's students. Isomöttönen was the pioneering lecturer at Jyväskylä and Laakso was the first PhD graduate in Turku.

All this builds the big picture that individual people who are highly interested in improving education form the core of the success in Finland. None of them were initially nominated as professors with a research area CER, but they have used their academic freedom to target their main work in the field, or give active support for junior teachers or PhD students who wish to work in CER. The academic freedom and strong impact of pioneering individuals is also likely a reason for the richness of activities and research topics in Finnish groups. Each of the teams has clearly a unique profile. At TKK/Aalto, there is a long tradition of developing tools to support programming and data structures and algorithms courses, as well as researching programming education more widely. At University of Turku, ViLLE system that was initially developed to support programming education, has later on been extended to full scale learning environment supporting multiple disciplines and forms the core data collection tool for the Centre of Learning analytics. At UEF, Sajaniemi's research focused on cognitive aspects of learning programming. In addition, research in the edTech group in Joensuu around the Jeliot programming visualization tool became a long-term research track. However, soon the scope of activities in edTech widened very much, covering activities for broadening participation, distance education, robotics, contextualized CS education for developing countries and many more. Lappeenranta has a strong focus on software engineering education, while Tampere has focused on programming education and supporting computational thinking in K-12 level contests. At University of Helsinki, there has been much work in learning analytics and software engineering education, and Jyväskylä has had more emphasis in qualitative research in multiple topics.

## *6.2 Networking and Recruitment*

One of the strengths in Finland is actually the small size of the country, which enables easy networking between people who are interested in CER. From early on, the Koli Calling conference became a venue where most of the active people in the field convened annually. Moreover, as the conference fee was kept low to support international visitors, this—combined with low domestic travel costs—allowed teams to send PhD students to the conference with a poster only or simply as visitors. Many seniors also attend the conference regularly regardless of whether

they have a paper presentation there or not, just to meet people. This has built a strong network in Finland where people know each other. Moreover, the small size of the conference and its format has strongly supported PhD students to familiarize with international colleagues, too. Further support has been provided by the major national level education development projects which were discussed above in Sect. 5. There is a win-win situation. As people know each other, it is easy to build even national level consortia which can apply such network funding when appropriate funding calls are available. On the other hand, the network activities bring new people into the field.

Two conferences have had a major role in international networking in the early years. The ITiCSE conference was organized in Uppsala, Sweden in 1997 and in Helsinki in 2000. For the pioneers, these events gave a good spark for international collaboration. Moreover, over many years, successful experiences from participating ITiCSE working groups have further extended international networks, not only for seniors but for many PhD students. All these factors have contributed to the significant share of publications with authors from Finland and other countries since the early 2000s, as can be seen in Fig. 1.

Another strength in many groups has been the relative ease of recruiting students in an early phase in their MSc or even BSc level studies. Many of them have been working as teaching assistants on basic computing courses, because they have been interested in helping their peers. They are therefore well aware of challenges that younger students have, e.g., in learning programming and can generate new ideas for improvement when they are already working together with the course teacher. Many of these students are also interested in developing software in an academic environment where they can work to solve real problems and there is more freedom to choose their goals compared with working in internships in companies. Moreover, getting their name as an author in a publication which concerns the tool they have been implementing, is certainly motivating for those who have an interest in research. Often their master's thesis project may result not just a new publication but also a stepping stone for their PhD research. From the team's point of view, recruiting students early is also relevant because many CS students aim at an industrial career. Recruiting new students even for a MSc thesis project may be difficult, as an industry MSc thesis project is often the gate for a working position. The situation is different if a student has already been integrated in a research group, and considers the academic career, or at least doctoral studies, as an interesting option.

Another opportunity to recruit new people to CER is persuading teacher colleagues, who have done great work in developing their courses, to collaborate and write joint papers. However, this has not been an easy task. From their perspective, the most relevant research topic is naturally their own course and teaching, and other topics might be less attractive. Thus, the effort in attracting them to CER for a longer term may be difficult.

### 6.3 Challenges in Funding Research

Despite these positive aspects, there is no lack of challenges. The work in CER is often not valued as relevant or high quality research among people working in "real" computer science. Some people consider that CER should be carried out by educational scientists at departments of education instead of computer science departments. The only way to mitigate this challenge has been to build academic credibility in the field in terms of writing good quality papers, completing PhD theses which are assessed with the normal academic procedures, as well as organizing and participating in international conferences, i.e., working as any other academic disciplines do.

However, the above tension has made getting research funding for CER projects a major challenge in Finland, as well as in most other countries. In Finland, the key academic research funding institution is Academy of Finland. If a CER proposal is submitted to its council of technology and natural sciences, many proposals have been evaluated with low grades, because their research goals are not considered relevant or interesting enough for computing sciences. On the other hand, if a proposal has been submitted to the council of social sciences, educational scientists may consider that the team is not competent enough for such research. Naturally collaborative proposals where CER people and researchers from social sciences work together as a consortium, have somewhat better chances. However, the competition for funding is generally very high and therefore many CER funding applications have been targeted to other venues. One option has been EU funding, where many different funding instruments exist. The challenge there is building a good consortium which is large and versatile enough to match the general criteria of EU projects. Institutes from several countries around Europe should be included, but CER is not an active research area in many EU countries. Moreover, managing the project application and the actual project, if funded, is often very complex and laborious.

At TKK/Aalto, a central factor in the beginning was gaining the Center of Excellence on Education (CEE) status which provided significant long term funding for the department, of which a considerable share could be used to start work with new educational innovations. This long term funding enabled creating scientific results, which could be used to support future applications. Unfortunately, the Ministry of Education decided to cease the CEE program some 10 years later, and only the University of Helsinki team managed to get this kind of funding, in addition to TKK/Aalto.

When gaining major research funding has been difficult, other sources have been explored. Many foundations fund research, but typically for a short time only from a few man months to a man year. Some departments or universities provide their own funded PhD student positions, in most cases based on competitive applications. Some of those resources have been successfully applied to CER PhD students. The Ministry of Education and some NGOs have provided funding for developing education in universities, which have been used for networking activities, as written

above, but also for work which strongly supports research, such as implementing new educational software. One important NGO funding organization has been the Technology Industries of Finland Centennial Foundation. Finland is a high tech country and there is a constant shortage of competent people, especially in the IT sector. The foundation has supported projects in which education is developed in various fields of technology, and some major projects have been successfully applied to support work in CER. While such funding is basically funding for development of education or educational software, it can greatly support research. Developing novel software requires much work, but when the new tools are ready and being used in real courses, it is relatively straightforward to design studies which can lead to good publications.

Similar types of funding opportunities have occasionally been available from the Ministry of Education, too. For example, the above mentioned networking projects received such funding, and the latter, ÄlyOppi project also focused mainly on developing software and new learning resources based on the available tools. It is likely that such funding either from the Ministry or from universities themselves will increase, as life-long learning is a growing area for universities and it needs advanced technologies to support it. Covid-19 pandemic caused a major step towards wider availability of blended and online learning and this development will continue. People in working life cannot attend campus teaching for longer times, and they must be supported with online resources and facilities, a challenge addressed in the above mentioned FITech project. From the perspective of CER, such projects provide rich opportunities for extending research from programming education to education of advanced CS topics.

Problems in gaining funding have likely had an effect on the diversity of topics which have been researched. As major funding is rarely available, PhD students are funded from smaller projects which may not match well for collaborating with their peers. This has led in several cases, e.g. in UEF and Aalto to a situation where research is too much individualized, and PhD students work less as a team than would be beneficial for all.

## 7 Future

The future of CER in Finland seems quite promising. Current group leaders in CER have been either professors or lecturers, depending on the university. Some of the professors work mainly in CER themselves while others have their main focus elsewhere despite their interest in CER. For full-time lecturers especially in large computing degree programmes, CER work offers an attractive option to combine their educational duties with academic qualification. It is particularly delightful that recently several young people with a solid background in CER have been appointed as professors (Matti Tedre at UEF, Mikko-Jussi Laakso at UTU, Petri Ihantola at University of Helsinki, and Antti Knutas at LUT). Even though the main research area of their office may not be specifically CER but some close by area, such as

learning analytics, they can target much of their energy in CER, bringing continuity to the field. It remains to be seen whether professorships with CER as their main research area will be founded in the future.

Moreover, the actors in the field are very well networked. Collaboration is easy, if some new funding is possible to gain. This network has also greatly supported the Koli Calling conference. Interestingly, the conference does not have any formal organization behind. The university of Eastern-Finland CER team has organized it well for 20 years, and each year some Finnish CER researcher is the other program chair while the other one is an international chair. This brings much continuity with fairly low effort. But it is also a strong evidence of Finnish "talkoohenki", which denotes joint free work for a common goal.

## References


1. Friday Joseph Agbo. *Co-designing a smart learning environment to facilitate computational thinking education in the Nigerian context*. PhD thesis, University of Eastern Finland, 2022.
2. Alireza Ahadi, Raymond Lister, Heikki Haapala, and Arto Vihavainen. Exploring machine learning methods to automatically identify students in need of assistance. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, pages 121–130, 2015.
3. Tuukka Ahoniemi. *Efficient use of teaching technologies with programming education*. PhD thesis, Tampere University of Technology, 2015.
4. Kirsti Ala-Mutka. Tietokoneavusteinen ohjelmoinnin opetus. Master's thesis, Tampere University of Technology, 1998. In Finnish.
5. Kirsti Ala-Mutka. *Automatic assessment tools in learning and teaching programming*. PhD thesis, Tampere University of Technology, 2005.
6. Mikko Apiola. *Creativity-supporting learning environments: Two case studies on teaching programming*. PhD thesis, University of Helsinki, 2013.
7. Mikko Apiola, Sonsoles Lopez-Pernas, Mohammed Saqr, Arnold Pears, Mats Daniels, Lauri Malmi, and Matti Tedre. From a national meeting to an international conference: A scientometric case study of a Finnish computing education conference. *IEEE Access*, 2022.
8. Tapio Auvinen. Rubyric. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, pages 102–106, 2011.
9. Tapio Auvinen. *Educational technologies for supporting self-regulated learning in online learning environments*. PhD thesis, Aalto University, 2015.
10. Rosalina Babo. *Improving individual and collaborative e-assessment through multiple-choice questions and WebAVALIA - A new assessment strategy implemented at a Portuguese university*. PhD thesis, University of Eastern Finland, 2020.
11. Ralph-Johan Back, Linda Mannila, Mia Peltomäki, and Tapio Salakoski. Improving mathematics and programming education – the IMPEd initiative. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research, Australian Computer Society*, 88:161–170, 2007.
12. Ralph-Johan Back and Joakim Wright. *Refinement Calculus - A Systematic Introduction*. Springer, 1998.
13. Bebras international challenge on informatics and computational thinking. https://www.bebras.org/. Accessed: 2022-04-07.
14. Roman Bednarik, Andėrs Moreno, and Niko Myller. Program visualization for programming education - case of Jeliot3. *Association for Computing Machinery New Zealand Bulletin*, 2(2), 2006.

15. Moti Ben-Ari, Roman Bednarik, Ronit Ben-Bassat Levy, Gil Ebel, Andèrs Moreno, Niko Myller, and Erkki Sutinen. A decade of research and development on program animation: The Jeliot experience. *Journal of Visual Languages and Computing*, 22:375–384, 2011.
16. Steve Benford, Edmund Burke, Eric Foxley, Neil Gutteridge, and Abdullah Mohd Zin. Early experiences of computer-aided assessment and administration when teaching computer programming. *ALT-J*, 1(2):55–70, 1993.
17. Pauli Byckling and Jorma Sajaniemi. Roles of variables and programming skills improvement. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, pages 413–417, 2006.
18. European Commission. Survey of schools: ICT in education. benchmarking access, use and attitudes to technology in Europe's schools, 2013.
19. Valentina Dagiene, Linda Mannila, Timo Poranen, Lennart Rolandsson, and Pär Söderhjelm. Students' performance on programming-related tasks in an informatics contest in Finland, Sweden and Lithuania. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education*, pages 153–158, 2014.
20. Valentina Dagiene, Linda Mannila, Timo Poranen, Lennart Rolandsson, and Gabriele Stupuriene. Reasoning on children's cognitive skills in an informatics contest: Findings and discoveries from Finland, Lithuania, and Sweden. In *Proceedings of the International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 66–77, 2014.
21. Rodrigo Duran, Lassi Haaranen, and Arto Hellas. Gender differences in introductory programming: Comparing MOOCs and local courses. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 692–698, 2020.
22. Johannes Eriksson. Tool-supported invariant-based programming. *TUCS Dissertations 127. Turku Centre for Computer Science*, 2010.
23. Fabian Fagerholm, Arto Hellas, Matti Luukkainen, Kati Kyllönen, Sezin Yaman, and Hanna Mäenpää. Designing and implementing an environment for software start-up education: Patterns and anti-patterns. *Journal of Systems and Software*, 146:1–13, 2018.
24. Petri Gerdt and Jorma Sajaniemi. A web-based service for the automatic detection of roles of variables. *ACM SIGCSE Bulletin*, 38(3):178–182, 2006.
25. Jyrki Haajanen, Mikael Pesonius, Erkki Sutinen, Jorma Tarhio, Tommi Teräsvirta, and Pekka Vanninen. Animation of user algorithms on the web. In *Proceedings. 1997 IEEE Symposium on Visual Languages (Cat. No. 97TB100180)*, pages 356–363. IEEE, 1997.
26. Lassi Haaranen. *Game-related learning and exposure in computer science*. PhD thesis, Aalto University, 2019.
27. Lassi Haaranen, Petri Ihantola, Juha Sorva, and Arto Vihavainen. In search of the emotional design effect in programming. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 428–434. IEEE, 2015.
28. Arto Haataja, Jarkko Suhonen, Erkki Sutinen, and Sirpa Torvinen. High school students learning computer science over the web. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 3(2), 2001.
29. Lasse Hakulinen. *Gameful approaches for computer science education: From gamification to alternate reality games*. PhD thesis, Aalto University, 2015.
30. Mustafa Muhammad Hassan, Andèrs Moreno, Erkki Sutinen, and Abdul Azil. On the participatory design of Jeliot Mobile: Towards a socio-constructivist mlearning tool. In *Proceedings of the International Conference on Learning and Teaching in Computing and Engineering*, pages 120–123, 2003.
31. Juho Heikkinen. Conceptualizing the role of multidisciplinarity and student perceptions of university-industry collaboration in project-based learning. In *Jyväskylä studies in computing*, volume 264. University of Jyväskylä, 2016.
32. Arto Hellas. *Retention in introductory programming*. PhD thesis, University of Helsinki, 2017.
33. Juha Helminen. *Supporting acquisition of programming skills in introductory programming education: Environments for practicing programming and recording and analysis of exercise sessions*. PhD thesis, Aalto University, 2014.

34. Antti Herala, Erno Vanhala, Antti Knutas, and Jouni Ikonen. Teaching programming with flipped classroom method: A study from two programming courses. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, pages 165–166, 2015.
35. Leena Hiltunen. Enhancing web course design using action research. In *Jyväskylä Studies in Computing*, volume 125. University of Jyväskylä, 2010.
36. Ville Hämäläinen and Ville Isomöttönen. What did CS students recognize as study difficulties? In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2019.
37. Roya Hosseini, Peter Brusilovsky, Michael Yudelson, and Arto Hellas. Stereotype modeling for problem-solving performance predictions in MOOCs and traditional courses. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 76–84, 2017.
38. Maija Hujala, Antti Knutas, Timo Hynninen, and Heli Arminen. Improving the quality of teaching by utilising written student feedback: A streamlined process. *Computers & Education*, 157:103965, 2020.
39. Juha Hyvönen and Lauri Malmi. TRAKLA - a system for teaching algorithms using email and a graphical editor. In *HYPERMEDIA in Vaasa, 1993*, pages 141–147. University of Vaasa, Finland, 1993.
40. Petri Ihantola. *Automated assessment of programming assignments: visual feedback, assignment mobility, and assessment of students' testing skills*. PhD thesis, Aalto University, 2011.
41. Petri Ihantola and Ville Karavirta. Open source widget for parson's puzzles. In *Proceedings of the Fifteenth Annual conference on Innovation and Technology in Computer Science Education*, pages 302–302, 2010.
42. Kalle Ilves, Juho Leinonen, and Arto Hellas. Supporting self-regulated learning with visualizations in online learning environments. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 257–262, 2018.
43. International Olympiad in Informatics, Statistics. https://stats.ioinformatics.org/countries/. Accessed: 2022-04-07.
44. Essi Isohanni. *Visualizations in learning programming: Building a theory of student engagement*. PhD thesis, Tampere University of Technology, 2013.
45. Ville Isomöttönen. Theorizing a one-semester real customer student software project course. In *Jyväskylä Studies in Computing*, volume 140. University of Jyväskylä, 2011.
46. Ville Isomöttönen, Antti-Jussi Lakanen, and Vesa Lappalainen. Less is more! Preliminary evaluation of multi-functional document-based online learning environment. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE, 2019.
47. Ville Isomöttönen and Emmi Ritvos. Digging into group establishment: Intervention design and evaluation. *Journal of Systems and Software*, 178:110974, 2021.
48. Ville Isomöttönen and Ville Tirronen. Flipping and blending — An action research project on improving functional programming course. *ACM Transactions on Computing Education Research*, 17(1):1:1–1:35, 2017.
49. Pertti Järvinen. Notes on educational planning: a systems approach. *ACM SIGCSE Bulletin*, 9(4):57–62, 1977.
50. Ilkka Jormanainen. *Supporting teachers in unpredictable robotics learning environments*. PhD thesis, University of Eastern Finland, 2013.
51. Ilkka Jormanainen and Markku Tukiainen. Attractive educational robotics motivates younger students to learn programming and computational thinking. In *Proceedings of the Eighth International Conference Technological Ecosystem for Enhancing Multiculturality Conference*, 2020.
52. Erkki Kaila. Utilizing educational technology in computer science and programming courses: theory and practice. *TUCS Dissertations 230. Turku Centre for Computer Science*, 2018.
53. Ville Karavirta. *Facilitating algorithm visualization creation and adoption in education*. PhD thesis, Helsinki University of Technology, 2009.
54. Ville Karavirta, Petri Ihantola, and Teemu Koskinen. Service-oriented approach to improve interoperability of e-learning systems. In *2013 IEEE 13th International Conference on Advanced Learning Technologies*, pages 341–345. IEEE, 2013.

55. Ville Karavirta, Ari Korhonen, Lauri Malmi, and Kimmo Stålnacke. Matrixpro - A tool for ex tempore demonstration of data structures and algorithms. In *Proceedings of the Third Program Visualization Workshop, University of Warwick, UK*, pages 27–33, 2004.
56. Ville Karavirta and Clifford A Shaffer. Creating engaging online learning material with the JSAV javascript algorithm visualization library. *IEEE Transactions on Learning Technologies*, 9(2):171–183, 2015.
57. Jyri Kemppainen. *Appropriating IT service management education in a Tanzanian university: Global and local perspectives*. PhD thesis, University of Eastern Finland, 2014.
58. Teemu Kerola and Harri Laine. SQL-trainer. *Kolin Kolistelut–Koli Calling Proceedings of the First Annual Finnish/Baltic Sea Conference on Computer Science Education*, 2001.
59. Teemu Kerola and Harri Laine. Creation of self tests and exam questions as a learning method. *Kolin Kolistelut—Koli Calling. Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education*, 2004.
60. Päivi Kinnunen. *Challenges of teaching and studying programming at a university of technology-Viewpoints of students, teachers and the university*. PhD thesis, Helsinki University of Technology, 2009.
61. Päivi Kinnunen and Lauri Malmi. Do students work efficiently in a group? - Problem-based learning groups in basic programming course. In *Kolin Kolistelut - Koli Calling Proceedings of the Fourth Finnish/Baltic Sea Conference of Computer Science Education*, pages 57–66. Citeseer, 2004.
62. Antti Knutas, Timo Hynninen, and Maija Hujala. To get good student ratings should you only teach programming courses? Investigation and implications of student evaluations of teaching in a software engineering context. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 253–260. IEEE, 2021.
63. Antti Knutas, Jouni Ikonen, and Jari Porras. Communication patterns in collaborative software engineering courses: A case for computer-supported collaboration. In *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*, pages 169–177, 2013.
64. Antti Knutas, Ahmed Seffah, Lene Sorensen, Andrey Sozykin, Fawaz Al-Zaghoul, and Alain Abran. Crossing the borders and the cultural gaps for educating PhDs in software engineering. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, pages 256–265. IEEE, 2017.
65. Antti Knutas, Rob Van Roy, Timo Hynninen, Marco Granato, Jussi Kasurinen, and Jouni Ikonen. A process for designing algorithm-based personalized gamification. *Multimedia Tools and Applications*, 78(10):13593–13612, 2019.
66. Sandhya Kode. *Enhancing Information Technology education in Indian context: a design story*. PhD thesis, University of Eastern Finland, 2019.
67. Marja Kopponen. *CAI in CS*. PhD thesis, University of Joensuu, 1997.
68. Ari Korhonen. *Visual algorithm simulation*. PhD thesis, Helsinki University of Technology, 2003.
69. Ari Korhonen and Lauri Malmi. Algorithm simulation with automatic assessment. In *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education*, pages 160–163, 2000.
70. Jami Kousa, Petri Ihantola, Arto Hellas, and Matti Luukkainen. Teaching container-based devops practices. In *International Conference on Web Engineering*, pages 494–502. Springer, 2020.
71. Jaakko Kurhila. *Considering individual differences in computer-supported special and elementary education*. PhD thesis, University of Helsinki, 2003.
72. Jaakko Kurhila and Erkki Sutinen. Sharing an open learning space by individualizing agents. *Journal of Interactive Learning Research*, 10(3):287, 1999.
73. Jaakko Kurhila and Erkki Sutinen. From intelligent tutoring systems to intelligent learning materials. In *EdMedia+ Innovate Learning*, pages 546–551. Association for the Advancement of Computing in Education (AACE), 2000.

74. Jaakko Kurhila and Arto Vihavainen. Management, structures and tools to scale up personal advising in large programming courses. In *Proceedings of the 2011 Conference on Information Technology Education*, pages 3–8, 2011.
75. Jaakko Kurhila and Arto Vihavainen. A purposeful MOOC to alleviate insufficient CS education in Finnish schools. *ACM Transactions on Computing Education (TOCE)*, 15(2):1–18, 2015.
76. Einari Kurvinen. Effects of regular use of scalable, technology enhanced solution for primary mathematics education. *TUCS Dissertations 260. Turku Centre for Computer Science*, 2020.
77. Mikko-Jussi Laakso. Promoting programming learning. Engagement, automatic assessment with immediate feedback in visualizations. *TUCS Dissertations 131. Turku Centre for Computer Science*, 2010.
78. Mikko-Jussi Laakso, Erkki Kaila, and Teemu Rajala. ViLLE – collaborative education tool: Designing and utilizing an exercise-based learning environment. *Education and Information Technologies*, 23:1655—1676, 2018.
79. Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3):14–18, 2005.
80. SP Lahtinen, T Lamminjoki, E Sutinen, J Tarhio, and AP Tuovinen. Towards automated animation of algorithms. In *Proceedings of Fourth International Conference in Central Europe on Computer Graphics and Visualization*, volume 96, pages 150–161, 1996.
81. Tommi Lahtonen and Ville Isomöttönen. Parsi: A tool for automatic assessment of office documents and basic IT skills. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, pages 174–180, New York, NY, 2012. ACM.
82. Antti-Jussi Lakanen. On the impact of computer science outreach events on K-12 students. In *Jyväskylä studies in computing*, volume 236. University of Jyväskylä, 2016.
83. Vesa Lappalainen, Jonne Itkonen, Ville Isomöttönen, and Sami Kollanus. Comtest: A tool to impart TDD and unit testing to introductory level programming. In *ITiCSE '10: Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education*, pages 63–67, New York, NY, 2010. ACM.
84. Jari Laru, Kati Mäkitalo, Matti Tedre, Teemu Valtonen, and Henriikka Vartiainen. Ohjelmoinnista digitaalisen ajatteluun – kuinka edistää ohjelmoinnin ja tietotekniikan opetusta esi- ja alkuopetuksessa. In *Esi- ja alkuopetuksen käsikirja*, pages 243–268. PS-Kustannus, 2020.
85. Juho Leinonen. *Keystroke data in programming courses*. PhD thesis, University of Helsinki, 2019.
86. Juho Leinonen, Petri Ihantola, and Arto Hellas. Preventing keystroke based identification in open data sets. In *Proceedings of the Fourth (2017) ACM Conference on Learning@Scale*, pages 101–109, 2017.
87. Juho Leinonen, Petri Ihantola, Antti Leinonen, Henrik Nygren, Jaakko Kurhila, Matti Luukkainen, and Arto Hellas. Admitting students through an open online course in programming: A multi-year analysis of study success. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, pages 279–287, 2019.
88. Juho Leinonen, Krista Longi, Arto Klami, and Arto Vihavainen. Automatic inference of programming performance and experience from typing patterns. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 132–137, 2016.
89. Krista Longi, Juho Leinonen, Henrik Nygren, Joni Salmi, Arto Klami, and Arto Vihavainen. Identification of programmers from typing patterns. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, pages 60–67, 2015.
90. Jan Lönnberg. *Understanding and debugging concurrent programs through visualisation*. PhD thesis, Aalto University, 2012.
91. Jan Lönnberg, Mordechai Ben-Ari, and Lauri Malmi. Java replay for dependence-based debugging. In *Proceedings of the Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging*, pages 15–25, 2011.

92. Sonsoles López-Pernas, Mohammed Saqr, and Mikko Apiola. Scientometrics: A concise introduction and a detailed methodology for the mapping of the scientific field of computing education. In Mikko Apiola, S López-Pernas, and Mohammed Saqr, editors, *Past, Present and Future of Computing Education Research*, pages xx–yy. Springer, 2023.
93. Sonsoles Lòpez-Pernas, Muhammed Saqr, and Olga Vberg. Putting it all together: Combining learning analytics methods and data sources to understand students' approaches to learning programming. *Sustainability*, 13(9), 2021.
94. Matti Luukkainen, Arto Vihavainen, and Thomas Vikberg. A software craftsman's approach to data structures. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, pages 439–444, 2012.
95. Hanna Mäenpää, Samu Varjonen, Arto Hellas, Sasu Tarkoma, and Tomi Männistö. Assessing IOT projects in university education - A framework for problem-based learning. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pages 37–46. IEEE, 2017.
96. Lauri Malmi, Ville Karavirta, Ari Korhonen, Jussi Nikander, Otto Seppälä, and Panu Silvasti. Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in education*, 3(2):267–288, 2004.
97. Linda Mannila. Teaching mathematics and programming - new approaches with empirical evaluation. *TUCS Dissertations 124. Turku Centre for Computer Science*, 2009.
98. Linda Mannila, Mia Peltomäki, and Tapio Salakoski. What about a simple language? Analyzing the difficulties in learning to program. *Computer Science Education*, 16(3):211–228, 2006.
99. Andèrs Moreno. *Re-designing program animation*. PhD thesis, University of Eastern Finland, 2014.
100. Salsen Mrong, Ilkka Jormanainen, and Tapani Toivonen. Visualization tool for teaching and learning artificial neural networks. In *Proceedings of the 9th Technological Ecosystems for Enhancing Multiculturality Conference*, 2021.
101. Nkundwe Moses Mwasaga, Mikko Apiola, Jarkko Suhonen, and Mike Joy. Integrating high performance computing into a Tanzanian IT engineering curriculum. In *Proceedings of the 21st ICE/IEEE International Technology Management Conference*, 2015.
102. Niko Myller. *Collaborative software visualization for learning: theory and applications*. PhD thesis, University of Joensuu, 2009.
103. Matti Nelimarkka and Arto Hellas. Social help-seeking strategies in a programming MOOC. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 116–121, 2018.
104. Seppo Nevalainen and Jorma Sajaniemi. Comparison of three eye tracking devices in psychology of programming research. In *Proceedings of the 16th Annual Workshop of the Psychology of Programming Interest Group*, 2004.
105. Pia Niemelä. *From Legos and Logos to Lambda: A hypothetical learning trajectory for computational thinking*. PhD thesis, Tampere University of Technology, 2018.
106. Pia Niemelä, Aulikki Hyrskykari, Timo Poranen, Heikki Hyyrö, and Juhani Linna. Flipped learning with peer reviews in the introductory CS course. In *Assessment, Testing, and Measurement Strategies in Global Higher Education*, pages 35–58. IGI Global, 2020.
107. Pia Niemelä, Tiina Partanen, Linda Mannila, Timo Poranen, and Hannu-Matti Järvinen. Code ABC MOOC for math teachers. In P. Escudeiro, G. Costagliola, S. Zvacek, J. Uhomoibhi, and B. McLaren, editors, *Proceedings of International Conference on Computers Supported Education. CSEDU 2017. Communications in Computer and Information Science*, pages 66–96. Springer, 2017.
108. Jussi Nikander. *Interaction and visualization methods in teaching spatial algorithms and analyzing spatial data*. PhD thesis, Aalto University, 2012.
109. Jussi Nikander, Juha Helminen, and Ari Korhonen. Algorithm visualization system for teaching spatial data algorithms. *Journal of Information Technology Education*, 9, 2010.

110. Uolevi Nikula, Jorma Sajaniemi, Matti Tedre, and Stuart Wray. Python and roles of variables in introductory programming: Experiences from three educational institutions. *Journal of Information Technology Education*, 6:199–214, 2007.
111. Jyrki Nummenmaa, Erkki Mäkinen, and Isto Aho (eds.). IOI'2001 competition. Technical report A-2001-7, University of Tampere, Department of Computer and Information Sciences, 2001.
112. Aletta Nylén and Ville Isomöttönen. Exploring the critical incident technique to encourage reflection during project-based learning. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, pages 88–97, New York, NY, 2017. ACM.
113. Solomon Sunday Oyelere. *Design and development of a mobile learning system for computer science education in Nigerian higher education context*. PhD thesis, University of Eastern Finland, 2018.
114. Jukka Paakki. *Opista tieteeksi - Suomen tietojenkäsittelytieteiden historia*. Tietojenkäsittelytieteen Seura ry, 2014.
115. Tiina Partanen, Pia Niemelä, and Timo Poranen. Racket programming material for Finnish elementary math education. In *Proceedings of Constructionism 2018*, pages 415–425, 2018.
116. Kukka-Maaria Polso, Heta Tuominen, Arto Hellas, and Petri Ihantola. Achievement goal orientation profiles and performance in a programming MOOC. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pages 411–417, 2020.
117. Timo Poranen, Valentina Dagienė, Åsmund Eldhuset, Heikki Hyyrö, Kubica Marcin, Antti Laaksonen, Mārtiņš Opmanis, Wolfgang Pohl, Jūratė Skūpienė, Pär Söderhjelm, and Ahto Truu. Baltic olympiads in informatics: Challenges for training together. *Olympiads in Informatics*, 3:112–131, 2009.
118. Jari Porras, Antti Knutas, Jouni Ikonen, Ari Happonen, Jayden Khakurel, and Antti Herala. Code camps and hackathons in education - literature review and lessons learned. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2019.
119. Teemu Rajala, Mikko-Jussi Laakso, Erkki Kaila, and Tapio Salakoski. Ville: A language-independent program visualization tool. *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research, Australian Computer Society*, 88:151–159, 2007.
120. Jorma Sajaniemi and Marja Kuittinen. Three-level teaching material for computer-aided lecturing. *Computers & Education*, 32:269–284, 1999.
121. Jorma Sajaniemi and Marja Kuittinen. An experiment using roles of variables in teaching introductory programming. *Computer Science Education*, 15(1):59–82, 2005.
122. Jorma Sajaniemi, Marja Kuittinen, and Taina Tikansalo. A study of the development of students' visualization of program state during an elementery object-oriented programming course. *Journal on Educational Resources in Computing*, 7(3):1–31, 2005.
123. Ismaila Temitayo Sanusi, Solomon Sunday Oyelere, and Joseph Olamide Omidora. Exploring teachers' preconceptions of teaching machine learning in high school: A preliminary insight from Africa. *Computers and Education Open*, 3, 2021.
124. Otto Seppälä. *Advances in assessment of programming skills*. PhD thesis, Aalto University, 2012.
125. Teemu Sirkiä. *Creating, tailoring, and distributing program animations-Supporting the production process of interactive learning content*. PhD thesis, Aalto University, 2017.
126. Teemu Sirkiä. Jsvee & Kelmu: Creating and tailoring program animations for computing education. *Journal of Software: Evolution and Process*, 30(2):e1924, 2018.
127. Teemu Sirkiä and Lassi Haaranen. Improving online learning activity interoperability with ACOS server. *Software: Practice and Experience*, 47(11):1657–1676, 2017.
128. Juha Sorva. *Visual program simulation in introductory programming education*. PhD thesis, Aalto University, 2012.
129. Juha Sorva and Teemu Sirkiä. UUhistle: A software tool for visual program simulation. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pages 49–54, 2010.

130. Calkin Suero Montero. Facilitating computational thinking through digital fabrication. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*, 2018.
131. Calkin Suero Montero and Ilkka Jormanainen. Theater meets robot – toward inclusive STEAM education. In *Educational Robotics in the Makers Era (Edurobotics 2016), Advances in Intelligent Systems and Computing*, pages 34–40. Springer, 2017.
132. Jarkko Suhonen. *A formative development method for digital learning environments in sparse learning communities*. PhD thesis, University of Joensuu, 2005.
133. Jarkko Suhonen and Erkki Sutinen. Learning computer science over the web: the ViSCoS odyssey. In *Cases on Global E-learning Practices: Successes and Pitfalls*, pages 176–188. IGI Global, 2007.
134. Jarkko Suhonen and Erkki Sutinen. The four pillar model - analysing the sustainability of online doctoral programmes. *TechTrends*, 58:81–88, 2014.
135. Ahmad Taherkhani. *Automatic algorithm recognition based on programming schemas and beacons - A supervised machine learning classification approach*. PhD thesis, Aalto University, 2013.
136. Toni Taipalus. Persistent errors in query formulation. In *JYU dissertations*, volume 283. University of Jyväskylä, 2020.
137. Toni Taipalus, Hilkka Grahn, and Hadi Ghanbari. Error messages in relational database management systems: A comparison of effectiveness, usefulness, and user confidence. *Journal of Systems and Software*, 181:111034, 2021.
138. Matti Tedre, Henriikka Vartiainen, Juho Kahila, Tapani Toivonen, and Valtonen Teemu. Machine learning introduces new perspectives to data agency in K-12 computing education. In *Proceedings of the IEEE Frontiers in Education Conference*. IEEE, 2020.
139. Matti Tedre, Henriikka Vartiainen, Juho Kahila, Tapani Toivonen, Teemu Valtonen, Ilkka Jormanainen, and Arnold Pears. Teaching machine learning in K-12 classroom: Pedagogical and technological trajectories for artificial intelligence education. *IEEE Access*, 9:110558–110572, 2021.
140. Ville Tirronen and Ville Isomöttönen. On the design of effective learning materials for supporting self-directed learning of programming. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, pages 74–82, New York, NY, 2012. ACM.
141. Ville Tirronen, Vesa Lappalainen, Ville Isomöttönen, Antti-Jussi Lakanen, Toni Taipalus, Paavo Nieminen, and Anthony Ogbechie. Incorporating teacher-student dialogue into digital course material: Usage patterns and first experiences. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE, 2020.
142. Tapani Toivonen, Ilkka Jormanainen, Calkin Suero Montero, and Andrea Alessandrini. Innovative maker movement platform for K-12 education as a smart learning environment. In *Proceeding of 2018 International Conference on Smart Learning Environments, Challenges and Solutions in Smart Learning, Lecture Notes in Educational Technology*, pages 61–66, 2018.
143. UNESCO. UNESCO prize awarded to a collaborative learning platform ViLLE from Finland. https://en.unesco.org/news/unesco-prize-awarded-collaborative-learning-platform-ville-finland. Accessed: 2022-05-04.
144. Teemu Valtonen, Matti Tedre, Kati Mäkitalo, and Henriikka Vartiainen. Media literacy education in the age of machine learning. *Journal of Media Literacy Education*, 11(2):20–36, 2019.
145. Ashok Kumar Veerasamy. Predictive models as early warning systems for student academic performance in introductory programming. *TUCS Dissertations 259. Turku Centre for Computer Science*, 2020.
146. Mikko Vesisenaho. *Developing university-level introductory ICT education in Tanzania: a contextualized approach*. PhD thesis, University of Joensuu, 2007.
147. Mikko Vesisenaho, Jyri Kemppainen, Carolina Islas Sedano, Matti Tedre, and Erkki Sutinen. How to contextualize ICT in higher education: A case study in Tanzania. *African Journal of Information & Communication Technology*, 2(2):88–109, 2006.

148. Arto Vihavainen, Matti Paksula, and Matti Luukkainen. Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, pages 93–98, 2011.
149. Arto Vihavainen, Thomas Vikberg, Matti Luukkainen, and Martin Pärtel. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, pages 117–122, 2013.
150. Thomas Vikberg, Arto Vihavainen, Matti Luukkainen, and Jaakko Kurhila. Early start in software coaching. In *International Conference on Agile Software Development*, pages 16–30. Springer, 2013.
151. Marjo Virnes. *Four seasons of educational robotics: Substansive theory on the encounters between educational robotics and children in the dimension of access and ownership*. PhD thesis, University of Eastern Finland, 2014.

# Computing Education Research in Australasia


**Simon, Judy Sheard, Andrew Luxton-Reilly, and Claudia Szabo**

## 1 Introduction

Although the word 'Australasia' has a number of different definitions, it refers most frequently to the two countries Australia and New Zealand together. Based on their populations, both of these countries have contributed disproportionately high numbers of publications to certain international computing education venues [125–127]. As reported in the ACM Digital Library, seven of the top 50 most prolific authors of SIGCSE publications are Australasian [2]. This chapter will survey some of the computing education research carried out in Australasia, and will suggest possible reasons for the high numbers of international computing education publications from these two countries.

New Zealand is a small island nation in the south Pacific with a population of approximately 5 million. It supports eight public funded universities and sixteen vocationally-focused polytechnic institutions. Polytechnics began offering formal programmes in computing in the 1960's, and Computer science departments were founded in Universities across New Zealand during the 1970s and 1980s, resulting today in significant computing departments in six universities: Auckland

Simon (✉)
Wadalba, NSW, Australia

J. Sheard
Monash University, Melbourne, Australia
e-mail: judy.sheard@monash.edu

A. Luxton-Reilly
University of Auckland, Auckland, New Zealand
e-mail: andrew@cs.auckland.ac.nz

C. Szabo
University of Adelaide, Adelaide, Australia
e-mail: claudia.szabo@adelaide.edu.au



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_17

University of Technology, Massey University, University of Auckland, University of Canterbury, University of Otago, University of Waikato, and Victoria University of Wellington.

Australia is a somewhat larger island nation a couple of thousand kilometres west of New Zealand, and has a population of a little more than 25 million. While Australia once had institutions comparable to New Zealand's polytechnics, a reform in its tertiary education system led to these institutions becoming universities, and it now has 36 public and seven private universities.

New Zealand and Australia were both colonised by England, in rather different manners, but their development since colonisation has been somewhat parallel and always collaborative.

## 2 Computing Education Publications from Australasia

New Zealand's contribution to the computing education literature began even before the establishment of computer science departments in the country, with a notable publication in the first SIGCSE Technical Symposium in 1970 discussing the challenges of introducing computing education in New Zealand universities [94]. Although it was difficult to access computing education conferences from the remote location of New Zealand, a scattering of publications throughout the 1970s, 1980s and early 1990s demonstrates an interest in computing education, although typically by academics with a primary focus in other disciplinary fields who leverage their expertise in a teaching context, discussing approaches to teaching their disciplinary subject matter. Examples include work related to computer architecture [20], database systems [96], artificial intelligence [153], and object-oriented programming [12].

A review of New Zealand's tertiary education in the late 1980's led to the 1988 foundation of the National Advisory Committee on Computing Qualifications (NACCQ), which focused on national computing qualifications in the polytechnic sector, and the New Zealand Education Act 1989, which gave polytechnics the authority to award degrees. The annual NACCQ conference (later renamed CITRENZ) provided a platform for computing educators across the tertiary sector to share scholarship of teaching and learning and computing education research.

While Australian computing educators also published at the SIGCSE Technical Symposium, both they and their New Zealand colleagues felt that there might be many more publications were it not for the expense of travelling to the United States. Based on the observation that approximately 10% of papers at the 1996 Technical Symposium were from Australasia [109], a group of academics at the University of Sydney proposed and held the first Australasian Conference on Computer Science Education in 1996. That conference, since renamed the Australasian Computing Education Conference, is still running, and while the majority of its publications are from Australia and New Zealand, some 14% of its papers are entirely from other countries [128], so it is clearly more than a regional conference.

The year in which ACE was established also saw the inception of ITiCSE, a computing education conference based in Europe. This was further to travel than the USA, but was better suited to many Australasian researchers, partly because it is held during the winter break of most Australasian institutions, and also because its submissions are due in their summer break, when it can be easier to make time to prepare submissions. ITiCSE's working groups also provided the opportunity for researchers to become involved with multinational research groups. And while publication at ACE had become a new option, publication at ITiCSE or the SIGCSE Technical Symposium promised greater international exposure for the work.

## 3 Building a Community

Several activities have been influential in encouraging and supporting computing education researchers in Australasia and helping to build and shape the community.

### *3.1 Conventicles*

In the early 2000s, attending a conference in the USA, Australian researcher Raymond Lister met for the first time some researchers from a university very near his own. Reflecting on the notion that he had travelled such a distance to find out about the work of people quite nearby, he devised the computing education 'conventicles' [81], one-day gatherings at which people would re-present papers already published at overseas conferences, for the purpose of sharing their work with a more local community. Conventicles were held, with varying persistence, in six major Australian cities, and one was recently held in New Zealand. These were an excellent way for computing education researchers to communicate with colleagues in their own vicinity, and have given rise to a number of productive collaborations.

### *3.2 BRACE and BRACElet*

The early decades of computing education publication consisted principally of show-and-tell papers, or experience reports, in which educators wrote about what they had tried in the classroom and what students thought of it. Gradually, though, an appreciation grew of the value that empirical research could bring: for example, it could help to establish whether these innovations had any appreciable impact on student outcomes.

Following the lead of two US-based projects, New Zealand's Anthony Robins and Australia's Raymond Lister brought to Australasia a project called Building Research in Australasian Computing Education, or BRACE. Held in conjunction

with ACE 2004, the project effectively launched computing education research in Australasia. A subsequent project, BRACElet, focused on multi-institutional research involving novice programming. The BRACElet project continued for 6 years and included thirteen workshops [29]. The early workshops were a vehicle to introduce new researchers into the computing education research community, and were instrumental in developing several influential avenues of research, including exploring relationships between types of programming skills [84], explain in plain English questions, and Parsons problems [36].

## 4 A Selection of Australasian Projects

In the remainder of this chapter we shall look briefly at some highlights of the computing education research that has been conducted in New Zealand and Australia.

### *4.1 Introductory Programming*

Studies of introductory programming have been a focus of work in Australasia since the late 1990s.

In New Zealand an early introductory programming paper describes a teaching approach for explaining the object-oriented concept of inheritance [13]. Work in Otago demonstrated that students struggled with programming competencies [55] due to a variety of factors [110]. A broad review of introductory programming in 2003 [107] focused on the difficulties that students encountered, and advanced an explanation for these difficulties, proposing that the inter-relationships between programming concepts leads misunderstandings to snowball [105]. More recently, this view has been challenged [86], suggesting that assessment practices may contribute more than subject matter to student difficulties [90], and the pass rates of introductory programming courses have been shown to be similar to those of other STEM courses [133]. A more recent review of literature on introductory programming [91] suggests that teachers are deploying a wide range of innovative teaching approaches, with substantial effort invested in development of tools to support student learning, and that students are frequently positive about their experiences .

In Australia, an early work explored the relationship between prior experience in programming and success in an introductory programming course, finding that students with experience perform significantly better than those with none [57]. Another study explored internal factors influencing the learning of programming using motivation and capability as a framework [19]. An investigation of the characteristics of programming tasks that lead to poor learning behaviours inspired the formulation of guidelines for task design for introductory programming students [18].

An Australian project led by Sheard explored what constitutes good practice in the first year of computing programs, including introductory programming. The project explored and documented issues concerning what we teach, where we teach, how we teach, how we assess, how we strengthen the learning environment, and how we support our students. It gave rise to publications on teaching [95], the teaching context [17], assessment [118], academic integrity [119], and student engagement [16].

### *4.2 Programming at All Levels*

BABELnot, another project led by Lister, was designed to formulate and document standards for the formal description of programming courses. One goal of the work was to facilitate meaningful comparison of programming courses around the world, thus making it easier to determine whether research findings based on one course might be applicable to other courses. Running for several years, the project produced 40 peer-reviewed publications, and gave rise to several subprojects that will be mentioned separately in this chapter.

### *4.3 Parson's Problems*

Parson's programming puzzles were introduced in 2006 as a web-based tool used at Otago Polytechnic to provide a fun way for students to focus on problem solving in a constrained environment that could provide immediate feedback [99]. The value in constrained problems was appreciated by those involved in BRACElet, leading to two papers investigating the use of Parson's problems (as they were renamed) in exams [36, 84]. These programming exercises are used as a form of scaffolding for code-writing exercises, and form the basis for a growing area of research [44].

### *4.4 Development of BlueJ*

BlueJ, a widely used integrated development environment (IDE) designed specifically for teaching and learning object-oriented programming, was first used in an introductory programming course in 1999 [57]. BlueJ has a graphical user interface through which students create and manipulate objects. A motivation for the development of BlueJ was the perceived lack of suitable programming environments for teaching introductory programming [71, 72]. The design of BlueJ is founded on sound pedagogical principles [74] that address the issues identified in other programming environments. In addition to the BlueJ IDE, guidelines for better course design and teaching approaches were developed [73]. From this early beginning, BlueJ gained and has maintained international recognition.

### 4.5 Learning Progression

In an investigation of the relationship between code tracing and code writing, naturally occurring data in the form of student test and exam results were categorised according to Bloom's taxonomy as it applies to computing [145] and the SOLO taxonomy [152]. Several significant papers investigating the relationship between programming skills followed, frequently using Bloom and SOLO to categorise questions and solutions. Subsequently, exam questions with the phrase "Explain in plain English" were deployed in examinations, and the relationship between code tracing, code writing, and code explaining was explored [84].

Work on scaffolding of problems has shown that students are more able to solve complex tasks when they have been presented with a collection of simpler but related tasks [35]. Additionally, focusing student attention on tasks through meta-cognitive prompts has been shown to improve outcomes [101, 102].

### 4.6 Student Learning Behaviour

Student behaviour is a key element in successful learning. Several avenues of research have investigated ways of affecting student behaviour, either explicitly through graded course activities, or implicitly through gamification of activities, or supporting progress through aspects of programming, such as debugging, that may block progression.

Student learning through peer interaction (sometimes referred to as contributing student pedagogies [58]) is explored in work on pair programming [93], peer assessment [59, 60, 63, 66, 85], and tools that focus on student-generated instructional content [61], such as PeerWise [33], CodeWrite [38], StudySieve [88].

Several studies have focused on understanding behaviours of students using empirical data, such as the use that students make of lecture recordings [100], and how simple changes in course structures, such as the timing of feedback, may influence those behaviours [79]. Gamification is one effective approach that increases the amount of activity in learning tools [32, 40, 64].

### 4.7 Assessment of Programming

A key concern in programming education is how to fairly and accurately assess student learning outcomes. Summative assessment of programming has been traditionally done via end-of-course formal examinations. A detailed investigation of a sample of introductory programming exam papers was conducted using a classification scheme for programming questions developed for the study [121]. The main finding was that there is great variation in the structure of exams and

the type and complexity of the questions [120, 135]. The variation in questions motivated work to develop a set of assessment questions that were benchmarked across programming courses in multiple institutions [123, 137, 138]. A further outcome of the work was a set of guiding principles for the design of programming exam questions [139]. A further study investigated the strategies that academics use to design their exams, resulting in greater understanding of the underlying principles used in the design of programming exams and the pedagogical intentions of the educators who construct these instruments [122].

Later, the assessments used in examinations were shown to use many different syntactic elements, requiring students to master a large number of different concepts to answer most of the questions posed in exams [90]. Further work showed that it is possible to construct questions that assess understanding of individual elements of programming languages [87], which may be useful for diagnostic purposes.

More recent work involves automated assessment tools such as CodeRunner [83] (developed at the University of Canterbury), and how such automated tools can play a role in higher-level courses such as computer graphics [154].

## *4.8 Academic Integrity*

Studies of academic integrity in programming courses have been a focus of research in Australia since 2001. Motivated by concerns about reports of cheating by computing students, in 2001 a large study of students' perceptions of cheating practice and acceptability found high levels of cheating and a variety of factors that motivated students to cheat or not to cheat [116, 117]. This study was the impetus for an ITiCSE working group led by the researchers that investigated staff experiences of student cheating and strategies that they used to address the problem [43]. Following this work, a series of student focus groups were conducted to get a deeper understanding of the issue and the students' opinions of possible strategies [42]. A follow-up study a decade later found that the levels of cheating had reduced and students perceived cheating to be less acceptable [114, 115].

Academic integrity in computing education has been source of much confusion and disagreement between students and academics, leading to challenges in devising a consistent and meaningful approach to academic integrity. Concerns about lack of guidance for students about what constitutes plagiarism and collusion with computer programming assignments led to an investigation comparing academic practices with regard to both essays and computing assessments. The study found considerable variation between student and staff attitudes to similar practices in the text and non-text environments, and about what was viewed as plagiarism or collusion [130]. The study, continued as an ITiCSE working group, also investigated differences between academic practice in computing education and professional practice in the computing industry, found many differences in the practices of these two groups, and what they considered acceptable [134, 141]. The findings of these studies suggested a need for academic integrity policies, procedures, and education

specific to computing, and clear guidance to students about what is acceptable and unacceptable for specific computing assessments. Follow-up work has proposed strategies for maintaining academic integrity in computing courses [119, 140].

Recently, a study of the impact of the covid-19 pandemic on assessment in computing courses found that increasing levels of academic violation were the main challenge facing academics in the move to online assessment [131], which was forced by the pandemic but appears likely to continue.

Students who are learning to program can easily become 'stuck' when their programs are faulty [39]. A study of student code indicated that errors involving identifiers and those involving types were among the most prevalent and time-consuming errors for students to overcome [37]. Studies of student difficulties with syntax [55] and logic errors [104] informed work to investigate how error messages might be improved [5, 34, 41], and how students could be more effectively taught debugging [80, 89, 151].

## *4.9 Women in Computing*

Under-representation of women in computing courses and industry has long been a topic of interest in Australasia and has been the focus of many research studies.

In 1985, prompted by concerns of low enrolments of women in computing science courses, a large Australian study investigated the issues women face in computing courses [67]. A survey of 36 Australian higher education institutions, and interviews with 19 academic staff and 86 students, found low proportions of women entering computing courses and high proportions discontinuing their courses in the early stages. Many of the problems that women experienced related to stereotyping of gender roles in computing. Another early investigation of under-representation of women in computing found that wide misunderstandings of the nature of computing careers and their suitability for women were influential in the decisions that girls make about courses and careers and the advice and encouragement they received to pursue a career in computing [143]. The factors influencing interest in computing courses and careers, particularly among young women, were explored in later studies of Australian secondary school students [76] and another of first-year university students [92], with similar findings.

The low number of women in computing courses is also a long-standing topic of interest in New Zealand. A study investigated the participation rates in five universities and polytechnics in the decade prior to 1996, outlined a set of factors deterring women from computing study, and proposed strategies for increasing the participation of women [15]. A study of the effectiveness of approaches implemented from 1991 and 1996 to reduce the withdrawal rate of women, including designing gender-neutral content, found little evidence that these changes were beneficial in raising female retention rates [113].

Identifying that one of the leading causes of gender disparity in computing is the lack of female role models and gender stereotyping in computing careers in the

media, a study examined the advertising material promoting computing degrees in Australian universities with a focus on student testimonials [75]. Findings show that there were more than twice as many male testimonials as female, and that women were disproportionately shown talking about soft skills. In the second part of the study, participants were shown videos of student testimonials. The results show that there was a positive change in female attitudes towards technical skill effectiveness after viewing the second video, which had a female student talking about soft skills. No changes were reported in the male participants after watching.

A framework for enhancing computing for social good suggested that both motivation and desirable professional skills could be developed through social good projects [56]. The value of contextualizing computing for social good has also been explored as a possible means of shifting the gender balance [68], but further study is needed to evaluate the effectiveness of this approach.

There have been many initiatives to interest girls in computing, with activities such as presentations from women in the computing industry and experience with computing technology through hands-on workshops. A study that explored the effectiveness of these programs [30] concluded that while all reported positive results, the long-term attitudinal changes had not been measured, and proposed that declining female enrolments in computing suggested that any changes may have been negligible. This work formed the foundation of the Digital Divas program for girls in early secondary school, which ran from 2008 to 2012 in ten Australian secondary schools. This outreach program aimed to engender positive perceptions of computing through a semester-long program of upbeat computing experiences and interactions with university undergraduates and industry professionals [77]. An in-depth evaluation after 4 years of the program found that the program had significantly increased student confidence but there was little evidence that it had motivated students to pursue a career in computing [78].

More recently in a 2021 study, Fletcher et al. [54] presented a longitudinal analysis on the impact of computing school outreach in Australia, focusing in particular on long-term changes in female students' attitudes and self-efficacy towards computing given different levels of intervention. The study found that regardless of the type of intervention, interest faded out within two terms, while self-efficacy, although fading, was still shown as having some residual beneficial effect. The study also found that female students benefit more from interventions, but this is potentially because male students start out at a higher level of knowledge and interest about the field.

## *4.10 Computing Curricula*

Several researchers from the Australasian community have had significant influence in the development of global computing curricula.

Software engineering practices and curriculum have also been explored critically [8, 24], and recommendations for the design of global software engineering courses have been developed [26]. The use of teams in learning software development [62, 103] is also explored in work that forces students to deal with unexpected issues [142], and the importance of preparing students for working in a global software development environment [7, 24, 25, 28]. Several authors have also contributed to our understanding of capstone projects [14, 27, 150].

New programs in cybersecurity [4], computing bootcamps [146], and other formal programs [65] developed in Australasia have been shared with the community. Alison Clear, elected as Chair of the SIGCSE Board in 2022, has a long history of contributing to the ACM computing curriculum [65, 82], more recently leading revisions to Computing Curriculum 2020 and the establishment of competencies and dispositions [21].

### *4.11 Introspection*

Introspective research focusing on the nature of the computing education community has been a feature of several research projects in Australasia. This includes the growth of computing education doctoral work [108], the challenges of computing education [53, 106], bibliometric analysis of computing education publications [98, 155], geographic diversity of the computing education community [6], the quality of citations in computing education [132], the research questions that educators want answered, and commentary on the community in the Cambridge Handbook of Computing Education Research [3]. Further, moral and ethical concerns about computing education practices have been raised through the sustained contribution of opinion pieces, columns and articles by Clear [22, 23].

In 2007 the journal Computer Science Education had a special issue devoted to computing education in Australasia. For that issue, Simon [124] examined all of the computing education publications in the prior three offerings of ACE and the NACCQ conference. Seeking to form a broad overview of these publications, and failing to find an existing classification system that would provide that overview, he devised his own classification system.

Simon then used the system as the basis for a workshop held in conjunction with ACE 2008, a workshop with the unwieldy name of Australasian Working Party On Classifying Computing Education Literature In Published Sources (AWPOCCELIPS). Along with validating the classification system by assessing its inter-rater reliability, members of that workshop applied the system to the previous 8 years of NACCQ [136], and then to all 3 years of ICER to that point [129]. The system has since been applied to a number of other bodies of work [125–128]. One of its clear findings is that all of the venues examined have shown a steady increase in the proportion of empirical research papers to experience reports and other papers.

## 5 School-Level Contribution

While most of the foregoing research has focused on tertiary-level education, Australasia has not fallen behind as computing education has moved into schools. As one early example, the CS Unplugged project [11] has had significant worldwide impact in delivery of computing principles without the use of computers, an attractive option for primary-school [45], and middle-school integration of computational thinking skills [9].

Research projects in Australia and New Zealand specifically in the K-12 space focus broadly on three main areas: outreach and diversity and inclusion, the presentation of the Australian and New Zealand digital technologies curricula, and the analysis of teacher perceptions and misconceptions about teaching digital technologies in K-12, which is then supported by ongoing professional development [148]. Moving from the local focus, recent work has been directed at worldwide implementations of computing curricula [47, 48].

### *5.1 Overview of Digital Curricula in Australia and New Zealand*

Falkner et al. [49] present the Australian digital technologies (DT) curriculum as it was introduced in 2014. The curriculum applies a systems thinking approach, with students being encouraged to understand the individual parts of a computer system with a holistic view of ethical, societal, and sustainability considerations. DT focuses on developing knowledge of digital systems, information management, and the computational thinking required to create digital solutions. As with the New Zealand curriculum, at the core of the Australian digital technologies curriculum lies the understanding of computational thinking skills. The curriculum describes learning outcomes across three broad year groupings: foundation to year 2 (ages 5–7); years 3–6 (ages 8–11); and years 7–10 (ages 12–16). Approaches to teaching vary according to these year groupings, starting from computational thinking in F–2, to wider understanding of the impact of technology in years 3–6, and deeper understanding of ethical and societal considerations in years 7–10.

The revision of the digital technology curriculum in NZ schools provided a rich opportunity for computing education researchers, including the logistical challenges of the curriculum [10, 144], supporting teachers to deliver CS standards [69, 70], and the impact on students [111, 112].

Crow et al. [31] discuss the newly introduced curriculum areas of digital technologies and *hangarau matihiko*, which were added to the New Zealand school curricula in 2017. The paper introduces the New Zealand digital technologies curriculum, covering content related to the fundamental principles of computer science and developing digital technologies. The curriculum is delivered through an English and a Maori medium. In the English medium curriculum, digital

technologies sits within the technology learning area, and the new content is split into two new technology learning areas: computational thinking, and designing and developing digital outcomes. In *te marautanga o Aotearoa*, *hangarau matihiko* (roughly translated to digital technology) is split into *te tupuranga whakaaro rorohiko* (computational thinking) and *te tupuranga tangata me te rorohiko* (the growth/future of people and computers). It is important to note that while the two curricula are similar, they are not translations of each other.

The new curriculum content describes significant learning steps that students take as they develop their knowledge and understanding of digital technology concepts. The learning steps are gradual, with early years focused on computational thinking delivered through non-computerised tasks, and moving on in later years to a specific programming language: an imperative, Turing-complete programming language that contains sequence, selection, iteration, inputs, outputs, and logical operators. An initial identified gap is the lack of detailed lesson plans and detailed overviews of how specific topics fit within units, and a lack of good examples of the integration of the English digital technologies curriculum with the Maori *hangarau matihiko* curriculum.

## *5.2 Teacher Engagement and Professional Education*

The introduction of the digital technologies curricula in Australia and New Zealand placed significant focus on teacher professional development [46, 97, 147, 149]. An early study by Duncan et al. [46] analyses detailed feedback about the new digital technologies curriculum from 13 teachers without any prior programming experience. The study identified that teachers were very capable of delivering the curriculum in engaging ways, and that professional development can help address misconceptions generated by the newly introduced material.

A large number of studies [97] focus on teacher misconceptions with respect to teaching either broadly within the digital technologies curricula [97], or related to specific concepts such as computational thinking. Difficulties were found with respect to computer jargon [97], teacher confidence [147], teacher self-esteem and assessment approaches [149], and teacher understanding of computational thinking [46, 97].

The most scalable approach to teacher professional development is through the use of MOOCs, and significant research is dedicated to presenting the design and development of K-6 MOOCs [50, 148] and the analysis of their effectiveness in building teacher confidence and self-efficacy, and in developing digital technologies communities of practice [51, 52].

## 6 Conclusions

The Australasian computing education community has made several substantial contributions, including the development of BlueJ, Parson's Problems, PeerWise and other Contributing Student Pegagogy tools, CSUnplugged, and significant work on introductory programming, global software engineering, introspective and critical research, women in computing and computing in schools. The computing education contributions of New Zealand and Australia are substantial, and out of proportion to their populations. In our conclusions, we consider why this might be so.

One possible reason is that smaller scale of the Australasian Computing community provides the opportunity for individuals and their initiatives to have greater influence than in bigger communities. The impact of the Australasian Computing Education conference in maintaining collaborations between colleagues is perceived to be high, and sustained collaborative projects such as BRACE, BRACElet, and BABELnot have undeniably promoted the development of computing education research, and researchers.

The benefits of close geographic proximity and regular collaboration is apparent within the Australasian community and may partially explain the sustained high level of contribution to Computing Education Research. For example, the critical mass of computing education researchers present in close geographic proximity in Auckland has resulted in high levels of research output, with Auckland University of Technology and the University of Auckland both appearing in the top 40 most prolific institutions contributing to SIGCSE publications [1].

Further, in recent decades, many institutions in both countries have recognized the value of computing education research and treated such work equally to disciplinary-based research for promotions and career purposes. Additionally, they have been willing and able to provide a reasonable quantum of funding for academic activities, thus facilitating international travel and collaboration, without which Australasia would be rather more insular, albeit with multiple islands. This level of institutional support has enabled computing education research to grow.

However, increasing awareness of the human impact on the world's climate has led to a growing concern about the carbon footprint of travel. This might lead in turn to a reduction in international conference travel: perhaps these countries will once more become somewhat insular, with their researchers dependent on virtual collaboration, and conferences continuing to permit remote attendance and presentation.

## References

1. ACM Digital Library SIGCSE Affiliations. URL https://dl.acm.org/sig/sigcse/affiliations?startPage=0&pageSize=50

2. ACM Digital Library SIGCSE Authors. URL https://dl.acm.org/sig/sigcse/authors?startPage=0&pageSize=50
3. The Cambridge Handbook of Computing Education Research. Cambridge Handbooks in Psychology. Cambridge University Press (2019). DOI https://doi.org/10.1017/9781108654555
4. Asghar, M.R., Luxton-Reilly, A.: A Case Study of a Cybersecurity Programme: Curriculum Design, Resource Management, and Reflections, p. 16–22. SIGCSE 2020 (2020)
5. Becker, B.A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D.J., Harrington, B., Kamil, A., Karkare, A., McDonald, C., Osera, P.M., Pearce, J.L., Prather, J.: Compiler error messages considered unhelpful: the landscape of text-based programming error message research. In: ITiCSE 2019 Working Group Reports, ITiCSE-WGR 2019, p. 177–210 (2019). DOI https://doi.org/10.1145/3344429.3372508
6. Becker, B.A., Settle, A., Luxton-Reilly, A., Morrison, B.B., Laxer, C.: Expanding opportunities: assessing and addressing geographic diversity at the SIGCSE Technical Symposium. In: 52nd Technical Symposium on Computer Science Education, SIGCSE 2021, p. 281–287 (2021). DOI https://doi.org/10.1145/3408877.3432448
7. Beecham, S., Clear, T., Lal, R., Noll, J.: Do scaling agile frameworks address global software development risks? An empirical study (2020). DOI https://doi.org/10.48550/ARXIV.2009.08193
8. Beecham, S., Noll, J., Clear, T.: Do we teach the right thing? A comparison of GSE education and practice. In: 12th International Conference on Global Software Engineering (ICGSE), pp. 11–20 (2017). DOI https://doi.org/10.1109/ICGSE.2017.8
9. Bell, T.: CS unplugged or coding classes? Commun. ACM **64**(5), 25–27 (2021). URL https://doi.org/10.1145/3457195
10. Bell, T., Andreae, P., Robins, A.: A case study of the introduction of computer science in NZ schools. ACM Trans. Comput. Educ. **14**(2) (2014). DOI https://doi.org/10.1145/2602485
11. Bell, T., Witten, I., Fellows, M.: CS unplugged: an enrichment and extension programme for primary-aged students (2015)
12. Biddle, R., Tempero, E.: Explaining inheritance: A code reusability perspective. SIGCSE Bulletin **28**(1), 217–221 (1996). DOI https://doi.org/10.1145/236462.236543
13. Biddle, R., Tempero, E.: Java pitfalls for beginners. SIGCSE Bulletin **30**(2), 48–52 (1998). DOI https://doi.org/10.1145/292422.292441
14. Bridgeman, N.: Capstone projects: An NACCQ retrospective. In: Annual Conference of the National Advisory Committe of Computing Qualifications (NACCQ). Citeseer (2008)
15. Brown, J., Andreae, P., Biddle, R., Tempero, E.: Women in introductory computer science: Experience at Victoria University of Wellington. In: 28th Technical Symposium on Computer Science Education, SIGCSE 1997, p. 111–115 (1997). DOI https://doi.org/10.1145/268084.268128
16. Butler, M., Morgan, M., Sheard, J., Simon, Falkner, K., Weerasinghe, A.: Initiatives to increase engagement in first-year ICT. In: 20th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2015, pp. 308–313 (2015). URL http://doi.acm.org/10.1145/2729094.2742629
17. Butler, M., Sheard, J., Morgan, M., Falkner, K., Simon, Weerasinghe, A.: Understanding the teaching context of first-year ICT education in Australia. In: 17th Australasian Computing Education Conference, ACE 2015, pp. 101–109 (2015). URL http://crpit.com/confpapers/CRPITV160Butler.pdf
18. Carbone, A., Hurst, J., Mitchell, I., Gunstone, D.: Principles for designing programming exercises to minimise poor learning behaviours in students. In: Proceedings of the Australasian conference on Computing education, pp. 26–33 (2000)
19. Carbone, A., Hurst, J., Mitchell, I., Gunstone, D.: An exploration of internal factors influencing student learning of programming. In: Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95, pp. 25–34 (2009)
20. Carpenter, B.E., Jenkins, P.C., Pearson, L.W., Thomas, L.K.: MUSIC: a simulated computer for teaching purposes. SIGCSE Bulletin **9**(4), 70–76 (1977). DOI https://doi.org/10.1145/382181.382599

21. Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., Lunt, B., Maiorana, F., Pears, A., Pitt, F., Riedesel, C., Szynkiewicz, J.: Designing computer science competency statements: A process and curriculum model for the 21st century. In: ITiCSE 2020 Working Group Reports, ITiCSE-WGR 2020, p. 211–246 (2020). DOI https://doi.org/10.1145/3437800.3439208
22. Clear, T.: 'social' or 'anti-social' software: Content production in web 2.0—who benefits? ACM Inroads **4**(4), 12–13 (2013). URL https://doi.org/10.1145/2537753.2537757
23. Clear, T.: Thinking issues: What's driving Uber? Values in computing and the 'sharing economy'. ACM Inroads **8**(4), 38–40 (2017)
24. Clear, T.: The arbitrary nature of computing curricula. XRDS **25**(1), 56–59 (2018). DOI https://doi.org/10.1145/3265905
25. Clear, T., Beecham, S.: Global software engineering education practice continuum. Special issue of the ACM Transactions on Computing Education. ACM Trans. Comput. Educ. **19**(2) (2019). DOI https://doi.org/10.1145/3294011
26. Clear, T., Beecham, S., Barr, J., Daniels, M., McDermott, R., Oudshoorn, M., Savickaite, A., Noll, J.: Challenges and recommendations for the design and conduct of global software engineering courses: A systematic review. In: ITiCSE 2015 Working Group Reports, ITICSE-WGR 2015, p. 1–39 (2015). DOI https://doi.org/10.1145/2858796.2858797
27. Clear, T., Goldweber, M., Young, F.H., Leidig, P.M., Scott, K.: Resources for instructors of capstone courses in computing. In: Working group reports from ITiCSE on Innovation and technology in computer science education, pp. 93–113 (2001)
28. Clear, T., Kassabova, D.: A course in collaborative computing: collaborative learning and research with a global perspective. In: Proceedings of the 39th SIGCSE technical symposium on Computer science education, pp. 63–67 (2008)
29. Clear, T., Whalley, J., Robbins, P., Philpott, A., Eckerdal, A., Laakso, M., Lister, R.: Report on the final BRACElet workshop. Journal of Applied Computing and Information Technology **15**(1) (2011). URL https://www.citrenz.ac.nz/jacit/JACIT1501/2011Clear_BRACElet.html
30. Craig, A., Lang, C., Fisher, J.: Twenty years of Girls into Computing days: has it been worth the effort? Journal of Information Technology Education: Research **7**(1), 339–353 (2008)
31. Crow, T., Luxton-Reilly, A., Wünsche, B.C., Denny, P.: Resources and support for the implementation of digital technologies in New Zealand schools. In: 21st Australasian Computing Education Conference, pp. 69–78 (2019)
32. Denny, P.: The effect of virtual achievements on student engagement. In: SIGCHI Conference on Human Factors in Computing Systems, CHI 2013, p. 763–772 (2013). DOI https://doi.org/10.1145/2470654.2470763
33. Denny, P., Hamer, J., Luxton-Reilly, A., Purchase, H.: Peerwise: students sharing their multiple choice questions. In: Fourth International Computing Education Research Workshop, ICER 2008, p. 51–58 (2008). DOI https://doi.org/10.1145/1404520.1404526
34. Denny, P., Luxton-Reilly, A., Carpenter, D.: Enhancing syntax error messages appears ineffectual. In: 19th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2014, p. 273–278 (2014). DOI https://doi.org/10.1145/2591708.2591748
35. Denny, P., Luxton-Reilly, A., Craig, M., Petersen, A.: Improving complex task performance using a sequence of simple practice tasks. In: 23rd Conference on Innovation & Technology in Computer Science Education, ITiCSE 2018, p. 4–9 (2018). DOI https://doi.org/10.1145/3197091.3197141
36. Denny, P., Luxton-Reilly, A., Simon, B.: Evaluating a new exam question: Parsons problems. In: Fourth International Computing Education Research Workshop, ICER 2008, p. 113–124 (2008). DOI https://doi.org/10.1145/1404520.1404532
37. Denny, P., Luxton-Reilly, A., Tempero, E.: All syntax errors are not equal. In: 17th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2012, p. 75–80 (2012). DOI https://doi.org/10.1145/2325296.2325318
38. Denny, P., Luxton-Reilly, A., Tempero, E., Hendrickx, J.: CodeWrite: Supporting student-driven practice of Java. In: 42nd Technical Symposium on Computer Science Education, SIGCSE 2011, p. 471–476 (2011). DOI https://doi.org/10.1145/1953163.1953299

39. Denny, P., Luxton-Reilly, A., Tempero, E., Hendrickx, J.: Understanding the syntax barrier for novices. In: 16th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2011, p. 208–212 (2011). DOI https://doi.org/10.1145/1999747.1999807
40. Denny, P., McDonald, F., Empson, R., Kelly, P., Petersen, A.: Empirical support for a causal relationship between gamification and learning outcomes. In: 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, p. 1–13 (2018). DOI https://doi.org/10.1145/3173574.3173885
41. Denny, P., Prather, J., Becker, B.A.: Error message readability and novice debugging performance. In: 25th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2020, p. 480–486 (2020). DOI https://doi.org/10.1145/3341525.3387384
42. Dick, M., Purcell, W., Sheard, J.I., Hasen, M.: What to do about cheating? Using the student perspective to develop curriculum to address the problem. In: Asia-Pacific Educational Integrity Conference (2005)
43. Dick, M., Sheard, J., Bareiss, C., Carter, J., Joyce, D., Harding, T., Laxer, C.: Addressing student cheating: definitions and solutions. SIGCSE Bulletin **35**(2), 172–184 (2002)
44. Du, Y., Luxton-Reilly, A., Denny, P.: A review of research on Parsons problems. In: 22nd Australasian Computing Education Conference, ACE 2020, p. 195–202 (2020). DOI https://doi.org/10.1145/3373165.3373187
45. Duncan, C., Bell, T.: A pilot computer science and programming course for primary school students. In: Proceedings of the Workshop in Primary and Secondary Computing Education, WiPSCE '15, p. 39–48. Association for Computing Machinery, New York, NY, USA (2015). URL https://doi.org/10.1145/2818314.2818328
46. Duncan, C., Bell, T., Atlas, J.: What do the teachers think? Introducing computational thinking in the primary school curriculum. In: 19th Australasian Computing Education Conference, pp. 65–74 (2017)
47. Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F., McGill, M.M., Quille, K.: An international comparison of K-12 computer science education intended and enacted curricula. In: 19th Koli Calling International Conference on Computing Education Research, pp. 1–10 (2019)
48. Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F., McGill, M.M., Quille, K.: An international study piloting the measuring teacher enacted computing curriculum (metrecc) instrument. In: ITiCSE 2019 Working Group Reports, pp. 111–142 (2019)
49. Falkner, K., Vivian, R., Falkner, N.: The Australian digital technologies curriculum: challenge and opportunity. In: 16th Australasian Computing Education Conference, ACE 2014, pp. 3–12 (2014)
50. Falkner, K., Vivian, R., Falkner, N.: Teaching computational thinking in K-6: The CSER digital technologies MOOC. In: 17th Australasian Computing Education Conference, *ACE 2015*, vol. 30 (2015)
51. Falkner, K., Vivian, R., Falkner, N., Williams, S.A.: Reflecting on three offerings of a community-centric MOOC for K-6 computer science teachers. In: 48th SIGCSE Technical Symposium on Computer Science Education, pp. 195–200 (2017)
52. Falkner, K., Vivian, R., Williams, S.A.: An ecosystem approach to teacher professional development within computer science. Computer Science Education **28**(4), 303–344 (2018)
53. Fincher, S., Lister, R., Clear, T., Robins, A., Tenenberg, J., Petre, M.: Multi-institutional, multi-national studies in CSEd research: some design considerations and trade-offs. In: First International Computing Education Research Workshop, ICER 2005, p. 111–121 (2005). DOI https://doi.org/10.1145/1089786.1089797
54. Fletcher, A., Mason, R., Cooper, G.: Helping students get IT: investigating the longitudinal impacts of IT school outreach in Australia. In: 23rd Australasian Computing Education Conference, ACE 2021, pp. 115–124 (2021)
55. Garner, S., Haden, P., Robins, A.: My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems. In: Seventh Australasian Computing Education Conference, ACE 2005, p. 173–180 (2005)

56. Goldweber, M., Barr, J., Clear, T., Davoli, R., Mann, S., Patitsas, E., Portnoff, S.: A framework for enhancing the social good in computing education: A values approach. ACM Inroads **4**(1), 58–79 (2013). URL https://doi.org/10.1145/2432596.2432616
57. Hagan, D., Markham, S.: Does it help to have some programming experience before beginning a computing degree program? In: Fifth Conference on Innovation & Technology in Computer Science Education, ITiCSE 2000, pp. 25–28 (2000)
58. Hamer, J., Cutts, Q., Jackova, J., Luxton-Reilly, A., McCartney, R., Purchase, H., Riedesel, C., Saeli, M., Sanders, K., Sheard, J.: Contributing student pedagogy. SIGCSE Bulletin **40**(4), 194–212 (2008). DOI https://doi.org/10.1145/1473195.1473242
59. Hamer, J., Kell, C., Spence, F.: Peer assessment using Aropä. In: Ninth Australasian Computing Education Conference, ACE 2007, p. 43–54 (2007)
60. Hamer, J., Purchase, H.C., Denny, P., Luxton-Reilly, A.: Quality of peer assessment in CS1. In: Fifth International Computing Education Research Workshop, ICER 2009, p. 27–36 (2009). DOI https://doi.org/10.1145/1584322.1584327
61. Hamer, J., Purchase, H.C., Luxton-Reilly, A., Sheard, J.: Tools for "contributing student learning". In: ITiCSE 2010 Working Group Reports, ITiCSE-WGR 2010, p. 1–14 (2010). DOI https://doi.org/10.1145/1971681.1971683
62. Hogan, J.M., Thomas, R.: Developing the software engineering team. In: Seventh Australasian Computing Education Conference, pp. 203–210 (2005)
63. Indriasari, T.D., Luxton-Reilly, A., Denny, P.: A review of peer code review in higher education. ACM Trans. Comput. Educ. **20**(3) (2020). DOI https://doi.org/10.1145/3403935
64. Indriasari, T.D., Luxton-Reilly, A., Denny, P.: Investigating accuracy and perceived value of feedback in peer code review using gamification. In: 26th Conference on Innovation & Technology in Computer Science Education Vol 1, p. 199–205 (2021)
65. Joyce, D., Young, A.: Developing and implementing a professional doctorate in computing. In: Sixth Australasian Computing Education Conference, ACE 2004, p. 145–149 (2004)
66. Kavanagh, S., Luxton-Reilly, A.: Rubrics used in peer assessment. In: Australasian Computer Science Week Multiconference, ACSW 2016 (2016). DOI https://doi.org/10.1145/2843043.2843347
67. Kay, J., Lublin, J., Poiner, G., Prosser, M.: Not even well begun: women in computing courses. Higher Education **18**(5), 511–527 (1989)
68. Khan, N.Z., Luxton-Reilly, A.: Is computing for social good the solution to closing the gender gap in computer science? In: Australasian Computer Science Week Multiconference, ACSW 2016 (2016). DOI https://doi.org/10.1145/2843043.2843069
69. Kirk, D., Crow, T., Luxton-Reilly, A., Tempero, E.: Mind the gap: searching for clarity in NCEA, p. 192–198 (2021)
70. Kirk, D., Crow, T., Luxton-Reilly, A., Tempero, E.: Teaching code quality in high school programming courses - understanding teachers' needs. In: Australasian Computing Education Conference, ACE 2022, p. 36–45 (2022). DOI https://doi.org/10.1145/3511861.3511866
71. Kölling, M.: The problem of teaching object-oriented programming, part 1: languages. Journal of Object-Oriented Programming **11**(8), 8–15 (1999)
72. Kölling, M.: The problem of teaching object-oriented programming, part 2: environments. Journal of Object-Oriented Programming pp. 6–12 (1999)
73. Kölling, M., Quig, B., Patterson, A., Rosenberg, J.: The BlueJ system and its pedagogy. Computer Science Education **13**(4), 249–268 (2003). DOI https://doi.org/10.1076/csed.13.4.249.17496
74. Kölling, M., Rosenberg, J.: Guidelines for teaching object orientation with Java. SIGCSE Bulletin **33**(3), 33–36 (2001)
75. Lamers, D., Mason, R.: Advertising CS/IT degrees to female students in Australia. In: 20th Australasian Computing Education Conference, ACE 2018, pp. 1–8 (2018)
76. Lang, C.: Sequential attrition of secondary school student interest in IT courses and careers. Information Technology & People (2012)
77. Lang, C., Craig, A., Fisher, J., Forgasz, H.: Creating digital divas: scaffolding perception change through secondary school and university alliances. In: 15th Conference on Innovation

& Technology in Computer Science Education, ITiCSE 2010, p. 38–42 (2010). DOI https://doi.org/10.1145/1822090.1822103
78. Lang, C., Fisher, J., Craig, A., Forgasz, H.: Outreach programmes to attract girls into computing: how the best laid plans can sometimes fail. Computer Science Education **25**(3), 257–275 (2015)
79. Leinonen, J., Denny, P., Whalley, J.: A comparison of immediate and scheduled feedback in introductory programming projects. In: 53rd Technical Symposium on Computer Science Education Vol 1, SIGCSE 2022, p. 885–891 (2022). DOI https://doi.org/10.1145/3478431.3499372
80. Li, C., Chan, E., Denny, P., Luxton-Reilly, A., Tempero, E.: Towards a framework for teaching debugging. In: 21st Australasian Computing Education Conference, ACE 2019, p. 79–86 (2019). DOI https://doi.org/10.1145/3286960.3286970
81. Lister, R.: A clandestine religious meeting. SIGCSE Bulletin **36**(4), 16–17 (2004). URL https://doi.org/10.1145/1041624.1041636
82. Little, J.C., Granger, M., Adams, E.S., Holvikivi, J., Lippert, S.K., Walker, H.M., Young, A.: Integrating cultural issues into the computer and information technology curriculum. In: ITiCSE 2000 Working Group Reports, ITiCSE-WGR 2000, p. 136–154 (2001). DOI https://doi.org/10.1145/571968.571975
83. Lobb, R., Harlow, J.: Coderunner: a tool for assessing computer programming skills. ACM Inroads **7**(1), 47–51 (2016). DOI https://doi.org/10.1145/2810041
84. Lopez, M., Whalley, J., Robbins, P., Lister, R.: Relationships between reading, tracing and writing skills in introductory programming. In: Fourth International Computing Education Research Workshop, ICER 2008, p. 101–112 (2008). DOI https://doi.org/10.1145/1404520.1404531
85. Luxton-Reilly, A.: A systematic review of tools that support peer assessment. Computer Science Education **19**(4), 209–232 (2009). DOI https://doi.org/10.1080/08993400903384844
86. Luxton-Reilly, A.: Learning to program is easy. In: 21st Conference on Innovation & Technology in Computer Science Education, ITiCSE 2016, p. 284–289 (2016). DOI https://doi.org/10.1145/2899415.2899432
87. Luxton-Reilly, A., Becker, B.A., Cao, Y., McDermott, R., Mirolo, C., Mühling, A., Petersen, A., Sanders, K., Simon, Whalley, J.: Developing assessments to determine mastery of programming fundamentals. In: 22nd Conference on Innovation & Technology in Computer Science Education, ITiCSE 2017, p. 388 (2017). DOI https://doi.org/10.1145/3059009.3081327
88. Luxton-Reilly, A., Bertinshaw, D., Denny, P., Plimmer, B., Sheehan, R.: The impact of question generation activities on performance. In: 43rd Technical Symposium on Computer Science Education, SIGCSE 2012, p. 391–396 (2012). DOI https://doi.org/10.1145/2157136.2157250
89. Luxton-Reilly, A., McMillan, E., Stevenson, E., Tempero, E., Denny, P.: Ladebug: an online tool to help novice programmers improve their debugging skills. In: 23rd Conference on Innovation & Technology in Computer Science Education, ITiCSE 2018, p. 159–164 (2018). DOI https://doi.org/10.1145/3197091.3197098
90. Luxton-Reilly, A., Petersen, A.: The compound nature of novice programming assessments. In: 19th Australasian Computing Education Conference, ACE 2017, p. 26–35 (2017). DOI https://doi.org/10.1145/3013499.3013500
91. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory programming: a systematic literature review. In: ITiCSE 2018 Working Group Reports, ITiCSE-WGR 2018, p. 55–106 (2018). DOI https://doi.org/10.1145/3293881.3295779
92. McLachlan, C., Craig, A., Coldwell, J.: Student perceptions of ICT: a gendered analysis. In: 12th Australasian Computing Education Conference, ACE 2010 (2010)
93. Mendes, E., Al-Fakhri, L.B., Luxton-Reilly, A.: Investigating pair-programming in a 2nd-year software development and design computer science course. In: 10th Conference on

Innovation & Technology in Computer Science Education, ITiCSE 2005, p. 296–300 (2005). DOI https://doi.org/10.1145/1067445.1067526
94. Moon, B.A.M.: The challenge of computer science in New Zealand. SIGCSE Bulletin **2**(3), 61–68 (1970). DOI https://doi.org/10.1145/873641.873653
95. Morgan, M., Sheard, J., Butler, M., Falkner, K., Simon, Weerasinghe, A.: Teaching in first-year ICT education in Australia: research and practice. In: 17th Australasian Computing Education Conference, ACE 2015, pp. 81–90 (2015). URL http://crpit.com/confpapers/CRPITV160Morgan.pdf
96. Mugridge, W.B., Hosking, J.G.: A method for introducing schemas. SIGCSE Bulletin **17**(4), 76–82 (1985). DOI https://doi.org/10.1145/989369.989381
97. Munasinghe, B., Bell, T., Robins, A.: Teachers' understanding of technical terms in a computational thinking curriculum. In: 23rd Australasian Computing Education Conference, ACE 2021, pp. 106–114 (2021)
98. Papamitsiou, Z., Giannakos, M., Simon, Luxton-Reilly, A.: Computing education research landscape through an analysis of keywords. In: 16th International Computing Education Research Conference, ICER 2020, p. 102–112 (2020). DOI https://doi.org/10.1145/3372782.3406276
99. Parsons, D., Haden, P.: Parson's programming puzzles: a fun and effective learning tool for first programming courses. In: Eighth Australasian Computing Education Conference, ACE 2006, p. 157–163 (2006)
100. Picardo, V., Denny, P., Luxton-Reilly, A.: Lecture recordings, viewing habits, and performance in an introductory programming course. In: 23rd Australasian Computing Education Conference, ACE 2021, p. 73–79 (2021). DOI https://doi.org/10.1145/3441636.3442307
101. Prather, J., Becker, B.A., Craig, M., Denny, P., Loksa, D., Margulieux, L.: What do we think we think we are doing? Metacognition and self-regulation in programming. In: 16th International Computing Education Research Conference, ICER 2020, p. 2–13 (2020). DOI https://doi.org/10.1145/3372782.3406263
102. Prather, J., Pettit, R., Becker, B.A., Denny, P., Loksa, D., Peters, A., Albrecht, Z., Masci, K.: First things first: providing metacognitive scaffolding for interpreting problem prompts. In: 50th Technical Symposium on Computer Science Education, SIGCSE 2019, p. 531–537 (2019). DOI https://doi.org/10.1145/3287324.3287374
103. Richards, D.: Designing project-based courses with a focus on group formation and assessment. ACM transactions on Computing Education (TOCE) **9**(1), 1–40 (2009)
104. Rigby, L., Denny, P., Luxton-Reilly, A.: A miss is as good as a mile: off-by-one errors and arrays in an introductory programming course. In: 22nd Australasian Computing Education Conference, ACE 2020, p. 31–38 (2020). DOI https://doi.org/10.1145/3373165.3373169
105. Robins, A.: Learning edge momentum: a new account of outcomes in CS1. Computer Science Education **20**(1), 37–71 (2010)
106. Robins, A.: The ongoing challenges of computer science education research. Computer Science Education **25**(2), 115–119 (2015). DOI https://doi.org/10.1080/08993408.2015.1034350
107. Robins, A., Rountree, J., Rountree, N.: Learning and teaching programming: A review and discussion. Computer Science Education **13**(2), 137–172 (2003). DOI https://doi.org/10.1076/csed.13.2.137.14200
108. Robins, A., Shapiro, R.B.: The growth of computing education doctoral research. SIGCSE Bulletin **48**(4), 3 (2016). DOI https://doi.org/10.1145/3015259.3015261
109. Rosenberg, J.: Foreword. In: First Australasian Conference on Computer Science Education, ACSE 1996, p. iii (1996)
110. Rountree, N., Rountree, J., Robins, A., Hannah, R.: Interacting factors that predict success and failure in a CS1 course. In: ITiCSE 2004 Working Group Reports, ITiCSE-WGR 2004, p. 101–104 (2004). DOI https://doi.org/10.1145/1044550.1041669
111. Samarasekara, C.K., Ott, C., Robins, A.: A decade of CS education in New Zealand's high schools. Where are we at? In: 16th Workshop in Primary and Secondary Computing Education, WIPSCE 2021 (2021)

112. Samarasekara, C.K., Ott, C., Robins, A.: Barriers to New Zealand high school CS education - learners' perspectives. In: 53rd Technical Symposium on Computer Science Education Vol 1, SIGCSE 2022, p. 927–933 (2022). DOI https://doi.org/10.1145/3478431.3499344
113. Selby, L., Ryba, K., Young, A.: Women in computing: what does the data show? SIGCSE Bulletin **30**(4), 62–67 (1998). DOI https://doi.org/10.1145/306286.306318
114. Sheard, J., Dick, M.: Computing student practices of cheating and plagiarism: a decade of change. In: 16th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2011, pp. 233–237 (2011)
115. Sheard, J., Dick, M.: Directions and dimensions in managing cheating and plagiarism of IT students. In: 14th Australasian Computing Education Conference, ACE 2012, pp. 177–186 (2012)
116. Sheard, J., Dick, M., Markham, S., Macdonald, I., Walsh, M.: Cheating and plagiarism: perceptions and practices of first year IT students. In: Seventh Conference on Innovation & Technology in Computer science Education, ITiCSE 2002, pp. 183–187 (2002)
117. Sheard, J., Markham, S., Dick, M.: Investigating differences in cheating behaviours of IT undergraduate and graduate students: the maturity and motivation factors. Higher Education Research & Development **22**(1), 91–108 (2003)
118. Sheard, J., Morgan, M., Butler, M., Falkner, K., Simon, Weerasinghe, A.: Assessment in first-year ICT education in Australia: research and practice. In: 17th Australasian Computing Education Conference, ACE 2015, pp. 91–99 (2015). URL http://crpit.com/confpapers/CRPITV160Sheard.pdf
119. Sheard, J., Simon, Butler, M., Falkner, K., Morgan, M., Weerasinghe, A.: Strategies for maintaining academic integrity in first-year computing courses. In: 22nd Conference on Innovation & Technology in Computer Science Education, ITiCSE 2017, pp. 244–249 (2017). DOI https://doi.org/10.1145/3059009.3059064
120. Sheard, J., Simon, Carbone, A., Chinn, D., Clear, T., Corney, M., D'Souza, D., Fenwick, J., Harland, J., Laakso, M.J., Teague, D.: How difficult are exams? A framework for assessing the complexity of introductory programming exams. In: 15th Australasian Computing Education Conference, ACE 2013, p. 145–154 (2013)
121. Sheard, J., Simon, Carbone, A., Chinn, D., Laakso, M.J., Clear, T., De Raadt, M., D'Souza, D., Harland, J., Lister, R., Philpott, A., et al.: Exploring programming assessment instruments: a classification scheme for examination questions. In: Seventh International Computing Education Research Workshop, ICER 2011, pp. 33–38 (2011)
122. Sheard, J., Simon, Carbone, A., D'Souza, D., Hamilton, M.: Assessment of programming: pedagogical foundations of exams. In: 18th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2013, pp. 141–146 (2013)
123. Sheard, J., Simon, Dermoudy, J., D'Souza, D., Hu, M., Parsons, D.: Benchmarking a set of exam questions for introductory programming. In: 16th Australasian Computing Education Conference, ACE 2014, pp. 113–121 (2014)
124. Simon: A classification of recent Australasian computing education publications. Computer Science Education **17**(3), 155–169 (2007). DOI https://doi.org/10.1080/08993400701538021
125. Simon: The Koli Calling community. In: 16th Koli Calling International Conference on Computing Education Research, Koli Calling 2016, p. 101–109 (2016). DOI https://doi.org/10.1145/2999541.2999562
126. Simon: A picture of the growing ICER community. ICER 2016, p. 153–159 (2016). DOI https://doi.org/10.1145/2960310.2960323
127. Simon: Twenty-four years of ITiCSE authors. In: 25th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2020, p. 205–211 (2020). DOI https://doi.org/10.1145/3341525.3387387
128. Simon: Twenty-two years of ACE. In: 22nd Australasian Computing Education Conference, ACE 2020, p. 203–210 (2020). DOI https://doi.org/10.1145/3373165.3373188
129. Simon, Carbone, A., de Raadt, M., Lister, R., Hamilton, M., Sheard, J.: Classifying computing education papers: process and results. In: Fourth International Computing Education Research Workshop, ICER 2008, pp. 161–172 (2008). DOI https://doi.org/10.1145/1404520.1404536

130. Simon, Cook, B., Sheard, J., Carbone, A., Johnson, C.: Academic integrity perceptions regarding computing assessments and essays. In: 10th International Computing Education Research Conference, ICER 2014, pp. 107–114 (2014)
131. Simon, Jha, M., Leemans, S.J., Berretta, R., Bilgin, A.A., Jayarathna, L., Sheard, J.: Online assessment and COVID: opportunities and challenges. In: 24th Australasian Computing Education Conference, ACE 2022, pp. 14–18 (2022)
132. Simon, Luxton-Reilly, A., Adelakun-Adeyemo, O.: Confirmation bias and other flaws in citing pass rate studies. In: 26th Conference on Innovation & Technology in Computer Science Education Vol 1, ITiCSE 2021, p. 575–581 (2021)
133. Simon, Luxton-Reilly, A., Ajanovski, V.V., Fouh, E., Gonsalvez, C., Leinonen, J., Parkinson, J., Poole, M., Thota, N.: Pass rates in introductory programming and in other STEM disciplines. In: ITiCSE 2019 Working Group Reports, ITiCSE-WGR 2019, p. 53–71 (2019). DOI https://doi.org/10.1145/3344429.3372502
134. Simon, Sheard, J.: Academic integrity and professional integrity in computing education. In: 20th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2015, pp. 237–241 (2015)
135. Simon, Sheard, J., Carbone, A., Chinn, D., Laakso, M.J., Clear, T., de Raadt, M., D'Souza, D., Lister, R., Philpott, A., Skene, J., Warburton, G.: Introductory programming: examining the exams. In: 14th Australasian Computing Education Conference, ACE 2012, pp. 61–70 (2012)
136. Simon, Sheard, J., Carbone, A., de Raadt, M., Hamilton, M., Lister, R., Thompson, E.: Eight years of computing education papers at NACCQ. National Advisory Committee on Computing Qualifications pp. 101–107 (2008)
137. Simon, Sheard, J., D'Souza, D., Klemperer, P., Porter, L., Sorva, J., Stegeman, M., Zingaro, D.: Benchmarking introductory programming exams: how and why. In: 21st Conference on Innovation & Technology in Computer Science Education, ITiCSE 2016, pp. 154–159 (2016)
138. Simon, Sheard, J., D'Souza, D., Klemperer, P., Porter, L., Sorva, J., Stegeman, M., Zingaro, D.: Benchmarking introductory programming exams: some preliminary results. In: 12th International Computing Education Research Conference, ICER 2016, pp. 103–111 (2016)
139. Simon, Sheard, J., D'Souza, D., Lopez, M., Luxton-Reilly, A., Putro, I.H., Robbins, P., Teague, D., Whalley, J.: How (not) to write an introductory programming exam. In: 17th Australasian Computing Education Conference, ACE 2015, pp. 137–146 (2015). URL http://crpit.com/confpapers/CRPITV160Simon.pdf
140. Simon, Sheard, J., Morgan, M., Petersen, A., Settle, A., Sinclair, J.: Informing students about academic integrity in programming. In: 20th Australasian Computing Education Conference, ACE 2018, pp. 113–122 (2018)
141. Simon, Sheard, J., Morgan, M., Petersen, A., Settle, A., Sinclair, J., Cross, G., Riedesel, C.: Negotiating the maze of academic integrity in computing education. In: ITiCSE 2016 Working Group Reports, ITiCSE-WGR 2016, pp. 57–80 (2016)
142. Smith, L., Mann, S., Buissink, N.: Crashing a bus full of empowered software engineering students. New Zealand Journal of Applied Computing and Information Technology **5**, 69–74 (2001)
143. Teague, J.: Women in computing: what brings them to it, what keeps them in it? SIGCSE Bulletin **34**(2), 147–158 (2002)
144. Thompson, D., Bell, T., Andreae, P., Robins, A.: The role of teachers in implementing curriculum changes. In: Proceeding of the 44th Technical Symposium on Computer Science Education, SIGCSE 2013, p. 245–250 (2013). DOI https://doi.org/10.1145/2445196.2445272
145. Thompson, E., Luxton-Reilly, A., Whalley, J.L., Hu, M., Robbins, P.: Bloom's taxonomy for CS assessment. In: 10th Australasian Computing Education Conference, ACE 2008, p. 155–161 (2008)
146. Tu, Y.C., Dobbie, G., Warren, I., Meads, A., Grout, C.: An experience report on a boot-camp style programming course. In: 49th Technical Symposium on Computer Science Education, SIGCSE 2018, p. 509–514 (2018). DOI https://doi.org/10.1145/3159450.3159541

147. Vivian, R., Falkner, K.: Identifying teachers' technological pedagogical content knowledge for computer science in the primary years. In: 15th International Computing Education Research Conference, ICER 2019, pp. 147–155 (2019)
148. Vivian, R., Falkner, K., Falkner, N.: Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development (2014)
149. Vivian, R., Quille, K., McGill, M.M., Falkner, K., Sentance, S., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F.: An international pilot study of K-12 teachers' computer science self-esteem. In: 25th Conference on Innovation & Technology in Computer Science Education, ITiCSE 2020, pp. 117–123 (2020)
150. Whalley, J., Goldweber, M., Ogier, H.: Student values and interests in capstone project selection. In: Proceedings of the Nineteenth Australasian Computing Education Conference, pp. 90–94 (2017)
151. Whalley, J., Settle, A., Luxton-Reilly, A.: Novice reflections on debugging. In: 52nd Technical Symposium on Computer Science Education, SIGCSE 2021, p. 73–79 (2021). DOI https://doi.org/10.1145/3408877.3432374
152. Whalley, J.L., Lister, R., Thompson, E., Clear, T., Robbins, P., Kumar, P.K.A., Prasad, C.: An Australasian study of reading and comprehension skills in novice programmers, using the Bloom and SOLO taxonomies. In: Eighth Australasian Computing Education Conference, ACE 2006, p. 243–252 (2006)
153. Witten, I.H.: A course on "expert systems" for electrical engineering students. In: 18th Technical Symposium on Computer Science Education, SIGCSE 1987, p. 257–260 (1987). DOI https://doi.org/10.1145/31820.31769
154. Wünsche, B.C., Chen, Z., Shaw, L., Suselo, T., Leung, K.C., Dimalen, D., Mark, W.v.d., Luxton-Reilly, A., Lobb, R.: Automatic assessment of OpenGL computer graphics assignments. In: 23rd Conference on Innovation & Technology in Computer Science Education, ITiCSE 2018, p. 81–86 (2018). DOI https://doi.org/10.1145/3197091.3197112
155. Zhang, J., Luxton-Reilly, A., Denny, P., Whalley, J.: Scientific collaboration network analysis for computing education conferences. In: 26th Conference on Innovation & Technology in Computer Science Education Vol 1, ITiCSE 2021, p. 582–588 (2021)

# Computer Science Education Research in Israel


**Michal Armoni and Judith Gal-Ezer**

## 1 Introduction

This chapter is devoted to research on computer science (CS) education conducted in Israel by Israeli CS educators and researchers. CS as an independent discipline has been offered in Israeli universities since the 1960s. The first department of CS in Israel was established in 1969 in the Technion, and the first program was a graduate program. Very soon thereafter, in the middle of the 1970s, computers were introduced into the K-12 arena, mainly in two tiers. The first was intended to teach CS as an independent subject and the second was intended to exploit the potential of computers to teach and learn other subjects, mainly scientific ones, such as mathematics and physics (e.g., Refs. [36, 50]). Later, when computers became more prevalent and the government came up with the goal and the slogan of "a computer for every child", which they were eager to achieve, a third tier was added, independent of CS: using computer applications (e.g., for writing documents and preparing presentations).

Gradually, CS departments, schools, or faculties were established in all the universities and most of the colleges. Later on, most of them also started to offer programs towards obtaining a CS teaching certificate. Obviously, over the years, they have occasionally updated their CS programs. Similarly, the K-12 CS subject has been updated a few times, but the first major change took place in the 1990s, when the high-school CS curriculum was in fact developed again, from beginning to end, resulting in a coherent curriculum, based on solid well-rationalized educational

M. Armoni (✉)
Department of Science Teaching, Weizmann Institute of Science, Rehovot, Israel
e-mail: michal.armoni@weizmann.ac.il

J. Gal-Ezer
Mathematics and Computer Science Department, The Open University of Israel, Ra'anana, Israel
e-mail: galezer@cs.openu.ac.il



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_18

principles; it emphasized the fundamental and scientific concepts of the discipline, rather than just programming or technology [47, 51].

CS educational research existed even before well-established curricula were published and implemented, and just like the CS programs, it has evolved and flourished, dealing with all educational levels, from young children to advanced undergraduate and even graduate students.

Research on K-12 CS education occupies a large portion of computing education research in Israel. It is independent of the educational system, but at the same time, it has rich and fruitful connections with the educational system. For example, as part of the development and implementation of the new curriculum in the 1990s, educational research teams in the universities were funded by the Ministry of Education to develop different parts of the curriculum, including matching textbooks. These teams (including university researchers as well as teachers) performed research-based design and investigated the gradual implementation of the curriculum in schools, in the classrooms of teachers who volunteered to pilot the new courses. New research directions have often emerged from these research-based design projects. In other cases, new research directions were initiated by the researchers, and their outcomes have influenced decisions taken by the educational system. Moreover, conducting research in schools requires permission from the Ministry of Education, and this is usually granted (provided it meets required standards and regulations). In addition, to prepare teachers for the new courses, the development teams at the universities operated professional development courses for in-service CS teachers, and universities also offered programs for prospective CS teachers, towards obtaining a teaching certificate, thus opening up new directions of research focusing on teachers.

Considering everything, in Israel there are four elements that interact, creating a synergy that promotes K-12 CS education [86]: a well-established *curriculum*, accompanied by *research* conducted at the universities, a mandatory formal *teaching license* provided by the Ministry of Education, and *teacher training* programs towards obtaining a teaching certificate offered by the universities.

Currently, CS education is a very active research area, with researchers who reside in schools of education, science teaching departments or CS departments, guiding students towards master's or doctoral degrees in CS education.

Numerous papers on CS education research have been published since the 1970s, dealing with teaching and learning of various CS areas. Some focus on students – K-12, undergraduate, or even graduate students – and other focus on K-12 teachers, in-service as well as prospective teachers. They deal with many aspects of teaching and learning, such as instructional approaches, visualization, assessment, gender, and affective factors, to name but a few. Obviously, it is impossible to mention here all or even most of these publications, or even touch on all the topics and aspects with which they deal.

Instead, we decided to tell the tale of Israeli computer science education research through the perspective of the discipline itself. Like other academic disciplines, CS is underlain by fundamental ideas and concepts [120] that define its nature; it is a discipline that is more than programming and is concerned with computational

problems – understanding, analyzing, and solving them. These ideas and concepts cut across the discipline, recurring in different fields and contexts, from basic to advanced ones. They can and should be taught at various levels, to K-12 and university students, revisited throughout the curriculum, because only a spiral teaching of them [37] can help students perceive them and help them understand the nature of our discipline. We chose to organize this chapter around such ideas, which were often addressed by CS education research conducted in Israel.

To contextualize the research work reviewed in this chapter, we will start by elaborating more on curricular issues, in Sect. 2. Section 3 will review research conducted by Israeli researchers that deal with fundamental concepts and ideas of CS. We will conclude and look forward to the future in Sect. 4.

## 2 Curricular Issues

As noted above, the history of CS as an independent discipline in Israeli universities began in the 1960s, starting with a graduate program offered by the Technion; then it was followed by other universities, and later by colleges. Today, universities in Israel offer undergraduate CS programs and graduate CS programs towards master's and Ph.D. degrees. The first undergraduate programs were designed on the basis of the first published ACM curriculum recommendations in 1968 [1] and have been updated over the years as the discipline and its education have evolved. In addition, as the computing field continued to evolve, the recommendations have been updated to include new computing disciplines, reflecting the new reports for the new disciplines written by the ACM and IEEE task forces. All the programs offered today by Israeli universities and colleges are regulated by the Israeli Council for Higher Education.

Research on CS education at higher education institutes has also evolved. Some of the educational efforts were accompanied by educational research or took advantage of the educational research results and many studies examined teaching and learning processes taking place during undergraduate CS programs (e.g., Refs. [2, 19, 105, 108, 116]).

In the middle of the 1970s, CS was introduced into high schools in Israel. The K-12 system in Israel is centralized; it is under the responsibility and control of the Ministry of Education, namely, all curricula for all subjects and for all age levels are determined by the Ministry of Education, and every school must adhere to them. The subject of CS was first taught in technological schools that offered practical studies. Nevertheless, these programs also included theoretical subjects (e.g., finite automata). These programs were updated to include Spreadsheet and the LOGO programming language, and formed an intermediate curriculum. The content of the curriculum was determined by a committee appointed by the Minister of Education, as is customary in the Israeli educational system. The committee was advised by the supervisor on CS education in the Ministry of Education. The implementation of this intermediate program was accompanied by research, praising the features of

LOGO on the one hand, but also challenging the prevailing opinions, for example, that meaningful learning through LOGO can take place without the assistance of teachers or instructors (e.g., [95]).

At the beginning of the 1990s, the Minister of Education appointed a new committee, composed of CS researchers and educators, officers from the Ministry of Education, and CS high-school teachers. This committee was assigned the task of designing a new high-school CS curriculum. Its goal was not to train the students to become CS professionals. Rather, the goal was to introduce the students to the discipline, so later they would be able to make an informed choice regarding their path in higher education. This curriculum [51] opened a new area of research in Israel, significantly expanding the Israeli CS education research community. Although this curriculum has been updated over the years after its initial implementation, the main principles that guided the designers still stand today.

A key criterion for defining the core subjects to be included in the curriculum was longevity. As we know even better today, the technology changes rapidly, far more than the basic ideas of CS, which have lasting and fundamental value. As a result of this criterion, the program covered CS knowledge and skills that are independent of specific computers or programming languages. The designers of the curriculum followed nine underlying principles, the most important and relevant world-wide are as follows:

- CS is a full-fledged scientific subject, the same as other sciences;
- The program should focus on the key concepts and foundations of the field;
- The program should include mandatory and elective courses (from which the teachers can choose their preferable courses);
- Conceptual and experimental issues should be interwoven – The Zipper Principle;
- Two quite different problem-solving paradigms should be taught;
- New course material must be developed for both students and teachers;
- Teachers certified to teach the subject must have adequate formal CS education.

Most of these principles apply to every subject and in particular, every scientific subject, but at that time, this was not clear regarding CS, and it is still not common knowledge in all countries even today. Indeed, the program introduced both facets of the discipline – the theoretical and the practical – and as the zipper principle states, both were interwoven throughout the program. The teaching of different problem-solving paradigms, namely, different approaches to problem solving, is implemented by using programming languages of different paradigms in different parts of the curriculum.

Requiring teachers, who are the corner stone of the implementation of any curriculum whatsoever, to have a formal education in CS is a major challenge for every country that implements or aims at implementing a new CS curriculum, since it requires a sort of a bootstrapping process, starting to teach CS when there are not enough certified teachers nor suitable programs for training pre-service CS teachers. Fortunately, Israel succeeded in this challenge, since even before the wide implementation of the curriculum, which followed the pilot phase, the

Ministry of Education invested massive efforts and funding to address this issue. Professional development courses that focused on how to teach the new curriculum were provided to all teachers. Those who did not have a CS background were provided with hundreds of hours of training programs, introducing them to the discipline.

After the design of the curriculum was completed, working teams were set up in the Technion, the Hebrew University, the Weizmann Institute of Science, and the Open University of Israel, sometimes two working groups in one institution. Each group was assigned one or two of the program units. Each unit was reviewed by the committee's members, and then initially implemented in a small number of schools. This implementation was accompanied by research for each of the units and the entire program. Each team consisted of researchers, educators, as well as CS high-school teachers, since their pedagogical knowledge and practical experience from their classrooms were invaluable.

Not long after the program was implemented, universities and teacher colleges started to offer complete programs towards obtaining a CS teaching certificate (e.g., Ref. [46]), thus ending the bootstrapping phase. In these programs the teachers acquire pedagogical knowledge and pedagogical content knowledge. They learn about instructional approaches and methods for teaching CS effectively, current curricula, the history of the discipline and its nature, employing current technology for teaching CS, and more. In addition, the "Machshava" center for computer science teachers was established [90], offering teaching resources and professional development activities (e.g., Ref. [91]). Today, professional development courses for CS in-service teachers are regularly offered by the Ministry of Education and by the universities. The massive and continuous efforts to educate, train, and support the teachers in their daily mission has been widely researched (e.g., Refs. [26, 35, 81, 115]), thus paving the way to new pedagogies and teaching tools.

In 2011, the Ministry of Education initiated a technological leadership program for excellent middle-school students, called the Science and Technology Excellence Program (STEP). CS was included in this program [133, 134], alongside the reinforcement of Mathematics and Physics. The goal was to expose students to the principles of CS, teach them to think in a systemic fashion, draw logical conclusions, and mainly develop abstraction and imagination abilities. The designers thought that learning CS and internalizing the principles of the discipline may contribute to success in other subjects learned in school. The curriculum introduced the learners to concepts in algorithmic thinking (also referred to as computational thinking), through unplugged and programming activities. After a few years, the CS part of this program was replaced by a new curriculum. It follows a problem-based and abstraction-based approach to algorithmic problem solving. The entire process of problem solving is carried out by moving between the four abstraction levels of algorithmic problem solving: the *problem* level, the *algorithm* level, the *program* level, and the *execution* level. In the 7th grade, algorithmic solutions are implemented in the event-driven language Scratch, whereas in the 8th and 9th grades the students use Python.

As K-12 computer science became more prevalent in countries worldwide, with some countries designing or adapting curricula that start from elementary school (and sometimes even before that), Israel started debating whether the pipeline of CS education can and should be extended down to elementary school [16]. In 2015, it was decided to start teaching CS in elementary schools to 4th–6th grade students. These grades were chosen because younger students are still coping with mother tongue acquisition and basic mathematics. A curriculum called *computer science and robotics* was designed by the Ministry of Education for a soft introduction to computer science. Algorithms are implemented in Scratch.

This curriculum is currently taught in a few hundred schools; it is still in a pilot phase, and it has already undergone some changes. However, its implementation poses a considerable challenge. First, teaching CS to young children, without reducing it only to programming (let alone coding), is a difficult educational task. Second, once again, we are facing the bootstrapping process of training teachers when training programs for prospective CS elementary teachers do not yet exist, and although we have a nice cadre of high-school teachers (though not enough), there are hardly any elementary school teachers who have the disciplinary knowledge required to teach CS. The Ministry of Education and the Ministry of Finance allocated funding for the purpose of teacher training and every teacher that plans to teach this curriculum takes a training course of a few dozen hours. Research on this issue is now taking place (e.g., Refs. [22, 43]).

## 3 Fundamental Ideas and Concepts of CS

There are many fundamental ideas and concepts of CS. Here we will focus on those that are addressed by Israeli CS education researchers. This section is organized into seven subsections, the first six of which deal with some CS fundamental ideas and concepts, respectively, starting with abstraction (Sect. 3.1), followed by problem-solving paradigm (Sect. 3.2), correctness and efficiency (Sect. 3.3), non-determinism (Sect. 3.4), concurrency (Sect. 3.5), and Reduction (Sect. 3.6). The seventh section will deal with research about problem-solving strategies.

### *3.1 Abstraction*

Abstraction is the core of CS, its most fundamental idea. CS experts use abstraction in different contexts and move freely between levels of abstraction. In fact, many of the ideas described below in the following sections, are manifestations of abstraction. Abstraction can be used for modeling, formalization, generalization, ignoring details, transfer among domains, and more. Since abstraction is also a central idea in the context of the mature discipline of mathematics, several

publications discussed the differences and similarities between these disciplines regarding this idea (e.g., Refs. [3, 96]).

Haberman et al. examined the teaching and learning of logic programming as a second problem-solving paradigm for high-school students (see Sect. 3.2). A major focus of this research project was the concept of abstract data types (ADT), through which the declarative and procedural aspects of logic programming can be taught. ADTs ignore or hide the concrete organization of data as well as the implementation details of manipulating the data. Haberman et al. introduced a didactic method for teaching ADTs, which uses evolving boxes – black, white, and gray, and explored its effectiveness and cognitive aspects [76, 79, 80]. This approach was shown to be effective for many students, although some used it only partially or violated the hiding of details in various ways.

The teaching and learning of ADTs was also studied in the context of the procedural paradigm, for example, the cases of the ADTs of a binary tree and a linked list in an advanced high school course [74, 75, 117]. Aharoni [2] studied the teaching and learning of ADTs in the context of a data structures course for undergraduate students. He showed the students' tendency to reduce the level of abstraction, working with ADTs from the perspective of a process rather than the perspective of an object.

The duality of process-object in terms of abstraction is a known framework from mathematics education [121]. Hazzan extended and generalized this framework, demonstrating students' tendency for reducing the level of abstraction when introduced to a new concept. She employed this framework in mathematics as well as in computer science [83], for undergraduate students in various curricular contexts. For example, computability theory [84] and graph algorithms [85]. Ginat and Blau [68] used Hazzan's framework to show the tendency of senior undergraduate CS students to reduce the levels of abstraction in the context of algorithmic problem solving. Based on Hazzan's framework, Armoni [11] introduced a didactic framework for teaching procedural abstraction for novices, differentiating between four levels of abstraction [111]: problem, algorithm (object), program, and execution. The effectiveness of this framework was shown in the context of an introductory CS course for middle-school students [124], indicating an even extended effect for girls [123]. Ginat [66] relied on the same 4-level hierarchy to show that following an intervention in which declarative observations were employed in an algorithmic problem-solving course for undergraduate students, the students were able to work at higher levels of abstraction.

Pattern-Oriented Instruction (POI) is an instructional approach for teaching problem-solving, using algorithmic patterns, which by nature employ procedural abstraction. Indeed, the integration of this approach in an introductory CS course for high-school students was shown to improve students' abstraction skills [104, 106]. Haberman and Muller [77] analyzed and compared the instructional approach of POI with an ADT-oriented approach regarding abstraction. Ginat and Menashe [69] utilized algorithmic patterns to define a taxonomy for assessing students' learning outcomes regarding algorithmic problem solving. This taxonomy was built on the SOLO taxonomy [34], which takes the perspective of abstraction. The

higher is the level reached by a student, the more abstract is the employment of algorithmic patterns in the student's algorithmic solution. Ginat and Menashe used this taxonomy to assess high-school students' solutions for several algorithmic problems.

Students' difficulties with abstraction were shown by several researchers, in different levels and curricular contexts. For example, Omar et al. [107] in the context of decomposition and reuse, with high-school students; Haberman et al. [82] in the context of procedural abstraction and in particular the concept of an algorithm, with high-school students; Lavy et al. [94, 108], in the context of an undergraduate course on object-oriented programming; Ginat and Alankry [67] in the context of concatenation of formal languages in a high-school course on computational models; Ginat et al. [70] in the context of algorithmic problem solving, with both undergraduate and high-school students; Ginat [64] in the context of the abstract notions of 'as-if' and 'don't-care' for solving algorithmic problems, with undergraduate students.

In contrast, Alexandron et al. [6] found that a course on the scenario-based paradigm had a positive effect on abstraction skills, of both graduate and high-school students. In a very different context of a high-school course on computational science, Taub et al. [128] found that when developing simulations of physics phenomena, moving between levels of abstraction in the context of CS facilitates moving between levels of abstraction in the context of physics.

In all these contexts, the level of programming in a high-level language was a lower level of abstraction, below the levels of the problem and its algorithmic solution. In contrast, Schocken and Nisan developed a course in which abstraction is a major recurring context, and in which students move between even lower levels of abstraction below the programming level. In a series of projects the students design a computer system in a bottom-up manner. The students start from the very low level of logic gates, and gradually move up through hardware and software levels (e.g., computer architecture, machine language, assembly, compiler, and operating systems) [118].

## *3.2 A Problem-Solving Paradigm*

Problem solving lies at the heart of CS, and hence, a problem-solving paradigm is an important CS idea. It expresses the expertise of computer scientists as problem solvers, who among other things, are experts in choosing the appropriate, best fit way of thinking by which they approach the problem at hand. In other words, they choose the appropriate form of modeling the problem space, or yet in other words – abstracting the problem space. Thus, every problem-solving paradigm has its own abstraction method. Furthermore, other recurring CS ideas and concepts (e.g., recursion) can be relevant to all or several paradigms but expressed in a different form in each of them.

A more concrete perspective on problem-solving paradigms is expressed through programming languages, where each problem-solving paradigm can be implemented by different programming languages. Although this concrete perspective is certainly important, the more abstract level of problem-solving paradigm is necessary to understand this idea (rather than just using it).

In general, educational research on problem-solving paradigms can be divided into a few types: how to teach a certain paradigm, what should be the first paradigm taught, how does the first paradigm affect the teaching of other paradigms, students' difficulties when learning a specific paradigm, comparing different paradigms regarding different aspects, and the meta-type of learning the idea of a problem-solving paradigm.

Typically, undergraduate CS programs include more than one problem-solving paradigm, where the first is introduced in CS1 and additional ones are introduced in more advanced courses. Several Israeli studies examined the idea of a problem-solving paradigm in the context of higher education. Some dealt with students' understanding of specific OO concepts [25, 94, 108]. Not surprisingly, some dealt with the controversial issue of Object-first-vs-object-later, which extensively occupied the international CS education community [53, 73, 132].

Alexandron et al. examined issues concerning the teaching of the scenario-based paradigm in a graduate course, using the LSC (Live Sequence Charts) programming language [39]. In the scenario-based approach, "a program consists of a set of multi-modal scenarios. The execution mechanism follows all scenarios simultaneously, adhering to them all, so that any run of the program is legal with relation to the entire set of scenarios" [23]. They found that when using LSC, the students tended to adopt an external and usability-oriented view, whereas when using another paradigm of their choice to solve a programming challenge, they adopted an internal and implementation-oriented view [5]. In their work they also referred to the "mother tongue" issue. They found [4] that previous programming experience can affect the learning of scenario-based programming, leading students to use familiar programming patterns in a manner that interferes with the new concepts, resulting in their poor usage and even unexpected behavior of the students' artifacts. The paradigm of scenario-based programming is also used for Plethora, a new educational environment for teaching computational problem solving to elementary-school students [23].

The issue of a problem-solving paradigm was widely examined in the context of K-12 CS education. As mentioned in Sect. 2, one of the underlying principles of the high-school CS curriculum [51] is that students should be exposed to more than one programming paradigm, to "another language, of radically different nature, that suggests alternative ways of algorithmic thinking" (p. 76). In line with this principle, several courses dealing with different paradigms were developed, and extensive research accompanied their development and enactment. For example, several papers (e.g., [93]) reported on the learning of functional programming (in Scheme), examining students' conceptions of automated assignment [109] and the functional evaluation process [92], and the conceptual conflict between the procedural and the functional paradigms demonstrated by the students [92]. An

interesting paper examined phenomena known from mathematics education, in the context of functional programming [110]. They found evidence of a clash between the conception of functions as actions on objects and their formal conception, a clash that occurred between the conception of a function as a change and its conception as a mapping, as well as between the conception of a chain of actions and the conception of composition of functions. Interestingly, this study led to further examination of the conception of functions in a mathematical context [97].

Teaching and learning the logic paradigm (using Prolog) was also reported in several papers, mainly through the perspective of abstract data types (see Sect. 3.1) and their use for knowledge representation, as well as problem solving (e.g., Ref. [80]). The concept of recursion was studied both in the context of the functional paradigm [98] and the logic paradigm [75], thus nicely demonstrating the idea of a problem-solving paradigm.

Numerous papers have dealt with the object-oriented (OO) paradigm, following the gradual shift in the high-school CS introductory course, from the procedural to the OO paradigm, which took place more than 15 years ago (e.g., Refs. [99, 112, 114, 122, 129]). These mostly focused on the learning of OO concepts and ideas. Haberman and Ragonis [78] discussed the similarities and differences between the OO paradigm and the logic paradigm and suggested establishing links between the two courses in the high-school program, thus supporting the learning of each paradigm, as well as promoting a coherent conception of the idea of a problem-solving paradigm.

Alexandron et al. [8, 9] developed and implemented a high-school course on scenario-based programming using LSC. They found that the course encouraged abstract thinking [6] and provided a good context for learning the concept of non-determinism [7].

Israeli middle-school students who study CS are introduced to the event-driven paradigm by means of the Scratch environment. Their learning of CS concepts in this course was reported by Meerbaum-Salant et al. [102]. A subsequent study [21] examined the transition from the event-driven paradigm (9th grade) to the OO paradigm (10th grade), by comparing the achievements of 10th-grade CS students who studied CS in middle school with the achievements of those who did not. They found that although some differences in understanding some CS1 concepts could be identified throughout the school year, by the end of the school year there were no significant differences. In another study, Meerbaum-Salant et al. [101] identified patterns of programming used by the students, which were not consistent with standard habits of programming (for example, modularization). Gordon et al. [72] argued that these habits are consistent with the scenario-based paradigm, thus hinting at its naturalness for the young students.

Finally, Stolin and Hazzan [127] reported on a course on programming paradigms for pre-service teachers, which was organized around the theme of abstraction and dealt with four paradigms (functional, procedural, OO, and concurrent). Their study investigated the students' understanding of the concept of a programming paradigm, particularly their way of relating to this concept when discussing the different paradigms.

### 3.3 Correctness and Efficiency

Correctness and efficiency are fundamental CS concepts when designing, analyzing, or reasoning about algorithms. They are relevant for any algorithm for a computational problem, in any model of execution. These concepts can be learned at various levels of rigor, for example, by using rigorous proofs of correctness and big-O analyses of complexity, by using verbal arguments, or by using representative test cases and loop-based estimated counts.

Ginat [55] argued for using assertions as a pedagogical tool for proving (or justifying) correctness and measuring efficiency, as well as a tool for algorithmic design (see Sect. 3.7). As a tool for establishing correctness, such assertions can be used for tagging the algorithm, as is done in algorithmic verification [42, 87], for example, as loop invariants or as entry and exit assertions for parts of an algorithm (e.g., segments and solutions for sub-tasks). Tagging with loop invariants can serve to analyze efficiency by establishing a worst-case bound of the number of rounds until the invariant does not hold.

Ginat contended and illustrated that despite such assertions' perceived formal nature and difficulty, their use can be taught to students at a rather intuitive level. This pedagogical tool was embedded in the textbook for the first introductory CS course for 10th-grade students. It was developed during the implementation of the 1990s' curriculum at the request of the Ministry of Education (the textbook used the procedural paradigm and Pascal as a programming language. It has been out of use, since the problem solving-paradigm used in the introductory courses and the programming language were changed).

This book also included a chapter on efficiency, in terms of both time and space. Ginat [54] argued that unlike the concept of complexity, which requires formal mathematical tools, efficiency can be taught relatively early, after introducing repeated execution. Students are introduced to problems with multiple solutions that vary in their efficiency; they learn how to express the number of iterations in a loop in terms of input size, and to examine the need to use arrays whose sizes depend on the input size. This relatively gentle introduction can serve as the basis of the spiral teaching of efficiency. Later, in a mandatory unit dealing with data structures, the students are re-acquainted with efficiency, this time using the big-O-notations; however, in line with spiral teaching, the learning of efficiency is suitable for the age level of sophomore or senior high-school students.

Gal-Ezer and Zur [49] investigated students' misconceptions of efficiency and related achievements following the learning of this concept as introduced in the textbook. They found that intuitive rules [125] often govern students' conception of efficiency. Specifically, they tend to think that the shorter a program, the more efficient it is; the fewer variables in a program, the more time-efficient it is; two programs that include the same statements (no matter the order) are equally efficient, and two programs that accomplish the same task are also equally efficient.

Gal-Ezer et al. [52] also aimed at an early introduction of efficiency, in the context of a CS1 course for undergraduate students. The introduction was gradual, but unlike

the introductory high-school course, it was more formal, also including the big-O notations. The instructional approach employed by Gal-Ezer et al. was found to be effective and students' achievements improved (compared to previous semesters).

In a series of studies, Ben-David Kolikant et al. studied students' conceptions of correctness. Ben-David Kolikant and Pollack [32] found that high-school students were tolerant regarding some errors in their programs and were satisfied with programs that "worked in general" or "worked for many input examples". Later, Ben-David Kolikant [29] found that high-school students as well as college CS graduates were satisfied with "relatively correct programs". Ben-David Kolikant and Mussai [31] took the perspective of incorrectness, and investigated high-school students' misconceptions and their connections to the students' existing knowledge. They found that students' conception of incorrectness did not complement their conception of correctness. Rather, they tended to assume a gray area in which programs are partially incorrect, since they achieve part of their goals. This conception stemmed from a summative grading scheme that was often used by teachers to grade students' assignments, according to which a program may get a non-zero score if it achieves some of its goals or made some progress towards a goal. For these students, incorrect programs were only those that deserved a score of 0.

Ben-David Kolikant and Pollack [32] challenged the norm among high-school students and their teachers, according to which it is sufficient to successfully run a program in order to establish its correctness. Their instructional approach was intended to establish a new norm, according to which incorrect programs cannot be tolerated. Using mathematical problems, the students developed mathematically oriented programming skills, realizing that correct algorithmic solutions can be found by first analyzing the problem at hand and then concluding what the output should be for all possible subsets of inputs. These subsets can later serve for choosing representative test cases, thus achieving a high-coverage testing together with explanatory proofs. To this end, they used class discussions, since social contexts are known to be effective for acquiring norms.

The idea of correctness is especially challenging in the context of concurrent and distributed computing, since a successful run of a specific test case is insufficient even when arguing that a program is correct regarding this test case. All possible interleavings should be considered. Indeed, several studies demonstrated students' difficulties with this idea in a concurrent context (see Sect. 3.5).

## *3.4 Nondeterminism*

Nondeterminism (ND) is a challenging abstract CS idea that is relevant in many CS areas and has many facets. For example, ND is manifested in the context of computational models through nondeterministic automata (NDA), as a way of augmenting expressiveness. Similarly, in the context of programming, ND can be manifested through nondeterministic algorithmic constructs, again augmenting expressiveness. In this context it represents the notion of *don't care*, since all

possible computations are considered equally good. In both these contexts, using ND is a matter of choice. In contrast, in the contexts of concurrent or distributed computing, ND represents unpredicted behaviors stemming from a set of possible interleavings or timings, over which the designer has no control. Note that students may face this kind of ND at a relatively early age, since even the block-based language of Scratch, which is very popular in K-9 CS education, is inherently concurrent. In all these manifestations of ND, ignoring details or being prepared for unpredicted situations is inherent, hence the deep connection of ND with abstraction.

In the Israeli high-school program, ND is included in the elective course on computational models, through NDA. Armoni and Gal-Ezer [14] described the rationale and underlying considerations as well as the guidelines for teaching ND in high school as part of the computational models course. They also studied the ways in which the students used ND. The findings indicated that many students tended not to fully exploit the expression power of NDA and demonstrated several patterns of partial use of ND. Following this study, they continued to study the teaching and learning of ND at the undergraduate level in the same curricular context – an undergraduate course on automata and formal languages [15]. They analyzed students' errors when using ND, their tendency to use ND and the quality of their solutions regarding ND, and found various levels (sometimes unsatisfactory) of their tendency to use ND and the quality of using it. This motivated further exploration of the teaching and learning of this concept. In another study, Armoni, Lewenstein, and Ben-Ari [20] found that undergraduate students do not perceive NDA as unpredictable entities. However, a simple change in the teaching process of the course proved highly efficient and led to students having better perceptions of NDA and ND. These perceptions were the motivation for turning to explore the development and treatment of ND throughout the history of CS and CS education, in its various facets and manifestations [12]. It yielded a taxonomy of ND manifestations as well as recommendations for the teaching of ND, at all curricular levels.

The teaching and learning of ND at the high-school level was also studied by Alexandron et al. [7], as part of their research on teaching scenario-based programming in high school using LSC. LSC includes various forms of ND [39]. As a scenario-based language, there are no assumptions regarding ordering (unless explicitly stated otherwise, using the language constructs). In addition, some of LSC's constructs are nondeterminisic (for example, a weighted *select* construct, which can be viewed as a probabilistic version of Dijkstra's guarded commands [40]). Alexandron et al.'s study demonstrated that high-school students can perceive, understand, and effectively use these manifestations of ND.

Ginat [63] studied the computational notion of *don't care*. This was done in an algorithmic deterministic context, where students were expected to find or understand efficient solutions in which certain aspects of the problem were ignored. He pointed out several substantial difficulties students had regarding this notion. Despite the deterministic context, Ginat linked this to ND, viewing the notion of "*don't care*" as universal, in the sense that its essence is the same in

both its deterministic and nondeterministic manifestations. In another study, Ginat and Alankry [67] studied high-school students' understanding and performance of concatenation in the context of the computational models course, and specifically, concatenation of formal languages. Concatenation of formal languages involves ND thinking (expressed in the nondeterministic canonical construction that based on two given automata, constructs a nondeterministic automaton for the concatenation language).

A noted above, ND is strongly related to concurrency. Several studies explored the teaching and learning of concurrency (see Sect. 3.5), but although the connection was acknowledged, the learning and teaching of ND was not discussed in those publications.

## *3.5 Concurrency*

Concurrency is another abstract and challenging CS concept. As noted above, it is strongly related to ND. Namely, ND is inherent in any model of computation in which multiple interleavings are possible. Traditionally, concurrency is considered an advanced idea, and it is usually addressed in advanced courses on concurrent and distributed algorithms, computational models (e.g., through Petri nets), semantics of programming languages (e.g., through constructs such as Dijkstra's guarded commands), and more. However, as noted in Sect. 3.4, nowadays concurrency may be inherent even in computing environments such as Scratch, which are designated for young children.

In Israel, concurrency at the K-12 level was addressed in two educational research contexts:

1. The context of teaching concurrent and distributed computing.
2. The context of basic CS courses for young children using a concurrent language.

The 1990s' high-school CS curriculum [47] included a 45-hour course on concurrent and distributed computing (CDC). It was an advanced course, taken at grade 11 or 12, following a chain of two introductory courses and a course presenting a second paradigm, and in parallel with an advanced course on data structures and software design. Several studies accompanied the iterative development of this course. Ben-Ari and Ben-David Kolikant [24] described the course, the motivation for teaching it, and the pedagogical considerations behind it, and investigated the students' learning. They found that the students had some difficulties, for example, with considering multiple interleavings, or in general with the concept of correctness in a concurrent model; however, students' understanding evolved throughout the course, and by the end of the course, most of them could solve problems that required process coordination and they reached high achievements. Moreover, they could discuss and explain concurrent and distributed systems using concepts learned during the course. In another study, Ben-David Kolikant et al. [33] investigated students' mental models of semaphores. Because non-viable models

were detected, indicating misconceptions of semaphores, the course was updated to avoid the development of these misconceptions. Ben-David Kolikant [27] has also studied students' preconceptions of concurrency. Students' prior knowledge on synchronization (computerized as well as human) was found to be rich, but when thinking of human agents, students tended to assume capabilities that are not viable in the context of computing. This includes, for example, being able to determine when one can stop waiting, being able to spontaneously adapt to a constant rate of actions, or being able to receive a message spontaneously, without the need to perform an action, whereas sending a message required an explicit action (as is the case with hearing and talking, respectively). In line with the theory of constructivism, the course was then updated to address these preconceptions. For example, the use of authentic settings from the world of computers was preferred to the use of imagined settings (such as in the "dining philosophers" problem). In another study, Ben-David Kolikant [28] analyzed the evolvement of students' understanding of synchronization. She found evidence of a pattern-based technique, where a set of pattern problems and their solutions is acquired and used to solve new given problems. Although this technique was efficient, it also limited students' performance when they faced a problem that did not meet a known pattern. Later, Ben-David Kolikant and Ben-Ari [30] suggested that these difficulties and perceptions stem from a cultural clash between the culture of computer users, from which most of the students arrive, and the professional CS culture. Instructional approaches aimed at resolving the clashes were shown to be effective. For example, the students could handle abstract problems (such as the "dining philosophers" problem), realizing that it represents a group of concrete problems.

Schwarz and Ben-Ari [119] examined the tendency of students who learned the CDC course to use state diagrams as a tool for explaining concurrent solutions, specifically for convincing others of or refuting their correctness. They found that in general, students preferred verbal arguments rather than the use of state diagrams. However, some of the students acknowledged the potential of state diagrams and used them, and others used them only occasionally, when verbal arguments failed. Schwarz and Ben-Ari conjectured that state diagrams facilitate the formation of mental models, even if they are not used often as an argumentation tool; hence, they recommended including them in the course.

The teaching and learning of concurrency was also studied in the context of an introductory CS course for middle-school students; it used the block-based Scratch environment [102]. As an event-driven language, concurrency is manifested in Scratch in two ways: by several sprites executing scripts concurrently (Type-I concurrency), and by a sprite executing more than one script simultaneously (Type-II concurrency). Type-I concurrency was more intuitive for the students, but they had difficulties with Type-II concurrency, probably due to a reduced perception of the concept (i.e., identifying it with its specific aspect of synchronization or with a concrete process such as the set of instructions for message passing). Meerbaum-Salant et al. pointed out that although Scratch is perceived by educators as mostly suitable for young children, because of its friendly colorful block-based interface and the colorful animated artifacts, one should bear in mind that it also poses

complex learning challenges, such as the abstract concept of concurrency (which cannot be avoided even in rather simple projects). Interestingly, when self-designing projects, students exhibited habits of programming that preferred fragmentized concurrency rather than sequential modularization [101], suggesting, as argued by Gordon et al. [72], that thinking in scenarios (where concurrency is inherent) is natural for the students.

### *3.6 Reduction*

Reduction in CS education is mostly mentioned in the context of courses on the theory of computer science, where it is used to prove non-decidability or difficulty of problems. Reduction is also useful in algorithmic design, when reducing a problem to be solved to another, already solved problem. From a generalized point of view, solving a problem by reduction means transforming the problem at hand into other problems (one or more), which are already solved or are easier to solve, and using their solutions as black boxes for obtaining a solution to the problem at hand. Using this perspective, reduction is a CS recurring concept, relevant in almost every area of CS.

Gal-Ezer and Trakhtenbrot [48] studied the classic use of reduction in an undergraduate course on the theory of computer science (computability and complexity). They identified five misconceptions, indicating that even in this familiar context, the teaching of reduction has yet to improve.

Armoni et al. [18] looked into other manifestations of reduction. They examined high-school students' use of reduction in the context of the course on computational models. In this context, reduction can be used to classify formal languages and design automata using closure properties or known construction algorithms. According to the findings, many students neglected to employ reduction, even when using reduction could lead to elegant and shorter solutions. They also found that when reduction was employed, the solution's level of reduction (in terms of the cognitive distance between the problem at hand and the reduced-to problem) was often relatively low.

The next step was to examine the same issue in the context of undergraduate students [13]. Although the students' employment of reduction was somewhat better, it was not as good as one could hope, and in general, the findings indicated an unsatisfactory level of reduction-related skills.

In a study that followed, Armoni et al. [19] extended the curricular perspective. The study focused on first- and third-year undergraduate CS students and on CS graduates who learned towards obtaining a CS teaching certificate, and examined their use of reduction on CS1 questions as well as questions related to the algorithms course or the computational models course. It was a qualitative interview-based study. The findings indicated that the development of reduction-related skills evolves over time. First-year students barely ever used reduction, whereas the more mature students exhibited higher levels of awareness of the concept of reduction as

well as of its potential use in different problem-solving situations. However, even they did not sufficiently exploit the power of reduction. For many students there was a clash between the abstract nature of reduction (expressed by the use of black boxes) and the tendency to work at lower levels of abstraction. They had difficulties in connecting between problems, and their use of reduction was often limited to certain curricular contexts and was not transferred to others. They doubted the legitimacy of using reduction in some contexts and did not perceive it as a rewarding problem-solving heuristic, whose use reflects high problem-solving skills. Often they did not use reduction properly, tending to open the black box and confusing a problem and its solution.

This was also studied in a quantitative study [10], focusing on the algorithms course at the undergraduate level; the findings were consistent with previous ones: often students' solutions by reduction were of a low level of abstraction, lacking a clear black-box component. The black box was corrupted, for example, by opening it and referring to the inner details of the solution hidden in it. It also seemed that students did not fully understand the nature of reduction, often failing to relate it to problems.

Based on all the findings above, a framework for teaching reduction was developed and investigated in an undergraduate algorithms course [44]. Integrating this framework into the course had a positive effect on the students' use and understanding of reduction.

## *3.7 Problem-Solving Strategies*

CS experts deal with computational problems. They analyze them, classify them, and try to solve them if possible. Hence, computational problem solving constitutes a major component of CS, and as such, it makes use of CS fundamental ideas and concepts, among which are those that were covered in Sects. 3.1, 3.2, 3.3, 3.4, 3.5, and 3.6. In particular, some of them can also serve as problem-solving strategies. For example, reduction (Sect. 3.6), and specifically the use of black-box solutions to other problems, is an effective problem-solving strategy. Similarly, each problem-solving paradigm (Sect. 3.2) also constitutes a high-level strategy for problem-solving, since by choosing a paradigm we also choose how to address the problem, and some choices may be better than others, depending on the problem's characteristics; they may induce an easier or more natural path to a solution.

Algorithmic patterns (APs) were mentioned in Sect. 3.1 as the core of the beneficial instructional approach of POI, which has a positive effect on the acquirement of abstraction skills. They also constitute a problem-solving strategy that can be used to obtain elegant and efficient solutions to computational problems. APs are algorithmic solutions for problems that are canonic (for example, searching, finding the maximal element, and scanning of adjacent elements in a sequence), in the sense that they can be used in the solutions of many other problems. Unlike reduction, in which one also uses a solution to one problem to solve another, APs

are not closed in black boxes, and hence can be used in various ways, including concatenation, interleaving, and nesting. The use of APs for problem solving was studied, for example, by Ragonis [113] in the context of prospective teachers, and by Muller [103], Ginat and Menashe [69] and Ginat et al. [71] in the context of high-school students.

Embedding assertions was discussed in Sect. 3.3 as a means for justifying correctness and measuring efficiency. It can also be used as a strategy for algorithmic design, as declarative insights into the problem at hand, which can lead to elegant algorithms. These are often more efficient than those based on operative thinking to which students often tend [59]. Ginat discussed and studied students' use of declarative reasoning vs. operative reasoning in additional publications (e.g., Ref. [66]).

Armoni and Ginat [17] identified and characterized the fundamental idea of reversing, or thinking in reverse. Recursion is a private case of reversing, and there are more, for example topological reversing, logical reversing, and mathematical inversion. Each of these forms is a rewarding problem-solving strategy, which requires the solvers to go beyond the natural forward thinking.

Ginat identified and reasoned about a rich variety of problem-solving strategies (e.g., on-the-fly computations [62], inductive progress [65], and binary perspectives [56]), although he did not always accompany his illuminating observations with empirical investigations. One of these strategies is decomposition, which is strongly connected to abstraction, expressing moving down the levels of abstraction. There are various ways to use decomposition, for example, top-down task decomposition or data decomposition (e.g., decomposing a range into two parts, as in divide-and-conquer). Ginat [57, 58] discussed this strategy, introducing various (sometime very sophisticated) forms of it. Ginat studied the ability of CS graduates, who taught CS or math (in high-school or college) to solve a non-standard task whose solution requires a sort of geometrical decomposition [61]. He found that most of the graduates did not turn to this strategy and failed to reach a correct solution. Omar et al. [107] studied high-school students' use of decomposition (by means of functions) when solving programming tasks. Similar to Ginat, they found unsatisfactory use of this form of decomposition.

Besides failing to employ certain rewarding strategies, insufficient mastery of problem-solving strategies may also be expressed by adopting ineffective strategies. Ginat [60] found that excellent high-school CS students often turn to the *design-by-keyword* strategy, in which their choice of solution is influenced by specific words or phrases in the problem description, which often misguide them to non-efficient or incorrect solutions.

## 4 Concluding Remarks

This chapter presented a review of Israeli research on computer science education. It is by no means an exhaustive review, which could hardly be done in one chapter.

Our decision to focus on studies that deal with fundamental ideas and concepts of computer science has led us to leave out of this chapter many other important studies. For example, there is rich body of research on visualization in computer science education, on teacher preparation and professional development, and more. By focusing on computer science we also left out many interesting publications on software engineering and in particular on project design. However, we see the issue of the nature of CS as a major aspect of CS education, especially since research worldwide indicates that there are many misconceptions regarding the nature of the disciplines among the general public and hence also among our students who are newcomers to CS. Addressing these misconceptions and the inaccurate image of CS includes also stressing what CS is. Research on its fundamental ideas and concepts is highly important for achieving this educational goal.

Israeli researchers cooperate with the international community of computing education, leading to extensive research that was not represented in this chapter. Specifically, Israeli researchers have been taking part in efforts initiated or motivated by international groups. These include for example, working groups [88, 89, 100, 130] and international educational committees [38, 41, 45, 126, 131]. These activities have had an impact worldwide and Israel became an influential factor regarding the development of K-12 CS curricula, their implementation, and their research. Some of these efforts examined the current K-12 CS curricula in the US and Europe [126, 131], and then touched upon challenging issues such as designing K-12 curricula, or at least an appropriate framework [38] that can serve all European countries. The main concerns of the different teams were, and still are, as follows: the nature of CS, and specifically, the confusion that still exists between CS and other areas such as ICT and digital competencies [131], inclusion [38], and the challenge of training knowledgeable teachers [41].

Future work is already underway. The Israeli CS education research community explores a wide range of issues, elaborating on themes mentioned above as well as others. CS education research has developed worldwide and is widely appreciated. Hence, more aim at enrolling in graduate research programs in this area, and more research is done. Additional computing areas have been developed and call for research regarding their teaching and learning at all levels. Furthermore, older curricula need to be updated and implemented, with research conducted in parallel. New problems arise, and researchers find more challenges to address. The Israeli CER community will broaden the connection with the World CER community and hopefully will continue to contribute to the area of CS education and CS education research.

## References

1. ACM Curriculum Committee on Computer Science (1968). Curriculum 68: Recommendations for academic programs in computer science. *Communications of the ACM* 11(3), 151–197.

2. Aharoni, D. (2000). Cogito, ergo sum! Cognitive processes of students dealing with data structures, In *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education*, 26–30.
3. Aharoni, D., and Leron, U. (1997). Abstraction is hard in computer-science too. In *Proceedings of the Conference of the International Group for the Psychology of Mathematics Education* (PME), 2:9–16.
4. Alexandron, G., Armoni, M., Gordon, G., and Harel, D. (2012). The effect of previous programming experience on the learning of scenario-based programming. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, 151–159.
5. Alexandron, G., Armoni, M., Gordon, G., and Harel, D. (2014). Scenario-based programming, usability oriented perception. *ACM Transactions on Computing Education* 14(3), 21:1–23.
6. Alexandron, G., Armoni, M., Gordon, M., and Harel, D. (2014). Scenario-based programming: reducing the cognitive load, fostering abstract thinking. In *Proceedings of the 36th International Conference on Software Engineering* (ICSE), 311–320.
7. Alexandron, G., Armoni, M., Gordon, G., and Harel, D. (2016). Teaching nondeterminism through programming. *Informatics in Education* 15(1), 1–23.
8. Alexandron, G., Armoni, M., Gordon, M., and Harel, D. (2017). Teaching scenario-based programming: an additional paradigm for the high school computer science curriculum, Part 1. *Computing in Science & Engineering* 19(5), 58–67.
9. Alexandron, G., Armoni, M., Gordon, M., and Harel, D. (2017). Teaching scenario-based programming: an additional paradigm for the high school computer science curriculum, Part 2. *Computing in Science & Engineering* 19(6), 64–71.
10. Armoni, M. (2009). Reduction in CS: a (mostly) quantitative analysis of reductive solutions to algorithmic problems. *Journal on Educational Resources in Computing* 8(4), 11:1–30.
11. Armoni, M. (2013). On teaching abstraction in computer science to novices. *Journal of Computers in Mathematics and Science Teaching* 32(3), 265–284.
12. Armoni, M., and Ben-Ari, M. (2009). The concept of nondeterminism: its development and implications for education. *Science & Education* 18(8), 1005–1030.
13. Armoni, M., and Gal-Ezer, J. (2006). Reduction – an abstract thinking pattern: the case of the computational models course. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 389–393.
14. Armoni, M., and Gal-Ezer, J. (2006). Introducing non-determinism. *Journal of Computers in Mathematics and Science Teaching* 25(4), 325–359.
15. Armoni, M., and Gal-Ezer, J. (2007). Non-determinism: an abstract concept in computer science studies. *Computer Science Education* 17(4), 243–262.
16. Armoni, M., and Gal-Ezer, J. (2014). Early computing education – Why? What? When? How? *ACM Inroads* 5(4), 54–59.
17. Armoni, M., and Ginat, D. (2008) Reversing: a fundamental idea in computer science. *Computer Science Education* 18(3), 213–230.
18. Armoni, M., Gal-Ezer, J., and Tirosh, D. (2005). Solving problems reductively. *Journal of Educational Computing Research* 32(2), 113–129
19. Armoni, M., Gal-Ezer, J., and Hazzan, O. (2006). Reductive thinking in computer science. *Computer Science Education* 16(4), 281–301.
20. Armoni, M., Lewenstein, N., and Ben-Ari, M. (2008). Teaching students to think nondeterministically. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, 4–8.
21. Armoni, M., Meerbaum-Salant, O., and Ben-Ari, M. (2015). From Scratch to "real" programming. *ACM Transactions on Computing Education* 14(4), 25:1–15.
22. Armoni, M., Gal-Ezer, J., and Ulmer, C. Professional development of primary school teachers participating in a pilot project on teaching computer science to fourth graders. In preparation.
23. Armoni, M., Gal-Ezer, J., Harel, D., Marelly, R., and Szekely, S. (In Press). Plethora of skills: a game-based platform for introducing and practicing computational problem solving, to be published in: H. Abelson & K. Siu-Cheung (Eds.) *Computational Thinking Curricula in K-12: International Implementations*. MIT Press. Cambridge, MA.

24. Ben-Ari, M., and Ben-David Kolikant, Y. (1999). Thinking parallel: the process of learning concurrency. In *Proceedings of the 4th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education*, 13–16.
25. Benaya, T., and Zur, E. (2008). Understanding object oriented programming concepts in an advanced programming course. In *Proceedings of the 2nd International Conference on Informatics in Secondary Schools: Evolution and Perspective* (ISSEP), *Lecture Notes in Computer Science* (LNCS 5090), 161–170.
26. Ben-Bassat Levy, R., and Ben-Ari, M. (2007). We work so hard and they don't use it: acceptance of software tools by teachers. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 246–250.
27. Ben-David Kolikant, Y. (2001) Gardeners and cinema tickets: high school students' preconceptions of concurrency. *Computer Science Education* 11(3), 221–245.
28. Ben-David Kolikant, Y. (2004). Learning concurrency: evolution of students' understanding of synchronization. *International Journal of Human-Computer Studies* 60(2), 243–268.
29. Ben-David Kolikant, Y. (2005). Students' alternative standards for correctness. In *Proceedings of the 1st International workshop on Computing Education Research* (ICER), 37–43.
30. Ben-David Kolikant, Y., and Ben Ari, M. (2008). Fertile zones of cultural encounter in computer science education. *Journal of the Learning Science* 18(1), 1–32.
31. Ben-David Kolikant, Y., and Mussai, M. (2008) "So my program doesn't run!" Definition, origins, and practical expressions of students' (mis)conceptions of correctness. *Computer Science Education* 18(2), 135–151,
32. Ben-David Kolikant, Y., and Pollack, S. (2004) Establishing computer science professional norms among high-school students. *Computer Science Education* 14(1), 21–35.
33. Ben-David Kolikant, Y., Ben-Ari, M., and Pollack, S. (2000). The anthropology of semaphores. In *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education*, 21–24.
34. Biggs, J.B., and Collis, K.F. (1982). *Evaluating the Quality of Learning: The SOLO Taxonomy (Structure of the Observed Learning Outcome)*. Academic Press.
35. Brandes, O., and Armoni, M. (2019). Using action research to distill research-based segments of pedagogical content knowledge of K-12 computer science teachers. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 485–491.
36. Breuer, S, Gal-Ezer, J., and Zwas, G. (1990). Microcomputer laboratories in mathematics education. *Computers and Mathematics* 19(3), 13–34.
37. Bruner, J.S. (1960). *The Process of Education*. Harvard University Press. Boston, MA.
38. Caspersen, M., Diethelm, I., Gal-Ezer, J., McGettrick, A., Nardelli, E., Passey, D., Rovan, B., and Webb, M. (2022). *Informatics References Framework for School.* https://www.informaticsforall.org/the-informatics-reference-framework-for-school-release-february-2022/
39. Damm, W., and Harel, D. (2001). LSCs: Breathing life into message sequence charts. *Formal Methods in System Design* 19(1), 45–80.
40. Dijkstra, E. W. (1975). Guarded commands, nondeterminacy and formal derivation of programs. Communications of the ACM 18(8), 453-457.
41. Ericson, B., Armoni, M., Gal-Ezer, J., Seehorn, D., Stephenson, C., and Tree, F. (2008). *Ensuring Exemplary Teaching in an Essential Discipline: Addressing the Crisis in Computer Science Teacher Certification, Final Report of the CSTA Teacher Certification Task Force*. ACM. New York, NY.
42. Floyd, R. W. (1967). Assigning meaning to programs. In *Proceedings of Symposia in Applied Mathematics, American Mathematical Society* 19, 19–32.
43. Friebroon-Yesharim, M., and Armoni, M. (2022). The tale of an intended CS curriculum for 4th graders, the case of abstraction. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE), pp. 623.

44. Gaber, I., Armoni, M., and Statter, D. (2021). Teaching reduction as an algorithmic problem solving strategy. In *Proceedings of the 3rd International Conference on Computer Science and Technology in Education* (CSTE), 19–26.
45. Gagliardi, F., Hankin, C., Gal-Ezer, J., McGettrick, A., and Meitern, M. (2016). *Advancing Cybersecurity Research and Education in Europe: Major Drivers of Growth in the Digital Landscape*. https://www.acm.org/binaries/content/assets/publipolicy/2016_euacm_cybersecurity_white_paper.pdf
46. Gal-Ezer, J., and Harel, D. (1998). What (else) should CS educators know?, *Communications of the ACM*, 41(9), 77–84.
47. Gal-Ezer, J., and Harel, D. (1999). Curriculum and course syllabi for high-school computer science program. *Computer Science Education* 9(2), 114–147.
48. Gal-Ezer, J., and Trakhtenbrot, M. (2016): Identification and addressing reduction-related misconceptions, *Computer Science Education* 26(2–3), 80–103.
49. Gal-Ezer, J., and Zur, E. (2004). The efficiency of algorithms – misconceptions. *Computers and Education* 42(3), 215–226.
50. Gal-Ezer, J., and Zwas, G. (1984). An Algorithmic Approach to Linear Systems. *International. Journal of Mathematics Education in Science and Technology* 15(4), 501–519.
51. Gal-Ezer, J., Beeri, C., Harel, D., and Yehudai, A. (1995). A high-school program in computer science. *Computer* 28(10), 73–80.1
52. Gal-Ezer, J., Vilner, T., and Zur, E. (2004). Teaching efficiency at CS1 level: a different approach. *Computer Science Education* 14(3), 235–248.
53. Gal-Ezer, J. Vilner, T., and Zur, E. (2009). Has the paradigm shift in CS1 a harmful effect on data structures courses: a case study. In *Proceedings of the 40th Technical Symposium on Computer Science Education* (SIGCSE), 126–130.
54. Ginat, D. (2001). Early algorithm efficiency with design patterns. *Computer Science Education* 11(2), 89–109.
55. Ginat, D. (2001). Loop invariants, exploration of regularities, and mathematical games. *International Journal of Mathematical Education in Science and Technology* 32(5), 635–651.
56. Ginat, D. (2002). Effective binary perspectives in algorithmic problem solving. *Journal on Educational Resources in Computing* 2(2), 4–12.
57. Ginat, D. (2002). On various perspectives of problem decomposition. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, 331–335.
58. Ginat, D. (2003). Decomposition diversity in computer science—beyond the top-down icon. *Journal of Computers in Mathematics and Science Teaching 22*(4), 365–379.
59. Ginat, D., (2003). Seeking or skipping regularities? Novice tendencies and the role of invariants. *Informatics in Education* 2(2), 211–222.
60. Ginat, D. (2003). The novice programmers' syndrome of design-by-keyword. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 154–157.
61. Ginat, D. (2008). Design Disciplines and Non-specific Transfer. In *Proceedings of the International Conference on Informatics in Secondary Schools: Evolution and Perspectives* (ISSEP), *Lecture Notes in Computer Science* (LNCS, 5090), 87–98.
62. Ginat, D. (2009). On the non-modular design of on-the-fly computations. *Inroads – SIGCSE bulletin* 41(4), 35–39.
63. Ginat, D. (2009). The overlooked don't-care notion in algorithmic problem solving. *Informatics in Education* 8(2), 217–226.
64. Ginat, D. (2010). The baffling CS notions of "as-if" and "don't-care". In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (SIGCSE), 385–389.
65. Ginat, D. (2014). On Inductive Progress in Algorithmic Problem Solving. *Olympiads in Informatics* 8, 81–91.
66. Ginat, D. (2021). Abstraction, declarative observations and algorithmic problem solving. *Informatics in Education* 20(4), 567–582.
67. Ginat, D., and Alankry R. (2012). Pseudo abstract composition: the case of language concatenation. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 28–33.

68. Ginat, D., and Blau, Y. (2017). Multiple levels of abstraction in algorithmic problem solving. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 237–242.
69. Ginat, D., and Menashe, E. (2015). SOLO taxonomy for assessing novices' algorithmic design. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (SIGCSE). 452–457.
70. Ginat, D., Shifroni, E., and Menashe, E. (2011). Transfer, cognitive load, and program design difficulties. In *Proceedings of The 5th International Conference on Informatics in Secondary Schools: Evolution and Perspective* (ISSEP), *Lecture Notes in Computer Science* (LNCS 7013), 165–176.
71. Ginat, D., Menashe, E., and Taya, A. (2013). Novice Difficulties with Interleaved Pattern Composition. In *Proceedings of the 5th International Conference on Informatics in Schools: Situation, Evolution and Perspective* (ISSEP), *Lecture Notes in Computer Science* (LNCS 7780), 57–67.
72. Gordon, M., Marron, A., and Meerbaum-Salant, O. (2012). Spaghetti for the main course?: observations on the naturalness of scenario-based programming. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 198–203.
73. Green, A., Armoni, M., and Ginat, D. Object-first vs. object-Second. In preparation.
74. Haberman, B. (2002). Frames and boxes – A pattern-based method for manipulating binary trees. *Inroads – SIGCSE Bulletin* 34(4), 60–64.
75. Haberman, B. (2004). High-School students' attitudes regarding procedural abstraction. *Education and Information Technologies* 9(2), 131–145.
76. Haberman, B. (2008). Formal and practical aspects of implementing abstract data types in the prolog instruction. *Informatica* 19(1), 17–30.
77. Haberman, B., and Muller, O. (2008). Teaching abstraction to novices: Pattern-based and ADT-based problem-solving processes. In *Proceedings of the 38th Annual Frontiers in Education Conference* (FIE), F1C:7–12.
78. Haberman, B., and Ragonis, N. (2010). So different though so similar? Or vice versa? Exploration of the logic programming and the object-oriented programming paradigms. *Issues in Informing Science and Information Technology* 7, 393–402.
79. Haberman, B., and Scherz, Z. (2009). Connectivity between abstraction layers in declarative ADT-based problem-solving processes. *Informatics in Education* 8(1), 3–16.
80. Haberman, B., Shapiro, E., and Scherz, Z. (2002). Are black boxes transparent? High school students' strategies of using abstract data types. *Journal of Educational Computing Research* 27(4), 411–436.
81. Haberman, B., Lev, E., and Langley, D. (2003). Action research as a tool for promoting teacher awareness of students; conceptual understanding. In *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 144–148.
82. Haberman, B., Averbuch, H., and Ginat, D. (2005). Is it really an algorithm? The need for explicit discourse. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 74–78.
83. Hazzan, O. (2003). How students attempt to reduce abstraction in the learning of mathematics and in the learning of computer science. *Computer Science Education* 13(2), 95–122.
84. Hazzan, O. (2003). Reducing abstraction when learning computability theory. *Journal of Computers in Mathematics and Science Teaching* 22(2), 95–117.
85. Hazzan, O., and Hadar, I. (2005). Reducing abstraction when learning Graph Theory. *Journal of Computers in Mathematics and Science Teaching* 24(3), 255–272.
86. Hazzan, O., Gal-Ezer, J., and Blum, L. (2008). A model for high school computer science education: the four key elements that make it! In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, 281–285.
87. Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM* 12(10), 576–580.

88. Holz, H. J., Applin, A., Haberman, B., Joyce, D., Purchase, H., and Reed, C. (2006). Research methods in computing: what are they, and how should we teach them? In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education* (ITiCSE-WGR), 96–114.
89. Hubwieser, P., Armoni, M., Brinda, T., Dagiene, V., Diethelm, I., Giannakos, M. N., Knobelsdorf, M., Magenheim, J., Mittermeir, R., and Schubert, S. (2011). Computer science/informatics in secondary education. In *Proceedings of the 16th Annual Conference Reports on Innovation and Technology in Computer Science Education – Working Group Reports* (ITiCSE-WGR), 19–38.
90. Israel National Center for Computer Science Teachers (2002). "Machshava": the Israeli National Center for high school computer science teachers, In *Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE), pp 234.
91. Lapidot, T., and Aharoni, D. (2007). The Israeli summer seminars for CS leading teachers. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE), pp. 318.
92. Lapidot T., Levy D., and Paz T. (1999). Implementing constructivist ideas in a functional programming course for secondary school. In *Proceedings of the Workshop on Functional and Declarative Programming in Education*, 29–31.
93. Lapidot T., Levy D., and Paz T. (2000). Teaching functional programming to high school students. In *Proceedings of the International Conference on Mathematics/Science Education and Technology* (M/SET).
94. Lavy, I., Rashkovits, R., and Kouris, R. (2009). Coping with abstraction in object orientation with a special focus on interface classes. *Computer Science Education* 19(3), 155–177.
95. Leron, U. (1985). Logo today: vision and reality. *The Computing Teacher* 12(5), 26-32.
96. Leron, U. (1987). Abstraction barriers in mathematics and computer-science. In *Proceedings of the Third International Conference on LOGO and Mathematics Education* (LME).
97. Leron, U., and Paz, T. (2014). Functions via everyday actions: Support or obstacle? The Journal of Mathematical Behavior 36, 126-134
98. Levy, D., Lapidot, T., and Paz, T. (2001). 'It's just like the whole picture, but smaller': Expressions of gradualism, selfsimilarity, and other pre-conceptions while classifying recursive phenomena. In *Proceedings of the 13th Workshop of the Psychology of Programming Interest Group* (PPIG), 249–262
99. Lieberman, N., Ben-David Kolikant, Y., and Beeri, C. (2011). Difficulties in learning inheritance and polymorphism. *ACM Transactions on Computing Education* 11(1), 4:1–23.
100. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Ben-David Kolikant, Y., Laxer, C., Thomas, L., Utting, I., and Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *In Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education* (ITiCSE-WGR), 125–180.
101. Meerbaum-Salant, O., Armoni, M., and Ben-Ari, M. (2011). Habits of programming in Scratch. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 168–172.
102. Meerbaum-Salant, O., Armoni, M., and Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education* 23(3), 239–264.
103. Muller, O. (2005). Pattern oriented instruction and the enhancement of analogical reasoning. In *Proceedings of the 1st International workshop on Computing Education Research* (ICER). 57–67.
104. Muller, O., and Haberman, B. (2008). Supporting abstraction processes in problem-solving through pattern-oriented-instruction. *Computer Science Education*, 18(3), 187–212.
105. Muller, O., and Haberman, B. (2009). A course dedicated to developing algorithmic problem solving skills – Design and experiment. In *Proceedings of 21st Annual Workshop of the Psychology of Programming Interest Group* (PPIG), 9:1–9.

106. Nakar, L., and Armoni, M. (2022). Pattern-oriented instruction and students' abstraction skills. In *Proceedings of the 27th ACM Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE), pp. 613.
107. Omar, A., Hadar, I., and Leron, U. (2017). Investigating the under-usage of code decomposition and reuse among high school students: the case of functions. *Lecture Notes in Business Information Processing* 286, 92–98.
108. Or-Bach, R., and Lavy, I. (2004). Cognitive activities of abstraction in object orientation: an empirical study. *Inroads – the SIGCSE Bulletin* 36(2), 82–86.
109. Paz, T., and Lapidot, T. (2004). Emergence of automated assignment conceptions in a functional programming course. In *Proceedings of the 9th Annual SIGCSE Conference on Innovations and Technology in Computer Science Education* (ITiCSE), 181–185.
110. Paz, T., and Leron, U. (2009). The slippery road from actions on objects to functions and variables. *Journal for Research in Mathematics Education* 40(1), 18–39.
111. Perrenet, J., Groot, J.F., and Kaasebrood, E. (2005). Exploring students' understanding of the concept of algorithm: levels of abstraction. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 64–68.
112. Ragonis, N. (2010). A pedagogical approach to discussing fundamental object-oriented programming principles using the ADT SET. *ACM Inroads* 1(2), 42–52.
113. Ragonis, N. (2012). Integrating the teaching of algorithmic patterns into computer science teacher preparation programs. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 339–344.
114. Ragonis, N., and Ben-Ari, M. (2005). A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education* 15(3), 203–221.
115. Ragonis, N., and Hazzan, O. (2009). Integrating a tutoring model into the training of prospective Computer Science teachers. *The Journal of Computers in Mathematics and Science Teaching* 28(3), 309–339.
116. Rubinstein, A., and Chor, B. (2014). Computational thinking in life science education. *PLOS Computational Biology* 10(11), 1–5.
117. Sakhnini, V., and Hazzan, O. (2008). Reducing abstraction in high school computer science education: The case of definition, implementation and use of abstract data types. *ACM Journal on Educational Resources in Computing* 8(2), 5:1–13.
118. Schocken, S., Nisan, N., and Armoni, M. (2009). A synthesis course in hardware architecture, compilers, and software engineering. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (SIGCSE), 443–447.
119. Schwarz, S., and Ben-Ari, M. (2006). Why don't they do what we want them to do? In *Proceedings of the 18th Annual Workshop of the Psychology of Programming Interest Group* (PPIG), 266–274.
120. Schwill, A. (1994). Fundamental ideas of computer science. *Bulletin-European Association for Theoretical Computer Science* 53, 274–295.
121. Sfard, A. (1991). On the dual nature of mathematical conceptions: Reflections on processes and objects as different sides of the same coin. *Educational Studies in Mathematics* 22, 1–36.
122. Shmallo, R., and Ragonis, N. (2020). What is "this"? Difficulties and misconceptions regard the "this" reference. *Journal of Education and Information Technologies* 26(1), 733–762.
123. Statter, D., and Armoni, M. (2017). Learning abstraction in computer science: a gender perspective. In *Proceedings of the 12th Workshop in Primary and Secondary Computing Education* (WiPSCE), 5–14.
124. Statter, D., and Armoni, M. (2020). Teaching Abstraction in Computer Science to 7th Grade Students. *ACM Transactions on Computing Education* 20(1), 8:1–37.
125. Stavy, R., and Tirosh, D. (2000). *How Students (mis-)Understand Science and Mathematics: Intuitive Rules*. Teachers College Press. New York, NY.
126. Stephenson, C., Gal-Ezer, J., Haberman, B., and Verno, A. (2005). *The New Educational Imperative: Improving High School Computer Science Education, Final report of the CSTA Curriculum Improvement Task Force*. ACM. New York, NY.

127. Stolin, Y., and Hazzan, O. (2007). *Students'* understanding of computer science soft ideas: the case of programming paradigm. *Inroads – the SIGCSE Bulletin* 39(2), 65–69.
128. Taub, R., Armoni, M., and Ben-Ari, M. (2014). Abstraction as a bridging concept between computer science and physics. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (WiPSCE), 16–19.
129. Teif, M., and Hazzan, O. (2006). Partonomy and taxonomy in object-oriented thinking: Junior high school students' perceptions of object-oriented basic concepts. *Inroads – the SIGCSE Bulletin* 38(4), 55–60.
130. Utting, I., Tew, A. E., McCracken, M. E., Thomas, L., Bouvier, D., Frye, R., Paterson, J., Caspersen, M., Ben-David Kolikant, Y., Sorva, J., and Wilusz, T. (2013). A fresh look at novice programmers' performance and their teachers' expectations. In *Proceedings of the ITiCSE Working Group Reports Conference on Innovation and Technology in Computer Science Education – Working Group Reports* (ITiCSE-WGR), 15–32.
131. Vahrenhold, J., Nardelli, E., Pereira, C., Berry, G., Caspersen, M. E., Gal-Ezer, J., Kölling, M., McGettrick, A., and Westermeier, M. (2017). *Informatics Education in Europe: Are We All in the Same Boat?* ACM. New York, NY.
132. Vilner, T., Zur, E., and Gal-Ezer, J. (2007). Fundamental concepts of CS1: procedural vs. object oriented paradigm – a case study. In *Proceedings of the 12th Annual ITiCSE Conference on Innovation and Technology in Computer Science Education*, 171–175.
133. Zur-Bargury, I., (2012). A new curriculum for junior-high in computer science. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 204–208.
134. Zur-Bargury, I., Pârv, B., and Lanzberg, D. (2013). A nationwide exam as a tool for improving a new curriculum. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE), 267–272.

# Computing Education Research in the UK & Ireland


**Brett A. Becker, Steven Bradley, Joseph Maguire, Michaela Black, Tom Crick, Mohammed Saqr, Sue Sentance, and Keith Quille**

## 1 Introduction

The December 1970 SIGCSE[1] Bulletin [149] published a member listing recording three members in England, and one each in Scotland and Wales. By 1983, there was at least one member in Ireland [160], and by 1995 these infrequently published

[1] The Association for Computing Machinery (ACM) Special Interest Group on Computer Science Education (SIGCSE) was founded in 1969: sigcse.org/about.

B. A. Becker (✉)
University College Dublin, Dublin, Ireland
e-mail: brett.becker@ucd.ie

S. Bradley
Durham University, Durham, UK
e-mail: s.p.bradley@durham.ac.uk

J. Maguire
University of Glasgow, Glasgow, Scotland
e-mail: joseph.maguire@glasgow.ac.uk

M. Black
Ulster University, Coleraine, Northern Ireland
e-mail: mm.black@ulster.ac.uk

T. Crick
Swansea University, Swansea, Wales
e-mail: thomas.crick@swansea.ac.uk

M. Saqr
University of Eastern Finland, Joensuu, Finland
e-mail: mohammed.saqr@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_19

member listings recorded at least one member in Northern Ireland [225]. Well before then authors in all of these countries had contributed to the growing volume of computing education literature.

In this chapter we present the context of Computing Education Research (CER) in the UK and Ireland, and to examine how the context has shaped the content of that research. The rest of the introduction sets the broad context, including the geographical and political scope and the major CER activities and structures. Section 2 then explores the early history of the discipline here, from the very earliest of days of computing and computing education. Like a group of siblings, for those outside the family there is a strong resemblance between them, but within the family we are perhaps more inclined to notice the differences. Therefore, in the following Sect. 3 we explore the variety within the nations and the different stages of education, in terms of the education systems, and in particular computing education, and other factors that have influenced the development of CER. Having seen where and how the CER community has been formed, we then move on to examine its outputs in Sect. 4, through a scientometric analysis of CER papers produced by authors from institutions in the UK and Ireland. Lastly Sect. 5 discusses our findings and looks towards the future.

## *1.1 The British Isles*

The British Isles are comprised geographically of the islands of Great Britain and Ireland and thousands of other smaller islands. Politically these isles comprise the United Kingdom of Great Britain and Northern Ireland (commonly referred to as the UK), and Ireland (commonly referred to as the Republic of Ireland in this context), along with several smaller entities such as the Isle of Man and the Channel Islands that are largely self-governing. Great Britain itself is comprised of the countries of England, Scotland and Wales. There are five countries with populations over 1 million in the British Isles: England (56m), Scotland (5m), Wales (3m), Northern Ireland (2m), and Ireland (5m), with the first four all being part of the UK. Ireland has been an independent country since 1922. Education within the UK is devolved, with each of the four constituent nations having separate systems and distinct approaches to policy-making [101]. This results in five different, independent, yet broadly similar educational systems in the British Isles.

S. Sentance
Raspberry Pi Computing Education Research Centre, University of Cambridge, Cambridge, UK
e-mail: ss2600@cam.ac.uk

K. Quille
TU Dublin, Dublin, Ireland
e-mail: keith.quille@tudublin.ie

### *1.2 CER Activities and Structures*

In 1998, the third ACM *Innovation and Technology in Computer Science Education* (ITiCSE) conference was held in Ireland. Since then, the countries in the British Isles have worked together in advancing computing education regionally and globally (including major national curriculum and qualifications reforms), hosting numerous ITiCSE and *International Computing Education Research* (ICER) conferences and spawning several influential research projects and groups. The ACM-affiliated Workshop in Primary and Secondary Computing Education (WIPSCE) was held in England in 2015 and Scotland in 2019. The last decade has seen the establishment of two new local annual conferences: the conference in Computing Education Practice (CEP), now in its seventh year; and the UK and Ireland Computing Education Research conference (UKICER), now in its fourth year. In 2022, Ireland again hosted ITiCSE (the 27th) and for the first time hosted UKICER.

The average SIGCSE membership for the decade 2010–2019 for the UK was 53, representing just over 2% of the total membership while Ireland was 8, representing just over 0.25% [26]. Despite these relatively small number of registered members, most of whom are likely researchers, from 2010-present, the UK has contributed 362 outputs to SIGCSE venues. In 2018, the UK & Ireland ACM SIGCSE Chapter was established out of the community that grew around the Computing Education Practice conference. In 2019, driven by the establishment of several Irish university-based computing education research groups, and particularly the establishment of Computer Science as an official national Irish school subject, the decision was taken to split the UK & Ireland ACM SIGCSE Chapter into two—the UK ACM SIGCSE Chapter and the Ireland ACM SIGCSE Chapter. To date, these chapters currently have over 250 members. These chapters work closely and today co-sponsor the UKICER conference. The UK chapter focuses mainly on tertiary education because Computing At School (CAS) (see Sect. 3.1.1) already existed to support computing education in schools across the UK, whereas there was no such body to support schools in Ireland. To identify the foundations of the CER community here, we now look at the historical perspective.

## 2 History: Formation of the CER Landscape

British computing historian Simon Lavington argues that individuals have an inclination to consider computing history either from a bottom-up or top-down viewpoint [138]. Bottom-up, in the sense that progress bubbles-up from academics devising theories and conducting experiments in response to their peers and the scientific community. Top-down, in the sense that industrialists and policymakers allocate funding through a lens of economic development, national defence and educational attainment. Lavington suggests the best insight in terms of how

things developed, is somewhere in-between. The actions and thoughts of countless individuals, organisations and structures reacting to the culture and environment of the time.

Similarly, any conversation or discussion around the present landscape of computing education research in the UK and Ireland must be situated in, or oriented around, the history of the domain in the region. An appreciation of *some* of the trends and milestones that shaped the direction of the domain is important as to provide insight into the emergence of the terrain of computing education research. An important aspect to consider is the source and motivation for funding of computing and computing education research in the UK and Ireland.

## *2.1 Pre-history: Babbage, Boole, Bletchley and Bombe*

The UK and Ireland have made significant historical contributions to the advancement of modern computing, driven in part to being industrial, maritime and trading nations. The importance of maritime activity and advancement drove many initial contributions from the region in computing [68].

Prior to hardware and software solutions, computers were human [217]. The Royal Greenwich Observatory employed a relatively great number of them at the time to produce the British Nautical Almanac. The nautical almanac or "Seaman's Bible" contained any number of mathematical tables that were used by seafarers and others to efficiently and effectively navigate the globe [67]. The accuracy of the astronomical tables very much relied upon the human computers that generated them [66].

The British Victorian polymath Charles Babbage argued that astronomical tables, and many such others, that modern industry relied upon could be computed far more efficiently and accurately, *mechanically*, an idea endorsed by the British Astronomical Society and subsequently funded by the UK Government. Babbage devised and engineered the *Difference Engine* over 10 years. Nevertheless, after 10 years of financial support and no working system, the UK Government withdrew support [112]. Unswayed, Babbage embarked on the design of his second system, the *Analytical Engine*, and spent 15 years producing over 300 engineering schematics for a system that never materialised in his lifetime [106]. Babbage worked closely with Ada Lovelace, who is often credited as being the first programmer, through her publication of an algorithm to calculate Bernoulli numbers [110], designed to execute on the Analytical Engine. She also had a broader vision of the applicability of computers beyond solving mathematical problems, including music and graphics.

Around this time, Babbage encountered George Boole, the first Professor in Mathematics at Queen's College, Cork Ireland. Babbage and Boole did not collaborate on any of the *Engines* but Boole would go on to introduce Boolean Logic, a contribution that to the fundamental foundation for digital electronics and programming languages [32, 33]. Nevertheless, a century later, further UK

Government funding in the form of the UK Government Code and Cypher School (GC&CS) and Alan Turing resurrected Babbage's engine as the Bombe at Bletchley Park, a specialised system designed to support in the deciphering of encrypted messages used by German forces in World War II [63, 218].

Turing joined the National Physical Lab (NPL) after the conflict and started designing a general purpose computer, advancing on the specialised Bombe that was focused on deciphering codes. The eventual system was known as the Automatic Computing Engine or ACE. ACE construction completed after Turing had left the NPL and the system executed its first program in 1950. The system was refined and commercialised by the English Electric Company as the Digital Electronic Universal Computing Engine (DEUCE) and sold for approximately £50,000. A total of 33 systems were manufactured, installed and employed by universities, industries and government.

The first computer laboratory in Scotland was established at the University of Glasgow with the DEUCE at the centre. Similarly, the UK Government Department of Scientific and Industrial Research in Glasgow was equipped with a DEUCE [62]. John Womersley who led the ACE project also recognised the need for a more inexpensive and accessible version of ACE for industry and worked with Andrew Booth to produce the Hollerith Electronic Computer (HEC). The Irish Sugar Company took delivery of one of the first HEC systems at the cost of £33,000 to calculate invoices for sugar beet producers.

It was not only hardware where the UK and Ireland were making contributions. Alick Edwards Glennie worked with Alan Turing on a number of projects and worked at the Atomic Weapons Research Establishment (AWRE) in Cardiff, Wales. Donald Knuth argues Glennie, along with others such as Grace Hopper, were responsible for the first computer compilers [133].

The importance of the aforementioned milestones and contributions is not to argue ownership or suggest the UK and Ireland made exclusive contributions to computing. Many countries and continents made early and significant contributions to computing, that are often under reported and represented [44]. However, the UK and Ireland clearly did perceive computing as powerful and worthy of significant investment. The assumption would be then, just as the region had invested before, it would do so again in the education of its people in utilising such scientific advancement and achievement. Moreover, a reasonable expectation may be that energy and research funding would be spent on trying to understand effective computing education. However, as the reader will come to observe, the focus of such funding is not always obvious or intuitive.

## *2.2 Mind the Gap: The British and Irish Retreat*

The 1960s and 1970s represented a great economic resurgence for the UK and Ireland as they engaged and participated in the global boom that occurred after the Great Depression and Second World War. The UK and Ireland were still making

significant contributions to the advancement of computing in the decade. The Altas Computer, one of the world's first supercomputers, pioneered ideas such as virtual memory, paging and one of the world's first modern operating systems [139]. The system was developed in partnership between academia and industry. Nevertheless, the commercial and industrial computing influence and contribution began to retreat and recede for the region.

The leading British catering company J. Lyons and Co, commissioned the first computer for commercial purposes [135]. The company initially contributed funds to Douglas Hartree and Maurice Wilkes to accelerate their work on the Electronic Delay Storage Automatic Calculator (EDSAC) at the University of Cambridge in advance of funding their own system based on the outcome of the project. Despite, such commercial beginnings for computing in the region, by the 1960s and 1970s it was limited in contrast to the United States as was influence of the region [137].

A partial explanation for this is the contrast in differences between the gap that existed between defence research and commercial endeavours in computing in the UK and United States [136]. In the United States, there was tighter integration and collaboration between parties with the resulting benefits, whereas in the UK there was tighter secrecy and looser connections. As a result, the UK did not reap the same benefits or influence. Frank Cousins, Minister of Technology, announced the formation of the National Computing Centre (NCC) in 1965 with the aim of ensuring the society and business could realise the practical benefits of computers. The NCC funded two universities, Imperial College London and the London School of Economics to address the gap in research and education for the applied use of systems [136].

Despite such investments, the reality was the United States remained far more influential in computing at this point and as a consequence so did its research and investigation into computing education. Guzdial identifies two streams of activity in the 1960s and 1970s [109]: the psychology of programming (driven by industry); and learning of programming in schools. This characterisation was largely true of early work in the UK and Ireland.

Evershed and Rippon argued that the significant investment made by the UK and Ireland in computers in industry, research and government operation would be wasted unless there was recognition that the infrastructure could not be effectively utilised by anyone other than programmers and coders [87]. They argued that high-level languages were required for "low-level users", that in order for professionals to utilise machines, programming their calculations needed to be more efficient than those that could be done by hand. Evershed and Rippon stated that a programming language that supported individuals in minimising errors and was easy to comprehend would not be possible without appreciation of human factors. Similarly, Sime, Green and Guest performed an empirical evaluation of conditional constructs between languages as to determine what would be optimal for "low-level users" [203]. Their work, and the work of others in the UK and Ireland represented the beginnings of a rich community of researchers and practitioners that resulted in

the Psychology of Programming Interest Group (PPIG)[2] and the Empirical Studies of Programming series (ESP) [30].

There were a number of different initiatives that received funding both in terms of software and hardware [180]. The National Development Programming in Computer Aided Learning (NDPCAL) was one of the earliest significant funding programming for exploring the use of computers in education within the UK. The programme funded a number of initiatives, £2.5 million spent over 5 years on 35 projects, across a number of contexts, including industry, defence training, further education, higher education as well as primary and secondary schools [117]. Similarly, the Schools Council funded the Computers in Curriculum project to support schools and teachers in developing and exchanging computer assisted learning materials. However, funding and resources were focused on *terraforming* education with computer software and hardware with little focus on the explicit value of such initiatives.

This is not to say no consideration was given to such concerns. The Nuffield Foundation, the Scottish Council for Research in Education, the Leverhulme Trust and the Social Science Research Council funded Howe and du Boulay to survey the roles of programs in education [119]. Howe and du Boulay identified: *application*, *simulation*, *drill and practice*, *tutorial* and *administration*. Identification of the different roles was significant as it demonstrates the wide spectrum of use of computers and programs. More importantly, Howe and du Boulay argued that due to the wide spectrum and potential use, educators without sufficient insight or appreciation of programs result in using them inappropriately or with negative consequence for learners. Consequently, they argued that computers had significant potential in education, but only through partnership with teachers.

## *2.3 Silicon Fen: Jet Set Willy and Mr Podd*

The 1980s continued the focus in the UK and Ireland to utilise computing infrastructure as well as expand it. In terms of CER, focus was still on individuals making the most of computing and programming. du Boulay et al. considered how to present programming concepts to novice individuals, advocating the concept of a notional machine [34], based on the programming language to be learned rather than specific hardware. The learner learns a BASIC or LISP machine, coming to appreciate the mechanisms to solve problems and the optimal problems they can be used to solve.

Marc Eisenstadt around the same time in the early 1980s was interested in the cognitive models employed by programmers and the design of the tools the utilised. Eisenstadt proposed the SOLO programming language [83] that was designed in part to make the underlying virtual machine explicit and visible to the user [169].

[2] ppig.org.

He devised the language to support students enrolled at the Open University on a cognitive psychology course that had to complete some programming. SOLO was devised with the idea that students (a) did not want to learn programming, (b) they were working remotely in various environments, (c) they did not have significant time to spend on such learning and (d) were not computing literate. It attempted to address these issues in various ways and the benefits of the approach were investigated by researchers.

The Computer Literacy Project (CLP) emerged from the Continuing Education Television department at the British Broadcasting Corporation (BBC) [6]. The foundation of the project was informed by a commissioned report on Microelectronics from Albury and Allen [2]. The aim of the CLP was to prepare British and Irish society so that it could steer technology rather by steered by it [31] and it was designed around successful approaches adopted by a similar BBC Adult Literacy Project. The BBC adopted a mixed economy approach to computing, embracing academia, vocational and cross curricula [103]

Kenneth Baker MP after witnessing the development of computer systems and software in Japan devised a manifesto for technology in the UK with one of the aims being a single computer in every school. Prime Minister Margaret Thatcher appointed Baker as Minister for Information Technology and when the UK was experiencing a deep economic recession in the early 1980s, Baker stated that he gave "*Margaret something nice to say*", which was getting a single computer into every school. Consequently, the Department of Trade and Industry for the UK worked with BBC Engineering to specify the BBC Micro that would later be engineered and manufactured by Acorn computers.

Subsequently, Kenneth Baker devised and deployed the Microelectronics Education Programme (MEP) and Scottish Microelectronics Development Programme. The programmes built on the NDPCAL investment of the 1970s, but with specific focus on schools [224]. Broadly the programmes can be considered as having two territories—(1) using computers in the most effective ways across the existing curricula and (2) introduction of new curricula for such systems: information retrieval, scientific instruments and control technology [100, 104]. The programmes drove widespread deployment of hardware and software into schools, but the programmes were widely criticised for deploying resources without sufficient consideration as to what was optimal for the domain of education. Fothergill, programme director, discussed the balance and challenge of research and deploying computer systems [99]. The challenge or concern was that by the time research delivered results, things would have changed. However, the Social Science Research Council planned a programme of research into microelectronics in education [104].

The concerns around education and the deployment of computers into schools is likely crystallised by the Mr Podd debate. Mr Podd is a character in software that children could instruct various actions, such as *walk* and *run*. A list of actions is not provided to children or teachers as an explicit part of the motivation for children is to learn and engage with vocabulary to get Mr Podd to perform various actions. Thorne argues that despite teachers considering the Mr Podd the best educational software of 1984, the software solution was not borne out of any research or evaluated in

terms of effectiveness [212]. O'Shea and Self, as emphasised by Thorne, argue that research is required both in deploying software and assessing it in terms of its relevance to education [176].

The early 1980s represented a period of excitement and innovation around personal computing with many consumers purchasing the Sinclair ZX Spectrum, the Acorn, BBC Micro and Commodore 64. However, while the personal computing market was vibrant it collapsed within a few years and while there may have been many visions of computers and education, many of the systems were used to play games [140]. The legacy of the movement in the UK was successful at least in terms of the economy and innovation. The City of Dundee became a destination for games development, a history that can be traced back to the production of the Sinclair ZX Spectrum in the Dundee Timex factory [154]. Acorn subsequently developed their own RISC architecture, the Acorn RISC Machine (ARM) which subsequently became known as Advanced RISC Machine and spun off through Arm holdings, now know simply as "Arm", one of the most valuable tech companies in the world.

The period of the 1960s through to the 1990s represented significant investment in computing and education. However, there was less interest, focus or appreciation of the importance of computing education research by policy and law makers. Despite appreciation for the significance and potential for computing by such individuals, there was less concern about refining the methods and infrastructure around computing education. The early computing education research efforts in the UK and Ireland tended to happen around the edges of funded projects or were driven by the interest of a few dedicated individuals.

## *2.4 Devolution: Things Can Only Get Better for Education and Research*

The 1990s did not represent a decade of significant change in the status quo for computing education research in the UK and Ireland. The region had spent considerable resources on computing in the decade prior and it was not clear how it was benefiting from it. However, the 1990s did represent a significant shift in governance for the United Kingdom, that would go on to shape computing education research, in the form of devolution.

In 1999, many significant elements of UK governance in terms of law, management of public services and spending priorities were devolved to institutions in Scotland, Wales and Northern Ireland away from the central UK parliament. The significance of the experiment, depends on the region. Scotland had strong public support and specific ideas whereas the public and politicians in Wales and Northern Ireland were still developing their own perspectives on powers [54]. England, with the biggest population, and Ireland, the neighbouring independent country, were largely unaffected by devolution.

The decisions taken in the different regions are best considered through the lens of *convergence* or *divergence* of governance and approach [102]. The potential impact for computing education from devolution is that the different regions could (1) adopt different approaches to research funding as well as (2) adopt different spending priorities. For example, Wales may diverge initially on some aspects, only to converge on the same approach that are adopted in England later.

In terms of general research funding, allocation of public funds is broadly determined by the Research Excellence Framework (REF), an evaluation that is completed across the UK. Higher Education Institutions are evaluated in terms of research quality and with resources allocated favouring those institutions that produce high-quality research. Strategically, the different regions adopted different approaches to the allocation of actual resources with England allocating based more on quality whereas Scotland and Wales favoured spreading resources more between their institutions. However, over time both Scotland and Wales have converged to adopt a similar approach to England.

Scotland in particular favours research pooling and initiatives to motivate institutions collaborate together. The strategic approach has been successful for Scotland, research impact for Science Technology Engineering and Mathematics in particular has been significant. Scotland also over-performs in securing research funding from the UK Research Council [129]. The Scottish Informatics and Computer Science Alliance (SICSA)[3] is an example of one such research pool. SICSA was launched with £14.5 million from the Scottish Funding Council and supported appointment of 30 academic members of staff across Scottish institutions to improve the research quality of computing science in the region. SICSA was unique as a research pool as its remit was extended beyond research to include education. However, the direct impact of such an expansion on specific computing science education research is less clear.

The other aspect of devolution that has the potential to shape computing science education is spending priorities and initiatives, specifically in school education. Different approaches to computing science education in schools can result in individuals being able to conduct research around the edges of such initiatives. England and Ireland were not impacted directly by devolution, but are *indirectly* impacted by the actions taken by other nations. If Scotland spends more on computing education research, for example, it is unlikely neighbouring nations can simply ignore it—especially if such a decision is successful or reaps significant benefits. Consequently, nations can *converge* on solutions and policies, if they prove optimal in different settings. In terms of the specific impact of devolution, it is how the nations diverge that may lead to interesting outcomes [43], and is the differences between nations that are the focus of the next section.

[3] www.sicsa.ac.uk.

## 3 Computing Education Research Context

Section 2 demonstrated some of the central trends and milestones that informed the present-day computing education research context in the UK and Ireland. Here we outline how recent developments have differentiated the computing, education and research contexts of each of the nations.

### *3.1 England*

We separate our presentation into a consideration of compulsory school education (referred to elsewhere as K-12) in Sect. 3.1.1 and post-compulsory education in Sect. 3.1.2.

#### 3.1.1 Schools in England

England has a long history of computing in school dating back to the 1970s and 1980s [42, 46]. Personal computers such as the BBC micro and ZX Spectrum in school, the use of Logo in mathematics to teach coding, and schools examinations at approximately 16 and 18 years of age (called GSCE and A-Level) in Computer Science or Computer Studies dating from the 1970s [46] all meant that there were opportunities for *some* children to learn some computer science and programming.

A significant change came to England when the Education Reform Act 1988 defined what all children should learn and the concept of a National Curriculum was born [226]. In 1988 a National Curriculum was introduced for schools in England (and initially Wales until devolution in 1999), through the Education Reform Act. The National Curriculum established information technology within the curriculum which needed to focus on building basic computer literacy skills, although it did touch on some aspects of computer science:

> *"While it is not envisaged that all pupils would undertake the detailed study of a programming language they should understand the concept of a computer program as a set of instructions. This understanding can be promoted by the use of certain drawing or control packages where a sequence of moves can be 'saved up' and executed together. The contribution of particular instructions to the whole can be examined without discussing in detail the underlying algorithm. Some pupils will have acquired a detailed knowledge of programming by using computers at home or by specialist study at school."* [114, p.26]

The Dearing review [73] led to an overhaul of the National Curriculum with a new version published in 1999 [122]. Information and Communication Technology (ICT) was statutory in schools for all children 5–16 (Key Stages 1 through to 4) from 2000 [77] but it was difficult to find any aspects of computer science in it. What this curriculum did was move the focus to an ICT literacy for all students, away from principles of computer science. A revision of the curriculum in 2007 did not change

this focus, and so, within a few years of that, we began to hear a call to bring back the "computer studies" element which had been lost from the curriculum [64].

The transition from ICT to Computing in the curriculum in England has been well documented and was informed by CER [42, 43, 222]. England introduced a new computing curriculum to schools in 2014, bringing mandatory computer science to all state-school pupils aged 5–16. The Royal Society, through an influential report, had redefined computing as having three elements: information technology, digital literacy and computer science [210]: this was a useful distinction to aid in this transition but is now outdated. At the time of writing, England has 7 years of experience of the implementation of computing in school, which has presented both exciting opportunities and some tough challenges.

**Creation of a National Centre for Computing**

In 2018, following another Royal Society report describing computing in England as "patchy and fragile" [211, p.6], the Department for Education in England awarded a contract for over £80 million for a 4-year programme of development of teacher training and student resources in computing, called the National Centre for Computing Education (NCCE) [199]. This represented one of the most substantial moves towards educating all children in the discipline of computing in the world. The NCCE provided professional development for almost 30,000 teachers in its first 2 years of delivery[4] and has enabled full curriculum resources, support on pedagogy, and a comprehensive in-service teacher education offer to be provided, free of charge to teachers. England is one of the only countries that provides mandatory computing in the curriculum for all children from age 5 upwards [222].

A recent report by the Brookings Institute comparing computer science education around the world highlighted seven policy actions that a country should undertake to bring computer science to young people effectively [222] with England being the only country to have implemented them all. These include: introduction of ICT education programs; requirement for CS in primary education; requirement for CS in secondary education; introduction of in-service CS teacher education programs; introduction of pre-service teacher education programs; availability of a specialised centre or institution focused on CS education research and training; access to regular funding allocated to CS education by the legislative branch of government. England has undertaken all these policy actions which has made it a useful comparison point for many other countries wishing to introduce CS into the formal school curriculum [222].

**Focus on Delivery: Not Pure Research**

Throughout this period, developments in England have been facilitated by different stakeholders working together to advocate for the importance of computing in

[4] static.teachcomputing.org/NCCE_Impact_Report_Final.pdf.

school. Computing At School (CAS) was set up in 2008 [42] and brought together industry, academia, education professionals and schools to campaign for a more CS-focused curriculum [43]. The current large-scale initiative in computing education, the NCCE, is run by a consortium of three organisations, the BCS, of which CAS is a part, the Raspberry Pi Foundation, and Stem Learning, showing the importance of collaboration and involvement of multiple stakeholders. However the characteristic of developments in England are that while considerable funding has been made available for delivery of professional development and creation of resources, there has been no corresponding funding for computing education *research*, and even rigorous evaluation of the aspects of the programme has not been a priority for the government. This could be seen as a lost opportunity given the huge numbers of young people currently studying computing in school on a daily basis in England, and there is an urgent need to understand better how and what to teach. Steadily the numbers of individual researchers and doctoral students studying computing education for young people have started to grow in England, but without a significant pot of funding. This is in contrast to, for example, the US, where the NSF and other statewide initiatives have provided specific and generous funding avenues for K-12 CS Ed research over the last 5 years.

### 3.1.2 Further & Higher Education in England

**Further Education: The Cinderella**

Within the UK and Ireland, Further Education (FE) is understood as post-school education which is not Higher Education (HE) i.e. it doesn't lead to the award of a degree—similar to continuing education in the USA or TAFE in Australia. The focus of FE colleges is in vocational training, including apprenticeships, and also in access courses for HE, with returners to education an important focus. FE is often referred to as a "Cinderella" service that, according to the influential 2018 Augur Review of post-18 education in England [9], has suffered "decades of neglect and a loss of status and prestige amongst learners, employers and the public at large" …"despite widespread acknowledgement that this sector is crucial to the country's economic success". Sadly, this neglect carries over into the realm of Computing Education Research, with very little attention focused on FE.

**Universities in England**

Higher education (as we now call it) in England started in 1096 at Oxford, followed by Cambridge in 1209, making them some of the most ancient universities in the world. There were no more new universities in England from then until the 1830s, with the founding of Durham and London universities. The start of the twentieth century saw large scale expansion in the "red brick" civic universities in Birmingham, Manchester, Liverpool, Leeds, Sheffield and Bristol between 1900 and 1909. Another group of universities were founded in the post-war period 1948–1957, developing from local university colleges working towards exams from London University. The 1960s saw a further doubling of the number of universities,

some based on existing institutions, but many (the "plate glass" universities) were entirely new, starting with the University of Sussex in 1961 and culminating in 1969 with the Open University, the UK's only university dedicated to distance learning—and having by far the largest student enrolment. The last step change in the number of universities came in 1992, when nearly all of the existing polytechnics became universities in their own right, having previously used the degree awarding powers of the Council for National Academic Awards (CNAA). These "new" or "post-92" universities developed research interests where they previously had mainly focused on teaching and with that developed a much stronger academic community exploring subject-specific pedagogy—such as CER. The twenty-first century has seen a steady stream of institutions newly gaining university status.

**Quality and Funding of HE Teaching**

The vast majority of English universities are public, in that they receive some funding from the government. One of the main differences in policy for universities in different parts of the UK relates to funding. From 1962 to 1998 full-time students were exempt from tuition fees, and also had access to a means-tested maintenance grant. Following the Dearing report [72] (not to be confused with the 1994 Dearing review of school curriculum), student fees were introduced, along with a system of government-backed loans for paying these fees, and for covering living expenses of students (maintenance loans). The level of these fees increased over time to a maximum of £9250 currently, as universities have become more dependent on student fee income as opposed to direct funding of teaching through the Higher Education Funding Council for England (HEFCE) and then the Office for Students (OfS) since 2018. From 2000 the HE sector has become increasingly marketised, although there is virtually no differentiation on price between institutions—the competition has really been on attracting student numbers. Established in 2005, the National Student Survey (NSS) has been an important metric for universities, very often used in published league tables. It provides half of the data points for the Teaching Excellence Framework (TEF), first introduced in 2017. However this has been far from controversial, with the National Union of Students (NUS) at one stage voting to boycott the NSS because of the link to TEF and marketisation of HE.

There is stark contrast between policy, and to some extent research interest, on education in schools and universities. Schools are tightly managed on the academic performance of their students in public examinations, and in particular on the progress they make. In universities, the focus is much more on customer (student) satisfaction and on graduate employability. Indeed, the idea of "learning gain",[5] the HE equivalent of progress measures in schools, is considered experimental and controversial, certainly within the context of TEF. Universities are therefore not incentivised to ensure their students learn a lot, but rather to make them satisfied and employable. Standards within degrees are monitored by the Quality Assurance Agency (QAA), commissioned by the Office for Students (OfS), along

[5] www.officeforstudents.org.uk/advice-and-guidance/teaching/learning-gain.

with the system of external examiners (first used by Durham University to ensure comparability with Oxford), but the issue of "grade inflation" has become an important one.[6] Even aside from the maintenance of standards, educators are often concerned with assessment, not least because scores for Assessment and Feedback are usually amongst the lowest of all the measures within the NSS.[7] For these reasons it is typically much easier for educators to analyse and report on anonymised student opinion of teaching, through module evaluation questionnaires designed to mimic the NSS, than looking at individual student understanding or progress.

**Quality and Funding of HE Research**

UK Research and Innovation (UKRI), which took over research funding responsibilities from HEFCE, allocates research funding to universities on a recurrent formula basis through Research England and also through competitive grant funding awarded by the research councils. A large portion of the recurrent funding to universities is quality-related (QR), as identified by the UK-wide Research Excellence Framework (REF), previously known as the Research Assessment Exercise (RAE). The research outputs (typically papers), impact and research environment of a university are assessed by panels covering different Units of Assessment on a semi-regular basis, with the most recent assessment points in 2021, 2014 and 2008. The issue for CER is that there are separate REF sub-panels for "Computer Science and Informatics" and "Education", so there is no natural home for CER research to be assessed: papers might be seen as "not real computer science" by one panel and "not real education" by another.

A similar situation exists within the funding councils that award research grants. Computing research (termed ICT: "information and communications technologies") falls within the remit of the Engineering and Physical Sciences Research Council (EPSRC) whereas education comes under the Economic and Social Research Council (ESRC). Although they have similar acronyms, they are very distinct, and neither funds computing education research projects.

Overall this leads to a research context which is largely unfunded, and based on the interests of practitioners. Sometimes industry has funded CER in England, notably the BlueJ project [134] largely funded by Oracle. This project was led by Michael Kölling, one of two England-based recipients of the SIGCSE Award for Outstanding Contribution to Computer Science Education. Despite substantial industrial funding, BlueJ never received government research council funding. Sally Fincher, the other recipient of the SIGCSE Outstanding Contribution award also never received any government research council funding, despite major contributions including the Cambridge Handbook of Computing Education Research [95]. None of this was for want of trying, but rather because the funding councils did

[6] www.officeforstudents.org.uk/news-blog-and-events/press-and-media/grade-inflation-remains-a-significant-and-pressing-issue-new-ofs-analysis.

[7] www.officeforstudents.org.uk/advice-and-guidance/student-information-and-data/national-student-survey-nss/nss-data-overview.

not consider it within their remit. The EPSRC has awarded grants for outreach and engagement within computing [84, 85] but not for pedagogical research directly.

**Focus on Employability**
Employment prospects for graduates are a key measure for success of HE courses, being routinely included in published league tables and TEF scores. One particularly paradoxical issue for computing degrees has been the reported shortage of skills in graduates, relatively low popularity of the subject area and high rates of unemployment amongst computing graduates. This issue was addressed in the influential Shadbolt review [201] of computer science degree accreditation and graduate employability, which found that "the supply of Computer Sciences graduates, and the needs of employers appears in some way misaligned". A complex range of factors came into play, and recommendations included: extending work experience; improving graduates foundational knowledge and softer skills; better understanding the needs of startups and SMEs; better engagement with accreditation by both industry and HEIs.

The UK Government introduced the Apprenticeship Levy in 2017. The levy required employers with an annual pay bill that exceeds £3 million pounds to pay an additional 0.5% levy or tax on their pay bill. The levy is then transferred to an account and supplemented with additional 10% contribution from the UK Government. Employers then have 24 months to spend the funds in their account on appropriate training programmes. This, combined with the introduction of degree-level apprenticeships, has led to many new computing degree programmes based on the apprenticeship model of work experience and part-time study, and in turn to CER in the area of curricula and pedagogy for apprenticeships, quite distinctive to the UK (although related to the idea of cooperative education in the USA [120]). Universities have also engaged with the Institute of Coding, a "collaborative national consortium of industry, educators and outreach providers" established in January 2018, with £20 million in funding from the Office for Students. As with government initiatives in schools, these do not directly fund CER, but CER often naturally follows this kind of funded activity.

## *3.2 Northern Ireland*

With the introduction of the Good Friday Agreement in 1997, a devolved administration in Northern Ireland (NI) had now two Ministers of Education—one with responsibility for the school sector and the other for further and higher education. The Good Friday Agreement provided a North-South Ministerial Council to discuss educational matters of interest between Dublin and Belfast. Practically, policy would either take into account UK directives but interpret and apply these around the special circumstances in NI, or be driven by NI's particular needs [198]. The Department of Education for NI (DENI) aims to promote the education of the NI people to ensure the effective implementation of education policy. DENI are now

supported by this one non-departmental sponsored public body CCEA. CCEA is NI's educational awarding organisation for a range of qualifications. As part of this they advise the DENI on matters concerned with the curriculum.[8]

### 3.2.1 The Northern Ireland Curriculum

Northern Ireland's constitutional position in UK has meant that government policy in education in Northern Ireland has often followed initiatives taken by the Department for Education in England and Wales. The statutory curriculum in Northern Ireland began with the Education Reform Order in 1988. This stated the curriculum for a grant-aided school included Science and Technology.

The curriculum itself was introduced from 1991. Shortly after, statutory teacher assessment began at the end of Key Stages 1 (Year 4) and 2 (Year 7), mainly for English and Maths. It was found in practice to be overloaded, so in 1996 it was significantly revised removing a large amount of content but unchanged in structure. In 1999 the then Education Minister gave permission for The Council for the Curriculum, Examinations and Assessment (CCEA) [53] to undertake a fundamental review of the statutory requirements of the curriculum. Evidence within this review of primary school showed that within NI there are well documented differences between high and low attaining children linked to social deprivation and to gender [53]. Data gathered from young people in NI showed that ICT was top of their agenda [198].

The resulting 2002 curriculum proposals focused on a range of skills including critical and creative thinking skills, including managing information, problem solving, and ICT. The revised Northern Ireland Curriculum was introduced in 2007 and implemented over a 3 year period, covering all 12 years of compulsory education.

The DENI's empowering Schools Strategy for ICT [219] focused on transforming education by 2020 with strategic deliverables for 2008. The overarching aim of this strategy was "that all young people should be learning, with, through and about the use of digital and online technologies". The strategy's key focus was the deployment of digital, multimedia and communication technologies to "enhance, improve, and ultimately to transform, education".

While ICT is included in the curriculum as a cross-curricular skill and relates to using software in school, schools have some flexibility to include teaching coding. However, there is evidence that coding is rarely taught in primary schools or key stage 3 [181]. CCEA introduced the A/AS level in Software Systems Development in 2015/16, the A/AS level and GCSE in Computing in 2017 and Digital Technology in 2018.

In 2021, Calder [49] provided a British Computer Society (BCS) landscape review on computing qualifications in the UK. In NI, ICT remains the main

[8] www.legislation.gov.uk/nisi/1998/1759/contents/made.

qualification of computing education across key stage 4 and post-16. However, there has been growing uptake of A-Level Computing. The last 5 years have seen a drop of around 50% in ICT entries for all qualification levels being replaced with growth in Digital Technology topic. A/AS level in Software Systems Development (SSD) has seen a decline in uptake since it became part of the curriculum, with just over 200 students taking this in 2020/21. Female: male participation rates show ratios of 1:2 studying ICT, 1:9 in GCSE computing and 4:1 at A-Level. In 2018–19 A-Level computing's popularity as one of the nine STEM subjects remains second least popular and in 2020–21 only 3% of A-Level cohort taking computing.

Matrix (NI Science Industry Panel)[9] commissioned a positioning paper in 2018 on Women in STEM in NI. The issue of STEM skills shortages continues to be prioritised as a barrier to growth in NI science and technology sectors. This work highlighted the continuing significant gender imbalance across the STEM skills pipeline as a major contributing factor. In 1999, 11,943 boys and 11,104 girls were born in NI, in 2014/15 87.6% of the girls took STEM GCSEs, compared to 91% of the boys. However, when it came to core STEM A-levels or FE vocational exams in 2016/17, only 31% of girls took one, in stark comparison to 85% of boys who took one. For NI futures the decline in girls participating in STEM between GCSE and A-level/FE is anticipated to be 65%, compared to a 6% drop off for boys.

### 3.2.2 Higher Education in Northern Ireland

NI has three universities (Queens University, Ulster University and Open University), two university colleges (St Mary's University College, Stranmillis University College), six further education colleges (Belfast Metropolitan, Northern Regional, Southern Regional, South Eastern, North West Regional, South West), and College of Agriculture, Food and Rural Enterprise (CAFRE) an agri-food and land-based college with 3 campuses, all of which offer opportunities to study for various higher education qualifications.

To try to meet the growing needs of NI's computing industry in 2010, Queens and Ulster designed and offered a one-year conversion masters to students who had completed a non-computing undergraduate course. These courses were immediately very popular, particularly with females, and employment figures for the graduates were very high.

### 3.2.3 Growing Computing Opportunities in NI Moving Forward

In May 2021 DfE published their new economic vision for NI for consultation.[10] It sets out the key themes and proposed commitments for a new Skills Strategy for

[9] matrixni.org/wp-content/uploads/2018/05/Women-in-STEM-Report-final-20-may.pdf.

[10] www.economy-ni.gov.uk/publications/10x-economy-economic-vision-decade-innovation.

Northern Ireland: Skills for a 10× Economy. The Strategy sets the strategic direction for the development of Northern Ireland's skills system to 2030. The primary challenge for Northern Ireland is to increase the number of individuals entering the labour market with qualifications in STEM, particularly in the "narrow STEM" fields: physical, environmental and computer sciences; engineering; and mathematics. Other projects feeding into this strategy are: joint DE/DfE "Transition of Young People into Careers (14–19) Project"; challenges in understanding and addressing declining participation in level 4 and 5 education[11] with the ongoing work on the review of HE in FE; and an "Independent Review of Education", announced by the Minister of Education in December 2020. One of DfE's strategic goals for the new Skills Strategy includes increasing the proportion of individuals leaving Northern Ireland higher education institutions with degrees and post-graduate qualifications in "in-demand" STEM subjects, including computer sciences.

Following the popularity of the MSc conversion courses, funding from the Northern Ireland Office and the Department of Finance offered support to a wide range of free short courses, delivered by the local FEs and HEIs. DfE have already funded up to 7000 free places, with many more to come.[12] Courses are offered on a range of digital skills including: applied cyber security, artificial intelligence, computer science, data engineering, data science, and software testing.

Extensive initiatives are being continually developed across NI with FE and HE providers, industry and STEM partners to grow and inspire young people at all levels to consider a future in computing, and particularly grow the number of females. These include campaigns such as Bring I.T. On,[13] engaging with partners on education policy, curriculum and content development, developing STEM engaging learning and STEM competitions with CCEA, Computing at School,[14] Matrix-NI,[15] BCS-NI[16] and Sentinus.[17]

An example of this comes from 2016 Digital ICT Report published by Matrix-NI[18] which identified four areas in which NI was already, or had the potential to be, world class: software engineering, advanced networks and sensors, data analytics and cyber security. Within this they also noted five sectors which had already been identified as key drivers of the NI economy that stand to benefit from advancements in AI. Matrix then commissioned The Alan Turing Institute to undertake a review of AI capabilities in NI. The report concluded the need for a single AI Centre of Excellence (AiCE@NI), which brings together the best of NI research and

[11] www.gov.uk/what-different-qualification-levels-mean/list-of-qualification-levels.

[12] www.nidirect.gov.uk/skillup#toc-4.

[13] BringITonNI.co.uk.

[14] www.computingatschool.org.uk.

[15] matrixni.org.

[16] www.bcs.org.

[17] www.sentinus.co.uk.

[18] matrixni.org/wp-content/uploads/2019/06/Artificial-Intelligence-Research-in-Northern-Ireland.pdf.

commercialisation, and provides a strategic focal point for internal and external NI AI activity.

In line with a recommendation from Calder [49], in 2021 BCS-NI supported a new committee BCS: Northern Ireland Computing Education Committee (BCS-NICEC) to facilitate communication between interested parties in computing education in Northern Ireland. This will include primary and secondary level school teachers, award and regulatory bodies, higher and further education staff, industry, government departments and learned societies. As part of this work BCS have also supported the formation of a Young Persons Advisory Board to enable a student voice in computing education. This group is modelled on Ulster University's Community Of Practice engaging with student groups and a model from the BCS Scottish Computing Education Committee to get young peoples input into designing and deploying methods of collecting the student voice.

### *3.3 Scotland*

The majority of computing education research in Scotland is directed and led by Higher Education Institutions (HEIs) in partnership with schools and colleges. In the context of Scotland, 14 HEIs are active in computing science research and education as evident in membership of SICSA.[19]

Similar to other nations in the United Kingdom, Scottish HEIs have a mixture of singleton researchers focused on computing education research as well as research groups. There are a number of such research groups emerging in Scotland. The Centre of Computing Education Research at Edinburgh Napier University is focused on employability and pedagogy research.[20] The Engineering and Computing Education Research Group at Glasgow Caledonian University is focused on a number of research areas, including assessment design and feedback.[21] The Centre for Computing Science Education at the University of Glasgow[22] is led by the School of Computing Science at the institution but represents an interdisciplinary partnership between many different areas.

Scottish institutions and academics have acted as hosts, programme chairs as well as leads for doctoral consortia, works in progress workshops and working groups for a number of leading computing education venues. Venues include the Innovation and Technology in Computer Science Education (ITiCSE) conference[23] in 2019,

[19] www.sicsa.ac.uk.

[20] www.napier.ac.uk/research-and-innovation/research-search/centres/centre-for-computing-education-research.

[21] www.gcu.ac.uk/aboutgcu/academicschools/cebe/research/researchgroups/engineering-and-computing-education-research-group-ecerg.

[22] www.ccse.ac.uk.

[23] iticse.acm.org.

the International Computing Education Research (ICER) conference[24] in 2014, as well as the Workshop in Primary and Secondary Computing Education (WiPSCE)[25] in 2019. Similarly, Scottish institutions and academics have performed the same service for national venues including the United Kingdom and Ireland Computing Education Research (UKICER) conference[26] in 2020 & 2021, and the Computing Education Practice (CEP) conference[27] in 2022.

Academics in Scottish institutions regularly contribute to many computing education venues, including the ACM Transactions on Computing Education (TOCE) journal [81], the ICER conference [214], the British Journal of Educational Technology journal [82], the Computers & Education journal [11], the WiPSCE conference [195], the Koli Calling conference [130], the Computing Science Education journal [29] and the Technical Symposium on Computer Science Education [178]. Furthermore, some academics in Scottish institutions have made significant and notable contributions to the computing education community. McGettrick from the University of Strathclyde received the Association for Computing Machinery's Karl V. Karlstrom Outstanding Educator Award. Purchase has been recognised as a contributing author to a significant Working Group paper [150]. Similarly, Cutts has been recognised as a contributing author to a Top 10 Working Group paper [182].

Moreover, researchers from Scottish institutions have made steady and notable contributions in recent years to the computing education landscape with research papers that have received either honourable mention [125, 213] or best paper recognition [69, 124, 177]. It is clear that Scotland has a vibrant and diverse computing education research community that regularly contributes to the community.

However, beyond typical and traditional computing education research areas, it would be reasonable to argue that computing education research output in Scotland is influenced indirectly by state funded educational initiatives. That is to argue, that such state funding is often not directly targeted or related to computing education research but some other aspect of education or training, that permits research around the edges. There are two areas that demonstrate the indirect influence on computing education research outputs from national funding and initiatives after devolution in the UK.

The first area is the *Curriculum for Excellence*, the national curriculum for Scotland. Scotland's national curriculum was introduced from 2010 onward after a consultation exercise conducted by the Scottish Government. It was introduced with the aim of shifting focus away from facts and knowledge to skills and competencies. The introduction of the national curriculum was significant for computing education as it cemented the position of computing in Scottish education prior to age 14 [123]. However, there were concerns about how teachers achieve the outcomes and experiences for computing education at this early age level [148]. The national

[24] icer.acm.org.

[25] www.wipsce.org/2022.

[26] www.ukicer.com.

[27] cepconference.webspace.durham.ac.uk.

curriculum introduced an opportunity for computing education research in Scotland in terms of funding and the opportunity to integrate and investigate computing education research at scale. We consider two examples from this area, Haggis the reference programming language for national assessments, Sect. 3.3.1, and PLAN C, the personal learning network for computing teachers, Sect. 3.3.2.

The second area is *Skills in Higher Education*. Skills Education was another development devolution in the UK that afforded an opportunity for computing education research in Scotland. The UK Government introduced an Apprenticeship Levy with the aim of strengthening skills education, including in HEIs. Scotland delivered Graduate Apprenticeships (GAs) as a result and this in turn provided opportunities in funding and resources around designing appropriate software engineering programmes for higher education. We consider one example from this area, a research-informed reference design for an apprenticeship programme in Software Engineering, Sect. 3.3.3.

### 3.3.1 Haggis Reference Language for School-Level Assessment

The qualification authority commissioned the development of a high-level reference programming language in 2010 as the means to effectively examine computing science assessment outcomes in Scotland. The 1980s had created a diverse zoo of systems and programming languages throughout Scotland and as a consequence, educators were permitted to internally assess using their programming language of choice. Prior to the national curriculum, national assessments relied upon pseudo-code, an informal blend of formal and natural language. The approach introduced instability and ambiguity as the pseudo-code altered from year to year with each assessment. Moreover, the overall approach is focused more on writing programs rather than understanding programming language code, an approach that is sub-optimal to support learning and teaching of formal languages. Consequently, the educational rationale was to ensure a *consistent* pseudo-code that supported the assessment of programming rather than writing programs [159].

The solution was Haggis, a bespoke pseudo-code for assessment developed by computing education researchers in Scotland and tailored specific to the Scottish context. The reference language had to be adaptable to programming languages taught in the Scottish curriculum as well as be sufficiently complex to support assessment from early years to advance qualifications. The aim of Haggis was to support rigorous assessment of core computing concepts and topics with research still ongoing to determine if this is the case.

### 3.3.2 Professional Development of School Teachers in Scotland

The national curriculum cemented or established computing education in one form or another across school education in Scotland. The approach is not unique as many countries around the world, including the comprising nations of the UK as

well as Ireland, have also prioritised the introduction of computing across school education. For example, the United States CS10K initiative aimed to have 10,000 teachers in 10,000 high schools delivering computing curriculum by 2015 [8, 40]. The task is not without concerns and just as other countries have experienced, there are significant challenges for teachers in rapidly introducing computing education [227]. The areas of particular concern are (1) professional practice and (2) pedagogical content knowledge.

For professional practice the concern is that computing educators, like many educators, are lone individuals within a school environment. Consequently, they have limited opportunity to discuss and debate their professional practice within the context of their specialised domain, i.e. computing. For pedagogical content knowledge, many computing educators within Scotland and elsewhere have a limited background in computing education. Furthermore, given the limited opportunity to discuss and debate professional practice with other computing educators in their context, they likely have weak or deteriorating pedagogical content knowledge [202].

Therefore, improving the continuing professional development of teachers is an important tool in delivering on objectives to introduce computing education across school education. Sentance at al. explored the use of community of practices for teachers [200] and Fincher et al. surveyed many different models that could form the basis of improving the professional practice of teachers [94]. Disciplinary Commons [209] is one such approach that has been used in higher education but can also be valuable for school educators.

In Scotland, the Scottish Government funded the project Continuing Professional Development for Teachers of Computing Science. The outcome was the Professional Learning Network for Computing (PLAN C). The professional development programme was largely designed around the idea of Disciplinary Commons where computing educators could meet and discuss computing research appropriate for deployment in practice, prior to use and after it. The network comprised of a number of communities of practices that spanned across Scotland guided by lead teachers. The approach required identification and training of lead teachers also using a Disciplinary Commons approach. PLAN C was subsequently evaluated and deemed to be successful with at least half of potential computing teacher candidates engaging with at least one PLAN C session. For those participants surveyed the majority deemed the solution a positive impact for teachers and students [70].

### 3.3.3 Scottish Industry Partnership Programmes for Higher Education

The Apprenticeship Levy, introduced in 2017, is UK wide and so is payable by all employers regardless of where they reside. However, Skills and Education is a devolved matter, see Sect. 2.4, and so the implementation of how the Apprenticeship Levy is accessed and utilised depends on the nation.

In the first year of collection, Scotland received £221 million pounds from the Apprenticeship Levy. The Scottish Government decided to fund a number

of initiatives in response to the allocation of the budget, including Graduate Apprenticeships (GAs). A Graduate Apprenticeship (GA) is essentially a 48-month programme delivered in partnership between HEIs and industry partners to the eventual attainment of a degree. The GA programme is administered by the national skills agency, Skills Development Scotland (SDS). The agency provided a select number of GA specifications developed in partnership with academia and industry. Employers and HEIs then form partnerships and agree to shepherd a number of students or employees through degrees, designed to approved specifications.

The concern in Scotland, particularly for research-led institutions, was the lack of familiarity and experience in delivering apprenticeship-style education in partnership with employers in the context of higher education. Skills Development Scotland funded different research and development projects around the specifications so that industry partners and employers could further refine delivery plans.

Maguire and Cutts [146] report on one such project that researched and developed the design of a GA programme in partnership with industry to deliver professional Software Engineers. The investigation outlines research into the history of cooperative and apprenticeship-style education in the context of higher education. They also outline case studies from Germany, Ireland and Canada that involved institutional visits and interviews with stakeholders involved in the delivery of apprenticeship-style programmes in higher education. Maguire and Cutts devise a number of principles to inform the design of apprenticeship-style programmes in higher education.

### *3.4 Wales*

Wales is a small nation to the west of England, with a rich and distinct history, grounded in a Celtic cultural identity and the Welsh language (*Cymraeg*, alongside English as one of the two official languages), with 29.1% of the population able to speak Welsh. Its south coast became pre-eminent during the UK's industrial revolution due to extractive mining and metallurgical industries, as well as associated heavy industries, transforming the country from an agricultural society into an industrial nation. Outside of the major population centres in the south and north of the country, Wales is largely rural and mountainous, and suffers from post-industrial socio-economic challenges, seasonal employment focused on the tourism industry, and the dependence on the public sector for a significant proportion of jobs. Wales also faces issues regarding inequality; almost a third of children live in poverty and its proportion of employees who are the lowest-paid is the highest in the UK. Overall, the poverty rate has been higher in Wales than for England, Scotland, and Northern Ireland in each of the last 20 years. Prior to the UK's exit from the European Union at the end of 2020, the majority of the country (apart from the south-east corner, including its capital city Cardiff, and the regions bordering England) had historically been designated by the European Union as so-called "Convergence areas", meaning the per-capita GDP was less than 75% of

the European Union average, making it eligible for a range of European strategic funding initiatives, resulting in large investments in skills and infrastructure.

Education in Wales has historically developed along similar lines to that of England, with UK legislation largely having force in both countries; especially following the establishment of the National Curriculum from the Education Reform Act 1988. In 1997, Wales held a referendum which determined the desire for self-government, leading to the Government of Wales Act 1998, which created the National Assembly for Wales—to which a variety of powers were devolved from the UK parliament on July 1, 1999. In particular, education—which until then was a UK-wide government portfolio (minus Scotland, which for historical reasons, has had a distinct legal and education system from England and Wales)—came under the control of the National Assembly for Wales (now, *Senedd Cymru* or Welsh Parliament). Now, the Welsh Government has control over education policy, teachers' pay and conditions through the Welsh Parliament, although the UK Government still retains control of certain areas, such as teachers' pensions. Education in Wales has developed a distinct identity, with education policy and the wider Welsh education system increasingly diverging from policies and practices in England. This is set to continue with the major education system-level reforms currently taking place at the time of writing, including significant changes to the national curriculum, assessment and qualifications.

### 3.4.1 Schools in Wales

Prior to devolution in 1999, the education system in Wales was essentially identical to that in England and was in a healthy state, outperforming other regions in the UK in the years prior to and immediately following devolution. However, ever since devolution saw the education portfolio transferred to the National Assembly of Wales, it has suffered a decline, as measured by key international measures such as the OECD's Programme for International Student Assessment (PISA).

Whilst broadly maintaining the general educational system used in England, the Welsh Government embarked on a 10-year revolutionary plan including the introduction of the Welsh Baccalaureate, an overarching qualification with a purely practical-based assessment incorporating transferable skills useful for higher education and employment, as well as explicitly using education as a lever to tackle socio-economic deprivation. Much of this plan was widely lauded by key stakeholders, being learner-focused and practitioner-led, placing an emphasis on skills development and ensuring that it is appropriate for the specific needs of Wales. However, since its implementation, it has been criticised for various reasons and by various stakeholders, in many cases due to the inconsistent approach to its implementation in schools. The Welsh Government's Minister for Education and Skills appointed in June 2010, in looking for the reasons behind Wales' failing education system, found cause to commission no fewer than 24 reviews before his resignation in February 2013—almost one per month, with a range of issues related to the teaching of information communications technology (ICT).

### 3.5 Ireland

From its outset, the academic study of computing in Ireland was strongly linked with the Irish software industry and the government's involvement in its growth [118]. Despite a population of less than 5 million, Ireland is home to the EMEA headquarters of many multinational tech firms, and 16 of the 20 top global technology companies have strategic operations in Ireland including Apple, Facebook, Google Microsoft and Twitter. Dell/EMC, Ericsson, HPE, IBM, Intel, and Oracle have all been present in Ireland since before 1990 and still maintain a significant presence there [14]. By 1988 Ireland was the second largest exporter of software in the world and the value of software exports exceeded that of agricultural exports [55]. Since then Ireland has remained the first or second largest exporter of software to present day.

A small population combined with a tightly integrated educational landscape (state-run second-level examinations, a centralised third-level admissions system, and the fact that all third-level computing department heads meet regularly—see Sect. 3.5.3) and a globally competitive technology industry creates a fertile environment for computing education and research. The presence of companies like Intel have directly influenced the computer science curricula of Ireland's third-level institutions [142]. Companies such as Ericsson have also had similar influence [14]. When asked what influenced their choice of programming language of instruction for introductory programming in a 2019 survey of introductory programming instructors representing 90% of all publicly-funded and 80% of privately funded institutions, 81% reported "relevant to industry" as the top reason out of 15 choices [14]. Ireland has also had a unique impact on global computing education. For example, CoderDojo was founded in Ireland in 2011 and is headquartered in Dublin [14].

CER activity in Ireland has seen unprecedented growth in the last 5 years. Buoyed by an exceptionally strong tech sector and spurred on by the launch of a national computing curriculum at upper second-level in 2018. These years have seen the birth of at least three new research groups, a sharp increase in publications, and hosting the ACM ITiCSE and UKICER conferences. In 2019 SIGCSEire, the Ireland ACM SIGCSE Chapter[28] was established and is now the second-largest SIGCSE chapter with 219 members spanning primary, secondary and higher education, students, industry and government representatives, as well as grass-roots educators such as CoderDojo mentors. As of mid-2022, Ireland has 137 publications at SIGCSE conferences going back to 1986 with every university represented. 72% of these have been published since 2017. Those with 10 or more contributions in the last 5 years include University College Dublin (67) TU Dublin (19), Maynooth University (15), Trinity College Dublin (14), Dublin City University (11) and NUI

[28] SIGCSEire.acm.org.

Galway (10). A detailed scientometric analysis of research publications from the UK and Ireland follows in Sect. 4.

### 3.5.1 Primary Computing Education in Ireland

There is no formal primary school computing curriculum, however research has investigated the inclusion of computing at primary level[29] as part of the national primary curriculum review.[30] In July 2016, the National Council for Curriculum and Assessment (NCCA) was directed to investigate approaches towards integrating "coding" and "computational thinking" into the primary curriculum.

In 2016 a review of primary-level efforts in 22 jurisdictions was conducted.[31] This report laid the foundations for a deeper investigation in 2018 when efforts in six locales (England, New Zealand, Finland, the US—Washington state with CSTA, Northern Ireland and Scotland) were investigated in detail.[32] This work reported commonality in terms of what is taught, insights on the need for cross-curricular implementations, and that continuing professional development of teachers is a priority in all six countries in the study. This research was preceded by a report from Millwood et al. [161] reviewing the literature on computational thinking. It concluded that computational thinking was an appropriate focus in primary education and should be implemented as a cross-curricular component of the wider curriculum. The authors also provided the caveat that "Unplugged approaches are useful, but must be clearly linked with progression to plugged activities".

Most often curricula are developed before they are implemented. However, the NCCA have investigated possible elements of a programming curriculum in parallel to the research previously mentioned. Phase 1 started in 2017 and involved working with schools.[33] Schools were selected based on the teachers having prior experience. The goals were to capture the current state and capabilities of informal computing at primary level, and to plan and share examples for phase 2. A range of school types and location were included including disadvantaged and rural schools. Phase two began in fall 2018,[34] with novice teachers with little to no prior coding experience. This was to inform future possible curricular developments and to gain understanding of the potential benefits of teaching coding, computational thinking, and physical computing through project-based pedagogical approaches. This was in

[29] ncca.ie/en/primary/primary-developments/coding-in-primary-schools/research.

[30] ncca.ie/en/resources/primary-coding_final-report-on-the-coding-in-primary-schools-initiative.

[31] ncca.ie/en/resources/primary-coding_desktop-audit-of-coding-in-the-primary-curriculum-of-22-jurisdictions.

[32] ncca.ie/en/resources/primary-coding_investigation-of-curriculum-policy-on-coding-in-six-jurisdictions.

[33] ncca.ie/en/primary/primary-developments/coding-in-primary-schools/work-with-schools-phase-1.

[34] ncca.ie/en/primary/primary-developments/coding-in-primary-schools/work-with-schools-phase-2.

essence a trial run for coding as the majority of teachers, if rolled out nationally, will not have taught coding before.

The capstone to the NCCA's research and working with schools phase one and two, prior to the national curriculum review, was the final report on the Coding in Primary Schools Initiative.[35] This brought together the foundational research, the phase 1 and 2 trials, and an additional research investigation involving collecting data from teachers, management, parents and students. The report concluded with the identification of three aspects of digital competence—creating with technology, understanding technology, and using technology—as fundamental to the inclusion of coding and computational thinking into the curriculum.

Following this, and with ongoing discussions and review, the NCCA have three current strands of commissioned research: creating with technology, understanding technology, and using technology. These are forming core components for the consideration of "digital technology" (the name suggested in the conclusion of the final report) in the current national primary curriculum review. The possible outcomes of this body of work are that digital technology: (1) will be a standalone subject in the new primary national curriculum, (2) will be integrated as a cross-curricular component in the new primary national curriculum, or (3) will not be considered in the new national primary curriculum.

**Research**

Several university groups work directly with primary schools and teachers. As all of these also work with second-level schools and teachers, they are included in Sect. 3.5.2 below.

Research themes at primary level have included: curriculum [207]; capacity, access & participation [132]; computational thinking primary school resources [141]; computing education policy [58, 162]; informal learning [3, 4]; and parental involvement in primary school computing education [36, 38, 39].

### 3.5.2 Second-Level Computing Education in Ireland

Ireland's first association for computing in primary and secondary education—with many third level members—was CESI (Computers in Education Society of Ireland).[36] CESI was established in 1973 and is the official (department of education associated) professional network for K-12 computing teachers.

[35] ncca.ie/en/resources/primary-coding_final-report-on-the-coding-in-primary-schools-initiative/.

[36] www.cesi.ie.

**The Early Years**

The early stages of computing education in Ireland were documented by McCarr in 2009 [151]. From 1975 CESI introduced early research in teacher professional development [151] and computing experiments in schools [167]. While there was a growth spurt in initiatives and activities such as teacher diploma courses in computing or a department of education white paper (all discussed in the McCarr report), a very noteworthy series of events occurred, which formed the first steps towards formal computing education at K-12 in Ireland. Perhaps most significant early success was in 1981 which saw the inclusion of an optional computing component in the upper second level mathematics subject. While this was a positive first step, there were several limitations including the computing component being optional, not formally examined by the State Examinations Commission (SEC), and a lack of consistent learning outcomes. Following this, the department of education developed a lower second level subject in 1984. However just like the computing component in mathematics at higher level, it was not formally assessed. Over time both of these initiatives were discontinued [151].

Similar to many other jurisdictions, the 1990s saw a focus on more generic ICT related skills. In Ireland, computing took a back seat to ICT during this time-frame for a multitude of reasons which are described in the McCarr report [151]. In 2000 the department of education published the IT2000 report[37] followed by the Blueprint for ICT in Education report [121]. These reports reflect the pressures and mindsets at the time in terms of the need for ICT skills. McCarr concluded that the rationale for a focus on ICT was based on economic factors as well as findings from the OECD, which showed that Ireland at the time was performing below average. The last serious move in the ICT direction was a 2007 NCCA report "ICT Framework—A structured approach to ICT in Curriculum and Assessment: Revised Framework".[38]

**Recent Years**

The 2010s saw a significant shift again, this time migrating towards the inclusion of computing curricula at lower and upper second level. The initial offering from the NCCA came in the form of a Junior Cycle Short Course in Coding, introduced in 2016.[39] A Junior Cycle short course is a 100-h course that can be delivered at varying stages across the 3 years of the Junior Cycle (approximate ages 12–15). They are classroom-based and assessed, with an emphasis on active learning. The short courses were not intended to replace existing subjects, but to allow schools to broaden the range of learning experiences for students, and to access areas of

[37] www.gov.ie/en/publication/eae94c-schools-it2000.

[38] ncca.ie/en/resources/ict_framework_a_structured_approach_to_ict_in_curriculum_and_assessment_-_revised_framework.

[39] www.curriculumonline.ie/getmedia/cc254b82-1114-496e-bc4a-11f5b14a557f/NCCA-JC-Short-Course-Coding.pdf.

learning not covered by the combination of curricular subjects available in the school. LERO (the Science Foundation Ireland Research Centre for Software[40]) was commissioned to write the short course. The process consisted of the specification (2014–2016), a pilot project (2016–2017) in collaboration with the JCT (Junior Cycle for Teachers support service) team and Intel.[41] This short course has three strands. The first is "computer science introduction". This has grounded links for computer science comprehension and the understanding of a notional machine. The second is titled "lets get connected". This strand develops communication and architecture comprehension with a related learning outcome to build a website using HMTL and CSS. The final strand is "coding at the next level".

The 2017 Digital Strategy for Schools report[42] was the first serious indication at what would become the computer science subject at upper second level—as a state assessed subject—putting it on equal footing to subjects like Biology, Geography or Physics. At this time, the then Minister for Education fast-tracked the development of the Leaving Certificate Computer Science (LCCS) subject.[43] After development of the curriculum (in Ireland called a specification), a staged roll-out began in 2018 with 40 schools.[44] A textbook for the curriculum was published in 2020 by Becker and Quille [25] and the subject is now being taught in over 150 of 722 schools. In 2020 a framework document was developed by the department of education to support the growth and uptake of the subject.[45]

The assessment of the LCCS consists of a 70% terminal examination (with discussion that it would be online - not yet realised) and a 30% mark for a practical project called an Applied Learning Task (ALT) based on one or more of the ALTs detailed in the course specification. The NCCA specifies that the subject be taught in Python and/or JavaScript. The main rationale for this was that a multitude of programming languages would be difficult to regulate or get assessors to grade (assessments are graded centrally by the SEC). Similar to the Junior Cycle Short Course, the Leaving Certificate course consists of three strands: "practices and principles", "core concepts" and "computer science in practice". The latter contains four applied learning tasks (ALTs), which compliment the first and second strands [192].

**Research**

Research themes at second-level have included: artificial intelligence & machine learning education [147]; computing education policy [37, 58]; capacity, access &

[40] lero.ie.

[41] lero.ie/epe/schools.

[42] assets.gov.ie/24382/7b035ddc424946fd87858275e1f9c50e.pdf.

[43] ncca.ie/en/senior-cycle/curriculum-developments/computer-science.

[44] www.gov.ie/en/press-release/1238e6-minister-bruton-announces-leaving-certificate-computer-science-subje/.

[45] www.gov.ie/en/publication/5986e-leaving-certificate-computer-science-framework-2020.

participation [132, 191]; computational thinking [116, 131] for teachers [163, 175]; confidence [90, 156]; curriculum [207], K-12 outreach [105, 157, 171, 205]; developing a nationwide MOOC for second-level students [173]; initial teacher education & professional development [47, 48, 88, 97, 98, 153, 174]; principals' & guidance counsellors' attitudes towards computer science in schools [152]; self-esteem, [223]; teacher & learner agency [197]; and teacher programming self-efficacy [89, 155]. It is unsurprising given that the Irish second-level Computer Science curricula was implemented in 2018, that there is a large body of research in K-12 curricula from Irish authors [57, 91–93, 192, 215].

### 3.5.3 Higher Education in Ireland

**Enter Industry and the Birth of Computing Courses**

IBM opened an office in Ireland in 1956 and the first academic computer—an IBM 1620—was installed at University College Dublin in March 1962 and UCD's computing strategy was initiated and led by its science faculty. This was quickly followed by the installation of another IBM 1620 in June 1962 at Trinity College Dublin in the engineering school [118]. In 1964 University College Cork installed an IBM 1620 model 2 in its electrical engineering building, and in 1967 University College Galway installed an IBM 1800 data acquisition and control system.[46]

Professor John Byrne, the founder and long-time head of Trinity's Department of Computer Science, had such an impact on Irish computing that he is known by many as the "Father of Computing in Ireland" [118]. He identified that a major limiting factor for the emerging technology sector globally was the lack of an appropriately skilled workforce. He was responsible for developing educational programmes and creating a foundation of skilled professionals that contributed to attracting computing businesses to Ireland [71, 118].

In 1963 an M.Sc course in Computer Applications began at Trinity and in 1969 a Computer Science Department was established there.[47] In 1970 UCD began a BSc in Computer Science [5]. It is unclear when their Computer Science Department was formally established, but it was in place by 1972 [196]. UCD graduated its first Computer Science BSc cohort in 1972, the same year the first Computer Science PhD graduated (supervised by the Mathematics department) [221].

**Organisation: Industry, Government and Education**

In 1967 the Irish Computer Society (ICS) was founded as the national body for Information and Communication Technology (ICT) professionals in Ireland. Since its foundation the ICS has promoted the development of professional ICT

[46] techarchives.irish/irelands-first-computers-1956-69.

[47] www.scss.tcd.ie/SCSSTreasuresCatalog/literature/TCD-SCSS-DeptHistoryFor50thBirthday-v0.30-27-A5.pdf.

knowledge and skills in Ireland. The ICS is a member of the Council of European Professional Informatics Societies (CEPIS) which maintains formal and informal links with the European Union and is recognised as a non-governmental organisation with consultative status by the Council of Europe.[48] In 1976 an Advisory Group for Computer Services was established by the Higher Education Authority. In addition to the Authority itself, representatives from higher education, institutions, government departments, and private sector companies would serve as members of the group [220].

In 1981 The Irish Science and Technology Agency (EOLAS)—a state-sponsored body focused on the IT area set up in 1977—produced a report on the Irish computing industry entitled Microelectronics: The Implications for Ireland. Included in its recommended policies for the sustainability of the IT industry was funding at tertiary level of computer-related education and the extension of information technology appreciation into all secondary schools [168]. In 1998 proposals were put to government covering actions including increased educational capacity at third level [111]. In 1991 The National Software Directorate (NSD) was set up to align industry with education, creating niches in the software market and value from research in the area of software technology [113] with a 1992 budget of 1.4 m [111]. Amongst its aims were helping coordinate educational activities, and promoting software as a career, particularly to second level students [55].

Interestingly, in 1998 Condon reported that there was a large gender disparity in Computer Science with students identifying as women significantly underrepresented and that one of the goals of the NSD was to make computing just as attractive for women as men [55]. Over 30 years later women are still very underrepresented in Irish computing. A 2017 survey of several hundred Irish introductory programming students found that only 24% of students identified as female, and noted that the Higher Education Authority reported only 15% of students in computing degrees identified as female [204]. Another 2017 study of over 600 students at ten Irish (and one Danish) institutions found that students identifying as female reported significantly lower programming self-efficacy [190]. A 2019 Higher Education Authority survey reported that only 19% of undergraduate and 24% of postgraduate Information and Communication Technologies (Computer Science is not analysed separately) students identify as female [86].

As director of the NSD, Condon identified that matching educational activities and industry needs in an area which had developed as rapidly as software was difficult. As a result, with the help of Enterprise Ireland, in 1992[49] a forum involving the heads of all third-level departments of computing, industry representatives, and the NSD itself was established. In 1998 this forum met about four times a year and had already emerged as a useful means for identifying common issues and concerns, and in providing a valuable input into policy formation [55]. This

[48] www.ics.ie/news/view/114.

[49] E-mail correspondence with Ted Parslow, Chairperson of the Third Level Computing Forum (2007-present).

forum still exists today as the Third Level Computing Forum,[50] and still meets four times per year, now with the Department of Education and Skills, the National Council for Curriculum and Assessment, and Enterprise Ireland all having regular representation. A key to the success of the Forum has been the participation of industry, government departments and semi-state and professional bodies in addition to all Irish higher education institutions and second level and guidance counsellor representation [179].

**Recent Years**

In recent years, increased focus has been placed on apprenticeships, upskilling, and teacher training (specifically for the second-level Leaving Certificate Computer Science curriculum). Currently there are dozens of university provided programmes at the Diploma and MSc levels available to those who wish to change careers or improve progression potential. Several of these are specifically designed for in-service teachers who would like to teach computer science at school level. There are also several Bachelor's degree programmes aimed at pre-service computer science teachers including the Bachelor of Arts Education (Computer Science and Mathematical Studies) at NUI Galway,[51] the Bachelor of Science (Education) in Mathematics and Computer Science at the University of Limerick,[52] and the BSc in Computer Science, Mathematics & Education at University College Dublin.[53] There are also postgraduate qualifications for in-service teachers who want to teach Computer Science such as the Masters in Computer Science for Teachers at the Technological University of the Shannon: Midlands Midwest[54] and the Masters in Computer Science Education Research at Atlantic Technological University.[55]

A major focus in computing education in Ireland in recent years has been on diversity, equality and inclusion. Four computing departments have received Athena Swan Bronze departmental awards: IT Carlow, Trinity College Dublin, University College Dublin, and the University of Limerick. In 2017 the Irish Network for Gender Equality In Computing: INGENIC was created to "to unite, coordinate, and boost efforts in addressing gender equality in computing across all third-level institutions in Ireland".[56] The network has representatives from the computing departments of every higher education institution in Ireland.

[50] thirdlevelcomputingforum.ie.

[51] www.nuigalway.ie/courses/undergraduate-courses/education-computer-science-mathematical.html.

[52] www.ul.ie/courses/bachelor-science-education-mathematics-and-computer-science.

[53] www.myucd.ie/courses/science/computer-science-mathematics-education.

[54] lit.ie/en-ie/courses/master-of-science-in-computer-science-for-teachers.

[55] lyit.inventise.ie/CourseDetails/D303/LY_KEDRS_M/ComputerScienceEducationResearch.

[56] ingenic.ie.

**Working With Schools**

Several universities lead efforts to work with schools in Ireland. The PACT (Programming + Algorithms ≈ Computational Thinking) group[57] based in Maynooth University develops resources and supports to allow teachers to teach topics in computer science at both primary and secondary school. PACT also coordinates school visits and workshops. Since 2012 PACT has engaged over 30,000 teachers and students with funding from Maynooth University Department of Computer Science, the Google CS4HS programme, and Science Foundation Ireland (SFI). $CS_{INC}$ (Computer Science Inclusive)[58] based in TU Dublin organise student camps and workshops, teacher professional development as well as research in K-12 computing education. $CS_{INC}$ has engaged tens of thousands of students and thousands of teachers in recent years. They also run $CS_{LINC}$, an online student learning environment consisting of several modules built upon international best practices which are tailored to Irish second-level students. $CS_{LINC}$ has engaged over 10,000 students in the last 2 years. Bridge 21,[59] based at Trinity College Dublin, organises teacher professional development as well as working directly with schools and students.

**Funding**

There is no specialised source of funding for CER in Ireland. Funding can be sought from several disparate sources, however this typically requires that the direction of the research must be shaped to align with the aims of the funding body or particular call. Sources include The National Forum for the Enhancement of Teaching and Learning in Higher Education,[60] and Science Foundation Ireland (e.g. "Discover" calls). Additionally, regional and international (e.g. European Union) funding is available for specialised calls, in addition to globally-scoped special funds (e.g. SIGCSE Special Projects Grants).

**Research**

There are several large computing education research groups in Ireland including NUI Galway, the Computer Science Education Research (CSER) Group at Maynooth University,[61] $CS_{INC}$[62] at Technological University Dublin, and the University College Dublin Computing Education Research Group (CERG@UCD).[63]

[57] pact.cs.nuim.ie.

[58] csinc.ie.

[59] b21.scss.tcd.ie.

[60] www.teachingandlearning.ie.

[61] www.cs.nuim.ie/research/cser.

[62] csinc.ie.

[63] cerg.ucd.ie.

LERO, the Science Foundation Ireland (SFI) Research Centre for Software[64] also conducts computing education research. Combined these groups have produced approximately a dozen PhDs and have over two dozen current PhD students focused on computing education.

Research themes in higher education have included: achievement goals / mindset [188, 228]; AI-generated code [96]; artificial intelligence in education [13]; assessment [35, 170, 194]; classroom terminology [12, 15, 19]; computing education theory [65, 206]; diversity, equality and inclusion [164, 172, 190]; frame-based editing [41, 80]; group projects [27]; identifying at-risk students [22]; introductory programming / CS1 [18, 24, 115, 144, 158] in Ireland [14]; metacognition [75, 143, 183–185]; non-native English speakers [1]; notional machines [78, 79]; novice programmer behaviour [127, 128]; predicting programming success [10, 28, 50, 51, 186, 187, 189, 193]; prior programming experience [204]; program comprehension [45]; programming anxiety [56, 61]; programming error messages [16, 17, 20, 21, 23, 52, 74, 76, 126]; retention [59]; sense of belonging [165, 166]; soft-skills and creativity [107, 108]; student anxiety [170]; and teaching programming to adults [60].

## 4 Scientometrics of CER in the UK and Ireland

Having surveyed the factors that have influenced the development of CER in the UK and Ireland, we now review the outputs in a scientometric analysis. The analysis is based on a data-set that was retrieved from SCOPUS through a search based on keywords and publication venue. The retrieved data were manually checked for relevance, cleaned, checked and verified as described in detail in an earlier chapter [145]. In addition to the venues already identified, we add Computing Education Practice (CEP) and the UK and Ireland Computing Education Research (UKICER) conferences. Not all relevant papers are captured through this search. In particular the International Journal of Computer Science Education in Schools (IJCES) https://www.ijcses.org/ is not indexed in SCOPUS, although it does appear in Google Scholar, ERIC and Crossref. For this chapter, only articles with an author with a UK or Irish affiliation at the time of article publication were included regardless of the author order. The total number of articles was 1301. The author, institutions, and country networks were constructed using the fractional counting methods. The structural topic model analysis was based on the topics created using the methods in another chapter [7].

[64] lero.ie.

### 4.1 Data Cleaning

Detailed manual cleaning of the identified set of papers was carried out for the base data-set, and then further work was done to identify current research institutions (mainly Higher Education Institutions) in the UK and Ireland corresponding to the institutional affiliations listed in the original papers. Different types of changes were made to institution titles

1. Grouping of different names of the same institution e.g. UNIVERSITY OF KENT, UNIVERSITY OF KENT AT CANTERBURY, UNIV OF KENT AT CANTERBURY, UNIV. OF KENT, UNIVERSITY OF KENT CANTERBURY
2. Renaming of institutions to their current title, for example SHEFFIELD POLYTECHNIC became SHEFFIELD HALLAM UNIVERSITY as part of the 1992 founding of "new universities" in the UK from former polytechnics and central institutions
3. Renaming following merger of institutions e.g. PAISLEY COLLEGE OF TECHNOLOGY became part of the UNIVERSITY OF THE WEST OF SCOTLAND
4. De-merging of institutions that were listed with the same title e.g. ABERYSTWYTH from UNIVERSITY OF WALES, MAYNOOTH from NATIONAL UNIVERSITY OF IRELAND
5. Shortening to familiar abbreviations e.g. QUEEN'S UNIVERSITY BELFAST to QUB
6. Identification of institution titles where sub-institutional titles had been extracted e.g. SCHOOL OF COMPUTING

This was not a straightforward process as no one author had enough knowledge of the different institutions. Most of the identification was done through automatically extracted institutional affiliations, but some required going back to the original papers e.g. for de-mergers.

### 4.2 Number of Publications and Citations

The most basic counts that can be made of published research are the number of publications and the number of citations. Figure 1 shows the historical trend of the total number of CER papers published by authors affiliated with institutions from the UK and Ireland. In many cases there is collaboration between institutions and countries, so these are apportioned according to the number of authors. Whilst it would be possible to further break down the paper count to differentiate between Wales, Scotland, Northern Ireland and England, presenting the sometimes small volumes and multiple combinations of collaboration makes this too difficult to present in this way. Figure 2 shows only the trend in papers including authors at institutions from the UK and Ireland.

There is a noticeable surge in publications in the mid-1990s, shortly after the conversion of former UK polytechnics to universities, and in Ireland the foundation of Institutes of Technology (IoT), both in 1992. It is possible that staff at these institutions, which had a mainly teaching remit, looked to generate research outputs based on their teaching. There is no comparable increase in publications internationally in the mid-1990s so the proportional contribution of the UK and Ireland to all CER publications increased dramatically during this period.

This late-90s surge then drops away, before another surge in 2005, possibly in response to the desire to address curriculum and pedagogy in the light of falling numbers of computing students in universities. Finally there is an increase from 2015 onward, possibly driven by curriculum reform in schools [43] and the Shadbolt report on employability [201]. Some of the peaks in proportional paper output coincide with ITiCSE conferences being held in Ireland and the UK (1998, 2004, 2019) but other ITiCSE conferences held here (2001, 2013) do not seem to have the same effect, so it seems likely that these other influences have an important effect. The current peak in outputs and proportion of all outputs is reflected in, and partly driven by, the establishment of the ACM SIGCSE chapters in Ireland and the UK, and outputs from Computing Education Practice (CEP) and UK and Ireland

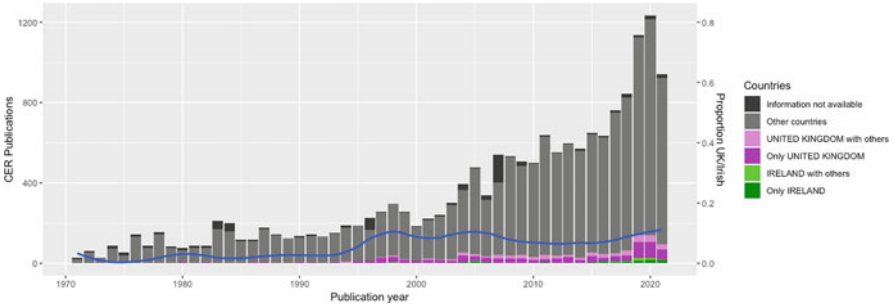


**Fig. 1** Total articles by year. Blue lines show proportion (smoothed) of all CER publications that included authors from the UK and Ireland

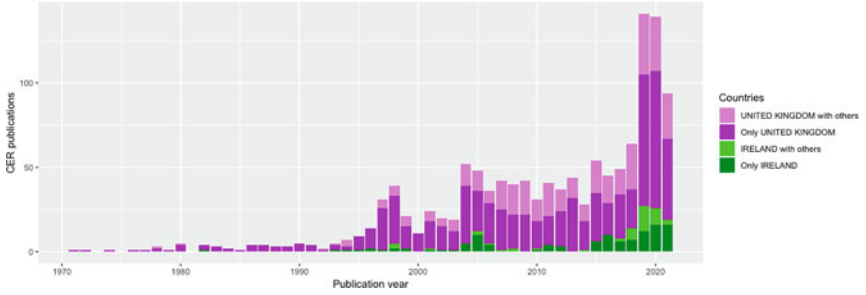


**Fig. 2** Total articles by year from UK and Ireland

Computing Education Research (UKICER) conferences. Both of these conferences started publishing with ACM ICPS—and hence are included in this scientometric analysis—since 2019.

### *4.3 Most Frequently Cited Papers*

Table 1 lists the papers that include authors from the UK and Ireland that have had the most citations per year (CPY) since publication. These are dominated by papers about teaching of programming, but with important contributions around school curriculum, particularly following on from the development of national curricula in schools from 2014.

### *4.4 Collaboration Networks*

It is relatively straightforward to identify the names of collaborating authors, as they are listed directly in the search results. These data can then be used to build a network, where the edges represent paper collaborations, apportioned according to the number of authors. Representing this network graphically, where frequently collaborating authors are placed near to each other, is shown in Fig. 3. The size of the nodes corresponds with the amount of collaboration—which is not a very reliable measure because it is assumed that for any given paper all listed authors make the same contribution to it.

Following the cleaning of institutional affiliation data, a collaboration network can similarly be constructed for institutions, see Fig. 4. It is noticeable that a large proportion of the institutions listed are not in the UK or Ireland. This may be because the community is outward looking and keen to engage with educators and students in other contexts—or that researchers are often isolated and hence more likely to find collaborators at international conferences than within their own or neighbouring institutions. Table 2 shows how international conferences dominate the venues of CER publications from the UK and Ireland. Figure 5 shows the co-publication relationships between different countries explicitly. It is noticeable that authors from the UK are much more likely to collaborate with colleagues in the USA than colleagues in Ireland, and vice versa, although this could be explained simply by the number of researchers and outputs from the USA. In general it is surprising how little the level of collaboration depends on geographical proximity, or even a common language.

**Table 1** Most-cited articles with total number of citations (C) ordered by citations per year (CPY) with CPY > 10. List of authors includes first author and any authors from the UK and Ireland

| Title | Year | Authors | C | CPY |
|---|---|---|---|---|
| Failure rates in introductory programming revisited | 2014 | Watson & Li | 283 | 35.4 |
| Introductory programming: a systematic literature review | 2018 | Luxton-Reilly, Simon, Becker, et al. | 125 | 31.3 |
| A multi-national, multi-institutional study of assessment of programming skills of first-year CS students | 2001 | McCracken, Utting, et al. | 453 | 21.6 |
| Computing in the curriculum: challenges and strategies from a teacher's perspective | 2017 | Sentance & Csizmadia | 107 | 21.4 |
| A survey of literature on the teaching of introductory programming | 2007 | Pears, Devlin, Paterson, et al. | 317 | 21.1 |
| Restart: the resurgence of computer science in UK schools | 2014 | Brown, Sentance, Crick & Humphreys | 142 | 17.8 |
| 37 Million compilations: investigating novice programming mistakes in large-scale student data | 2015 | Altadmri & Brown | 119 | 17.0 |
| Computer science in K-12 school curricula of the twenty-first century: why, what and when? | 2017 | Webb, et al. | 81 | 16.2 |
| Automatic test-based assessment of programming: a review | 2005 | Douce, Livingstone & Orwell | 250 | 14.7 |
| Educating the internet-of-things generation | 2013 | Kortuem, Bandara, Smith, Richards & Petre | 117 | 13.0 |
| The impact of covid-19 and "emergency remote teaching" on the UK computer science education community | 2020 | Crick, Knight, Watermeyer & Goodall | 26 | 13.0 |
| A systematic review of approaches for teaching introductory programming and their influence on success | 2014 | Vihavainen & Watson | 103 | 12.9 |
| The greenfoot programming environment | 2010 | Kölling | 149 | 12.4 |
| Compiler error messages considered unhelpful: the landscape of text-based programming error message research | 2019 | Becker, et al. | 32 | 10.7 |
| No tests required: comparing traditional and dynamic predictors of programming success | 2014 | Watson, Li & Godwin | 84 | 10.5 |
| Teaching introductory programming: a quantitative evaluation of different approaches | 2014 | Koulouri, Lauria & Macredie | 84 | 10.5 |
| A multi-national study of reading and tracing skills in novice programmers | 2004 | Lister, Fone, Thomas, et al. | 185 | 10.3 |
| Developing assessments to determine mastery of programming fundamentals | 2018 | Luxton-Reilly, Becker, McDermott, et al. | 40 | 10.0 |
| 50 years of CS1 at SIGCSE: a review of the evolution of introductory programming education research | 2019 | Becker & Quille | 30 | 10.0 |
| Source-code similarity detection and detection tools used in academia: a systematic review | 2019 | Novak, Joy, et al. | 30 | 10.0 |

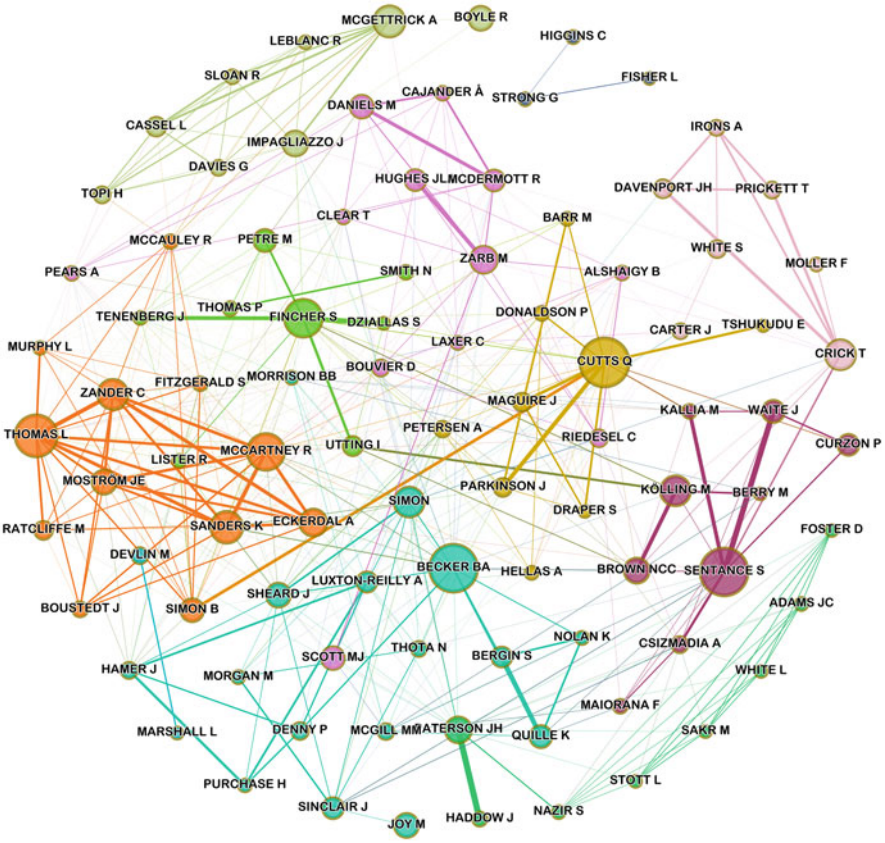


**Fig. 3** Author collaboration for outputs including institutions in the UK and Ireland. Colours denote clusters of collaborating authors

## *4.5 Topic Modelling*

Figure 6 shows how the subject content of published articles has changed over time. This analysis was performed by carrying out topic modelling on the titles and abstracts of papers, with apportionment of papers between topics where multiple topics were identified (see [7]). There are some interesting trends to note in terms of the total number of articles published, and we break these down into five main patterns: steady; emerging; receding; fluctuating; and missing.

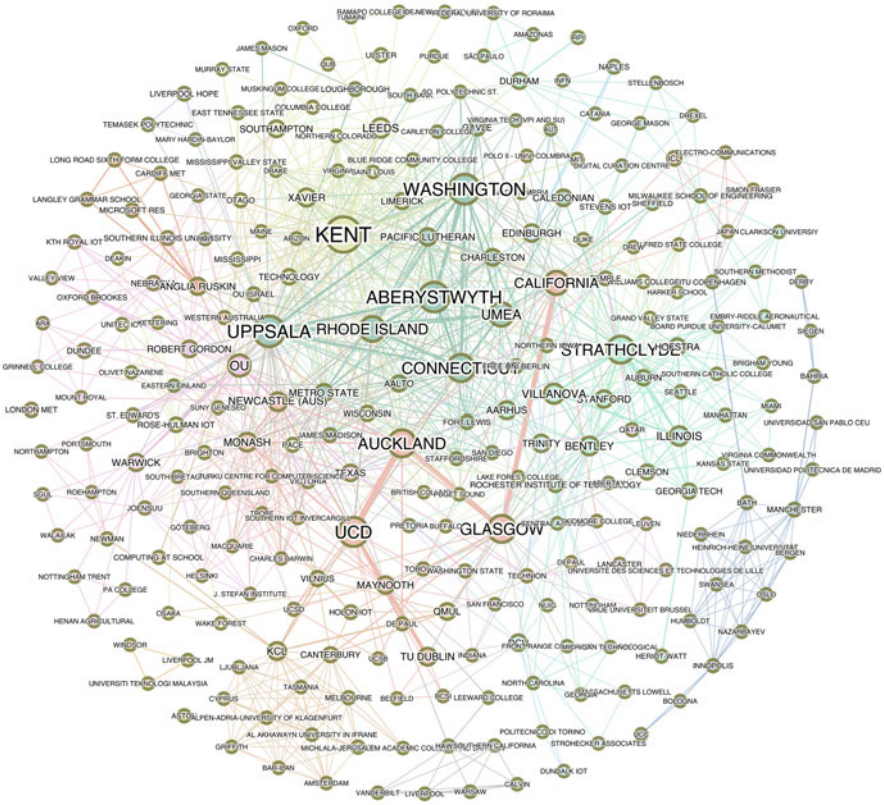


**Fig. 4** Institution collaboration for outputs including institutions in the UK and Ireland. Colours denote clusters of collaborating institutions

### 4.5.1 Steady

Some topics have sustained a fairly constant rate of publication, particularly when considering the overall increase in the number of papers. Programming, Assessment and Pedagogy fall into this category, although at different levels of activity. Programming has always had a high number of papers, and the top three papers in terms of citations per year are also in this area (see Table 1).

### 4.5.2 Emerging

Coverage of these topics has increased substantially over the period: Computational Theory; Computational Thinking; Data Mining; Educational Psychology; Gender and Diversity; Introductory courses; Projects; STEM. In some cases this is relatively unsurprising, for example Gender and Diversity have rightly had increasing public

**Table 2** Venues that have published 10 or more papers by authors from the UK and Ireland

| Venue | Publications |
|---|---|
| Innovation and Technology in Computer Science, ITiCSE | 437 |
| ACM Technical Symposium on Computer Science Education, SIGCSE | 140 |
| ACM SIGCSE Bulletin | 92 |
| International Conference on Educational Research, ICER | 70 |
| Conference on Computing Education Practice, CEP | 59 |
| Workshop in Primary and Secondary Computing Education, WiPSCE | 44 |
| Koli Calling International Conference on Computing Education Research | 41 |
| Computer Science Education | 36 |
| Frontiers in Education Conference, FIE | 33 |
| UK and Ireland Computing Education Research Conference, UKICER | 30 |
| ACM Transactions on Computing Education, TOCE | 29 |
| International Conference on Informatics in Schools, ISSEP | 13 |
| ACM Journal on Educational Resources in Computing | 10 |

focus in general, and specifically within HE through the Athena Swan scheme [216]. Computational Thinking is (arguably [208]) a new subject area and research into Data Mining education reflects the growing amount of data that is collected and used as part of our every day lives. Perhaps more surprising is the growth in the numbers of papers about Projects, given that project work has long been an important and sometimes difficult area of computing education.

### 4.5.3 Receding

After early interest in Design and OOP there have been relatively few recent articles on these topics, following peaks in 2004 and 2008 respectively. Education Technology within computing education research has also experienced a reduction in the total numbers of papers, and a marked diminution in the proportion of all CER articles written in the UK and Ireland.

### 4.5.4 Fluctuating

All of the topics show fluctuation, but there are some particularly noticeable variations. Software Engineering (SE) has fallen and then risen again, possibly reflecting a move away from traditional SE techniques and an emergence of interest in agile methodologies. Programming Languages have followed a similar pattern, perhaps following the trend for adoption of Java in the early part of the century (relating also to a growth in OOP), a period of stability and then a move towards python. Curriculum has also had a surge of interest coinciding with the introduction of national curricula in schools in the 2010s. It might be assumed that artificial

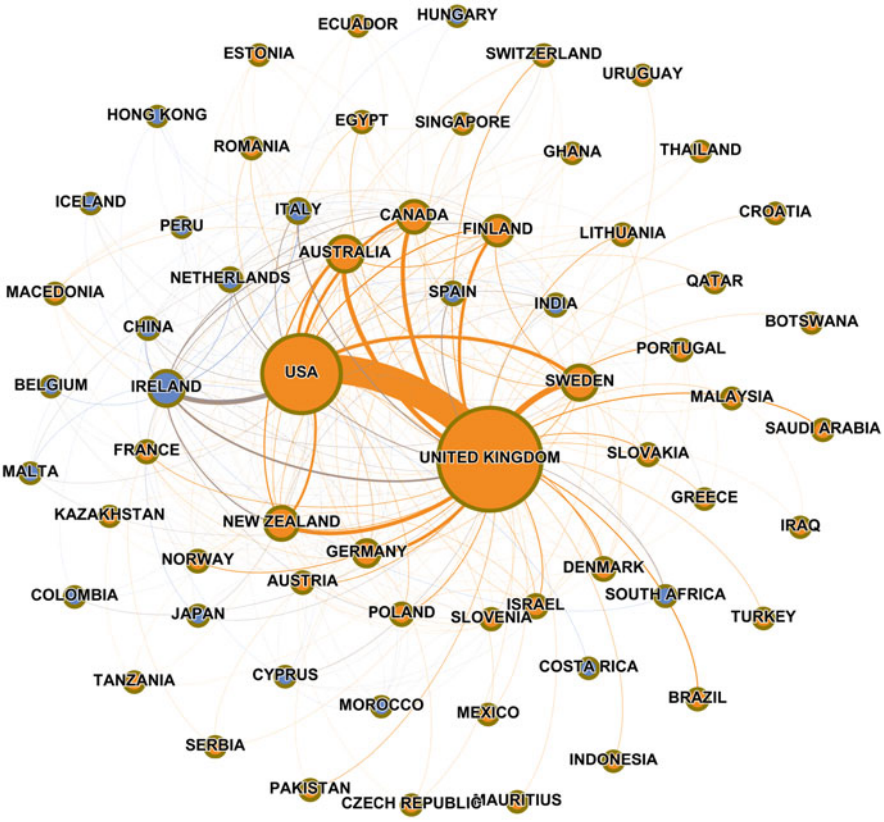


**Fig. 5** International collaboration for outputs including institutions in the UK and Ireland. Colours denote clusters of collaborating countries

intelligence and machine learning would be relatively recent ventures, in parallel with Data Mining, but these have really experienced only a slight recent renaissance, with the peak of interest (proportionally at least) in 2004/5.

### 4.5.5 Missing

Some topics—Databases and Networks—are surprising in their absence from the list, given how much curriculum time is given to them at school and university. Alongside small numbers for Computer Architecture and Operating Systems, it seems that computer systems topics in general are under-represented. We might similarly have expected to see more on web and internet systems/programming given how central they are to industrial practice, as well as computing syllabuses. Security, mobile, and cloud computing are also missing from this list of the most common topics, possibly because they are relatively new and systems-focused.

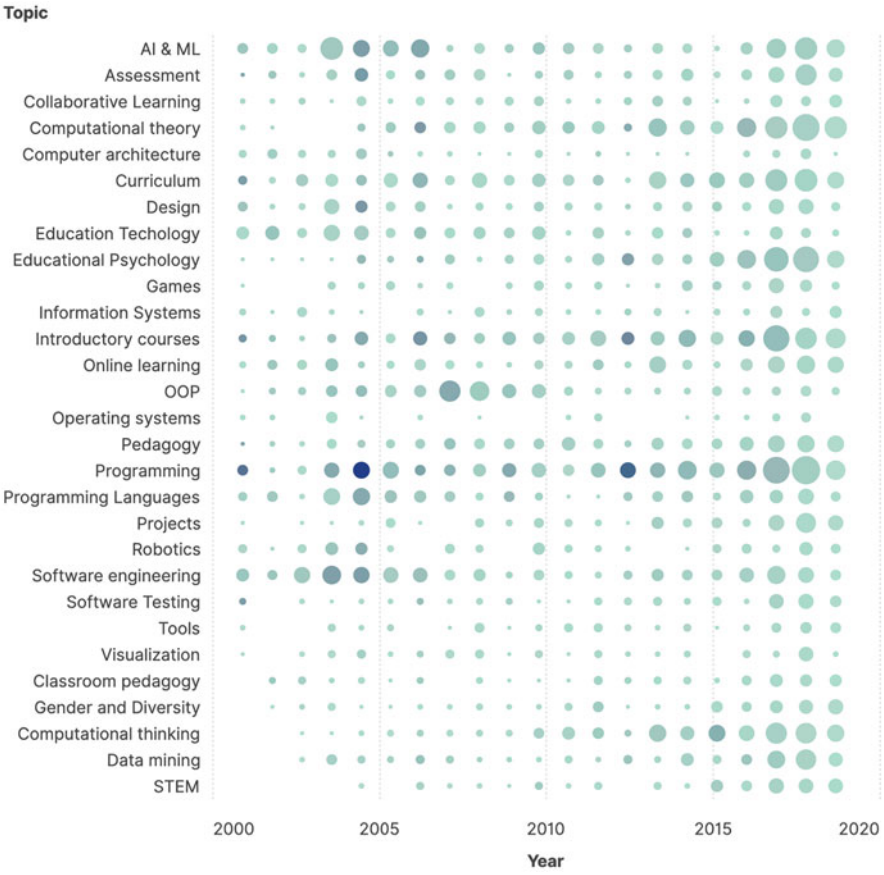


**Fig. 6** Topics by year. The size of the circle represents the number of papers published, the darkness represents the number of citations received

## 5 Discussion

CER in the UK and Ireland has a long history for such a young subject. The British Isles have made strong contributions to the global CER community, and maintain robust international collaborations. At the time of writing that contribution is at an all-time peak of activity based on the proportion of all CER papers published internationally. Our CER community has contributed strongly to the establishment of national curricula, as the UK and Ireland have been in the vanguard in terms of pre-university computing.

However, much of the CER in the British Isles has come from researchers whose interest is strong enough, and university situations and commitments flexible enough, to allow for the dedication of time and resources to effect meaningful work.

Government funding, when it has come, has usually targeted the delivery of training, such as the National Centre for Computing Education or the Institute of Coding. Some basic research has fallen out of this funding driven by the interest of the people working on it.

Nonetheless, CER is not well funded in the UK and Ireland, with no national government grant-awarding bodies having the area within their specified remit. To some extent this is due to issues of intersectionality: should it sit within education research funding or computing research funding? Should the quality of the research be assessed by an education panel or a computing panel? And should research into teaching children about computing be done in schools or universities? Much of this can be put down to the relative youth of the subject within university settings, where funding for more traditional subjects dominate. However it is notable that other countries do have national funding that is accessible to computing education researchers e.g. the NSF in the USA.

Because most tenured academics are required to carry out teaching as well as research activities, CER outputs are often rooted in the authors' own teaching practice or interests, rather than in funded projects. To some extent this gives a welcome freedom because the direction of research is not usually dictated by national policy initiatives, and researchers can choose to follow their own path. Most teachers in schools, however, don't have the time or training to engage in research, so outputs are more often focused in Higher Education, where scholarship and research is a more central expectation.

This chapter has documented the CER conducted in the UK and Ireland, demonstrating that there is a vibrant CER community that has conducted significant research at primary, second-level and higher education. This research spans the entire range of CER including classroom practice and pedagogy, student well-being, tools, outreach, theory, diversity, curriculum design, policy, and global engagement to name a few.

Looking forward, recent growth in computing student numbers has lead to an increase in university staff, which may lead to further engagement in CER. It would be interesting to explore the scientometric data to see the extent to which new people are joining the CER community. Emerging topics identified from the analysis are AI and machine learning, educational psychology, computational theory and diversity, equity & inclusion. Like most of the rest of the world, we continue to struggle with introductory programming, assessment (particularly plagiarism) and effective and realistic pedagogical approaches. Whether these topics will still be the main focus in 10 years time depends on developments in the subject and its pedagogy, but mainly on the interests of staff engaged in teaching—unless a source of regular government funding is made available to steer the direction of the community.

Like the rest of the global CER community, much work is outstanding. In the UK and Ireland, both of which have relatively well established second-level computing programmes, research into the transition from secondary school to higher education is needed. Additionally, how to teach computing to primary school children is an open question. Further, the current state of poor retention at university level, and poor uptake in secondary schools, combined with significant under-representation

of many groups in student bodies, all demand more research. These topics will require more research into sense of belonging, the student experience, and tackling persistent problems with misconceptions of computing in the general population, particularly pre-university students and their parents. Finally, the UK and Ireland, along with all other countries, are now at the beginning of a new era—one defined by artificial intelligence finally, and rather suddenly, delivering real changes—and concerns—that affect the very nature of not only the academic discipline of computing but the way that it is taught and learned. One of the biggest challenges is that AI is now advancing faster than institutional, governmental, and social processes can adapt. This demands significant research presently.

**Acknowledgments** Brett Becker and Keith Quille would like to thank Chris Bleakley, John Dunnion, Henry McLoughlin, Andrew Hines, Barry Smyth, and Catherine Mooney of the School of Computer Science at University College Dublin; Ted Parslow, Chairperson of the Third Level Computing Forum (2007-present); and Karen Nolan & Roisin Faherty of the Department of Computing at TU Dublin for their thoughtful assistance. Steven Bradley would like to thank Jacob Bradley for using his expertise in R to assist with data analysis and visualisation.

## References

1. Alaofi, S., Russell, S.: A validated computer terminology test for predicting non-native english-speaking CS1 students' academic performance. In: Australasian Computing Education Conference, ACE '22, p. 133–142. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3511861.3511876
2. Albury, R., Allen, D.: Microelectronics. (1979). URL: https://clp.bbcrewind.co.uk/media/BBC-Microelectronic-government-submission.pdf
3. Alsheaibi, A., Huggard, M., Strong, G.: Teaching within the CoderDojo movement: An exploration of mentors' teaching practices. In: 2020 IEEE Frontiers in Education Conference (FIE), pp. 1–5 (2020). https://doi.org/10.1109/FIE44824.2020.9273998
4. Alsheaibi, A., Strong, G., Millwood, R.: The need for a learning model in coderdojo mentoring practice. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education, WiPSCE '18. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3265757.3265785
5. Anderson, F.: UCD trains future computocrats. Irish Times p. 17–17 (1972)
6. Anderson, R.E.: National computer literacy, 1980. In: Computer Literacy, pp. 9–17. Elsevier (1982)
7. Apiola, M., López-Pernas, S., Saqr, M.: The Evolving Themes of Computing Education Research: Trends, Topic Models, and Emerging Research. In: Past, Present and Future of Computing Education Research. Springer, Rochester, NY (2023)
8. Astrachan, O., Cuny, J., Stephenson, C., Wilson, C.: The CS10K project: Mobilizing the community to transform high school computing. In: Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, pp. 85–86 (2011)
9. Augar, P.: Post-18 review of education and funding: Independent panel report (2019). URL www.gov.uk/government/publications/post-18-review-of-education-and-funding-independent-panel-report
10. Azcona, D., Casey, K.: Micro-analytics for student performance prediction. Int. J. Comput. Sci. Softw. Eng **4**(8), 218–223 (2015)
11. Barr, M.: Video games can develop graduate skills in higher education students: A randomised trial. Computers & Education **113**, 86–97 (2017)

12. Becker, B.: The roles and challenges of computing terminology in non-computing disciplines. In: United Kingdom and Ireland Computing Education Research Conference., UKICER '21. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3481282.3481284
13. Becker, B.A.: Artificial intelligence in education: What is it, where is it now, where is it going? In: B. Mooney (ed.) Ireland's Yearbook of Education 2017–2018, *30*, vol. 1, pp. 42–48. Education Matters, Dublin, Ireland (2017). ISBN: 978-0-9956987-1-0, educationmatters.ie/download-irelands-yearbookeducation/
14. Becker, B.A.: A survey of introductory programming courses in Ireland. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '19, p. 58–64. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3304221.3319752
15. Becker, B.A.: What does saying that 'programming is hard' really say, and about whom? Commun. ACM **64**(8), 27–29 (2021). https://doi.org/10.1145/3469115
16. Becker, B.A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D.J., Harrington, B., Kamil, A., Karkare, A., McDonald, C., Osera, P.M., Pearce, J.L., Prather, J.: Compiler error messages considered unhelpful: The landscape of text-based programming error message research. In: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE-WGR '19, p. 177–210. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3344429.3372508
17. Becker, B.A., Denny, P., Siegmund, J., Stefik, A.: The Human Factors Impact of Programming Error Messages (Dagstuhl Seminar 22052). Dagstuhl Reports **12**(1), 119–130 (2022). https://doi.org/10.4230/DagRep.12.1.119
18. Becker, B.A., Fitzpatrick, T.: What do CS1 syllabi reveal about our expectations of introductory programming students? In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, p. 1011–1017. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3287324.3287485
19. Becker, B.A., Gallagher, D., Denny, P., Prather, J., Gostomski, C., Norris, K., Powell, G.: From the horse's mouth: The words we use to teach diverse student groups across three continents. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2022, p. 71–77. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3478431.3499392
20. Becker, B.A., Goslin, K., Glanville, G.: The effects of enhanced compiler error messages on a syntax error debugging test. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18, p. 640–645. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3159450.3159461
21. Becker, B.A., Mooney, C.: Categorizing compiler error messages with principal component analysis. In: 12th China-Europe International Symposium on Software Engineering Education (CEISEE 2016), Shenyang, China, 28–29 May 2016 (2016)
22. Becker, B.A., Mooney, C., Kumar, A.N., Russell, S.: A simple, language-independent approach to identifying potentially at-risk introductory programming students. In: Australasian Computing Education Conference, ACE '21, p. 168–175. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3441636.3442318
23. Becker, B.A., Murray, C., Tao, T., Song, C., McCartney, R., Sanders, K.: Fix the first, ignore the rest: Dealing with multiple compiler error messages. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18, p. 634–639. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3159450.3159453
24. Becker, B.A., Quille, K.: 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, p. 338–344. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3287324.3287432
25. Becker, B.A., Quille, K.: Computer Science for Leaving Certificate. Golden Key Educational Publishing (2020). ISBN: 978-19998293-1-5, url: goldenkey.ie/computer-science-for-leaving-cert/
26. Becker, B.A., Settle, A., Luxton-Reilly, A., Morrison, B.B., Laxer, C.: Expanding opportunities: Assessing and addressing geographic diversity at the SIGCSE Technical Symposium.

In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, p. 281–287. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3408877.3432448
27. Bergin, S., Mooney, A.: An innovative approach to improve assessment of group based projects. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli Calling '16, p. 12–20. ACM, NY, NY, USA (2016). https://doi.org/10.1145/2999541.2999543
28. Bergin, S., Mooney, A., Ghent, J., Quille, K.: Using machine learning techniques to predict introductory programming performance. International Journal of Computer Science and Software Engineering (IJCSSE) **4**(12), 323–328 (2015). URL mural.maynoothuniversity.ie/8682/
29. Bikanga Ada, M., Foster, M.E.: Enhancing postgraduate students' technical skills: perceptions of modified team-based learning in a six-week multi-subject bootcamp-style cs course. Computer Science Education pp. 1–25 (2021)
30. Blackwell, A.F., Petre, M., Church, L.: Fifty years of the psychology of programming. International Journal of Human-Computer Studies **131**, 52–63 (2019)
31. Blyth, T.: The legacy of the BBC Micro: Effecting change in the UK's cultures of computing. London, UK: Nesta (2012)
32. Boole, G.: The mathematical analysis of logic. Philosophical Library (1847)
33. Boole, G.: An investigation of the laws of thought: On which are founded the mathematical theories of logic and probabilities. Dover (1854)
34. du Boulay, B., O'Shea, T., Monk, J.: The black box inside the glass box: Presenting computing concepts to novices. International Journal of man-machine studies **14**(3), 237–249 (1981)
35. Bouvier, D., Lovellette, E., Matta, J., Alshaigy, B., Becker, B.A., Craig, M., Jackova, J., McCartney, R., Sanders, K., Zarb, M.: Novice programmers and the problem description effect. In: Proceedings of the 2016 ITiCSE Working Group Reports, ITiCSE '16, p. 103–118. ACM, NY, NY, USA (2016). https://doi.org/10.1145/3024906.3024912
36. Bresnihan, N., Bray, A., Fisher, L., Strong, G., Millwood, R., Tangney, B.: Parental involvement in computer science education and computing attitudes and behaviours in the home: Model and scale development. ACM Trans. Comput. Educ. **21**(3) (2021). https://doi.org/10.1145/3440890
37. Bresnihan, N., Millwood, R., Oldham, E., Strong, G., Wilson, D.: A critique of the current trend to implement computing in schools. Pedagogika **65**(3), 292–300 (2015)
38. Bresnihan, N., Strong, G., Fisher, L., Millwood, R., Lynch, A.: OurKidsCode: A national programme to get families involved in CS education. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '19, p. 298. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3304221.3325574
39. Bresnihan, N., Strong, G., Fisher, L., Millwood, R., Lynch, Á.: Increasing parental involvement in computer science education through the design and development of family creative computing workshops. In: H.C. Lane, S. Zvacek, J. Uhomoibhi (eds.) Computer Supported Education, pp. 479–502. Springer International Publishing, Cham (2020)
40. Briggs, A., Snyder, L.: Computer Science Principles and the CS 10K initiative. ACM Inroads **3**(2), 29–31 (2012)
41. Brown, N., Kyfonidis, C., Weill-Tessier, P., Becker, B., Dillane, J., Kölling, M.: A frame of mind: Frame-based vs. text-based editing. In: United Kingdom and Ireland Computing Education Research Conference., UKICER '21. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3481282.3481286
42. Brown, N., Sentance, S., Crick, T., Humphreys, S.: Restart: The resurgence of computer science in UK Schools. ACM Trans. Comput. Educ. **14**(2) (2014)
43. Brown, N.C.C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., Sentance, S.: Bringing computer science back into schools: lessons from the UK. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education, pp. 269–274. ACM (2013). URL dl.acm.org/citation.cfm?id=2445277
44. Bruderer, H.: Computing history beyond the UK and US: Selected landmarks from continental Europe. Commun. ACM **60**(2), 76–84 (2017)

45. Buckley, J., Exton, C.: Bloom's taxonomy: A framework for assessing programmers' knowledge of software systems. In: 11th IEEE International Workshop on Program Comprehension, 2003., pp. 165–174 (2003). https://doi.org/10.1109/WPC.2003.1199200
46. Burns, J.: Coding on tape - computer science A-level 1970s style. BBC News (2016). URL www.bbc.com/news/education-35890450
47. Byrne, J.R., Fisher, L., Tangney, B.: Computer science teacher reactions towards Raspberry Pi continuing professional development (CPD) workshops using the Bridge21 model. In: 2015 10th International Conference on Computer Science & Education (ICCSE), pp. 267–272 (2015). https://doi.org/10.1109/ICCSE.2015.7250254
48. Byrne, J.R., Fisher, L., Tangney, B.: Empowering teachers to teach CS — Exploring a social constructivist approach for CS CPD, using the Bridge21 model. In: 2015 IEEE Frontiers in Education Conference (FIE), pp. 1–9 (2015). https://doi.org/10.1109/FIE.2015.7344030
49. Calder, D.P.M.: BCS landscape review: Computing qualifications in the UK (2021). URL www.bcs.org/media/8665/landscape-review-computing-report.pdf
50. Casey, K.: Using keystroke analytics to improve pass-fail classifiers. Journal of Learning Analytics **4**(2), 189–211 (2017). URL mural.maynoothuniversity.ie/10183/
51. Casey, K., Azcona, D.: Utilizing student activity patterns to predict performance. International Journal of Educational Technology in Higher Education **14**(1), 1–15 (2017)
52. Caton, S., Russell, S., Becker, B.A.: What fails once, fails again: Common repeated errors in introductory programming automated assessments. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2022, p. 955–961. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3478431.3499419
53. CCEA: CCEA curriculum review (2002). URL www.nicurriculum.org.uk/docs/background/curriculum_review/primsubt.pdf
54. Cheung, A., Paun, A., Valsamidis, L.: Devolution at 20. London: Institute for Government (2019)
55. Condon, J.: The Irish software industry and education. ACM SIGCSE Bull. **30**(3), 1–4 (1998). https://doi.org/10.1145/290320.282995
56. Connolly, C.: Addressing programming anxiety and underperformance among first year computing students through pedagogical innovation: An in-depth analysis. Ph.D. thesis, University of Limerick (2007)
57. Connolly, C.: Computer science at post primary in Ireland: Specification design and key skills integration. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education, WiPSCE '18. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3265757.3265760
58. Connolly, C., Byrne, J.R., Oldham, E.: The trajectory of computer science education policy in Ireland: A document analysis narrative. European Journal of Education **57**(3), 512–529 (2022). https://doi.org/doi.org/10.1111/ejed.12507
59. Connolly, C., Murphy, E.: Retention initiatives for ICT based courses. In: Proceedings Frontiers in Education 35th Annual Conference, pp. S2C–10 (2005). https://doi.org/10.1109/FIE.2005.1612215
60. Connolly, C., Murphy, E., Moore, S.: Second chance learners, supporting adults learning computer programming. In: International Conference on Engineering Education–ICEE (2007)
61. Connolly, C., Murphy, E., Moore, S.: Programming anxiety amongst computing students—a key in the retention debate? IEEE Transactions on Education **52**(1), 52–56 (2009). https://doi.org/10.1109/TE.2008.917193
62. Copeland, B.J.: Alan Turing's automatic computing engine: The master codebreaker's struggle to build the modern computer. OUP Oxford (2005)
63. Copeland, B.J.: Alan Turing's electronic brain: The struggle to build the ACE, the world's fastest computer. Oxford University Press (2012)
64. Crick, T., Sentance, S.: Computing at School: stimulating computing education in the UK. In: Proceedings of the 11th Koli Calling International Conference on Computing Education Research, pp. 122–123 (2011)

65. Cristaldi, G., Quille, K., Csizmadia, A.P., Riedesel, C., Richards, G.M., Maiorana, F.: The intervention, intersection and impact of social sciences theories upon computing education. In: 2022 IEEE Global Engineering Education Conference (EDUCON), pp. 1561–1570 (2022). https://doi.org/10.1109/EDUCON52537.2022.9766704
66. Croarken, M.: Mary Edwards: Computing for a living in 18th-century England. IEEE Annals of the History of Computing **25**(4), 9–15 (2003)
67. Croarken, M.: Tabulating the heavens: Computing the nautical almanac in 18th-century England. IEEE Annals of the History of Computing **25**(3), 48–61 (2003)
68. Croarken, M.: Human computers in eighteenth-and nineteenth-century Britain. The Oxford Handbook of the History of Mathematics p. 375 (2008)
69. Cutts, Q., Barr, M., Bikanga Ada, M., Donaldson, P., Draper, S., Parkinson, J., Singer, J., Sundin, L.: Experience report: Thinkathon–countering an "I got it working" mentality with pencil-and-paper exercises. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, pp. 203–209 (2019)
70. Cutts, Q., Robertson, J., Donaldson, P., O'Donnell, L.: An evaluation of a professional learning network for computer science teachers. Computer Science Education **27**(1), 30–53 (2017)
71. Deane, T.: Memorial discourse honours father of computing in Ireland, professor John Byrne (2018). URL www.tcd.ie/news_events/articles/memorial-discourse-honours-father-of-computing-in-ireland-professor-john-byrne
72. Dearing, R.: Higher education in the learning society (1997). URL www.educationengland.org.uk/documents/dearing1997/dearing1997.html
73. Dearing, Ron: The Dearing review (1994). URL www.educationengland.org.uk/documents/dearing1994/dearing1994.html
74. Denny, P., Prather, J., Becker, B.A.: Error message readability and novice debugging performance. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, p. 480–486. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3341525.3387384
75. Denny, P., Prather, J., Becker, B.A., Albrecht, Z., Loksa, D., Pettit, R.: A closer look at metacognitive scaffolding: Solving test cases before programming. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research, Koli Calling '19. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3364510.3366170
76. Denny, P., Prather, J., Becker, B.A., Mooney, C., Homer, J., Albrecht, Z.C., Powell, G.B.: On designing programming error messages for novices: Readability and its constituent factors. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, CHI '21. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3411764.3445696
77. DfEE: The national curriculum handbook for secondary teachers in England (1999)
78. Dickson, P.E., Brown, N.C.C., Becker, B.A.: Engage against the machine: Rise of the notional machines as effective pedagogical devices. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, p. 159–165. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3341525.3387404
79. Dickson, P.E., Richards, T., Becker, B.A.: Experiences implementing and utilizing a notional machine in the classroom. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2022, p. 850–856. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3478431.3499320
80. Dillane, J., Karvelas, I., Becker, B.A.: Portraits of programmer behavior in a frame-based language. In: Proceedings of the 10th Computer Science Education Research Conference, CSERC '21, p. 49–56. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3507923.3507933
81. Draper, S., Maguire, J.: The different types of contributions to knowledge (in CER): All needed, but not all recognised. ACM Trans. Comput. Educ. (2021)
82. Draper, S.W.: What are learners actually regulating when given feedback? British Journal of Educational Technology **40**(2), 306–315 (2009)

83. Eisenstadt, M.: A user-friendly software environment for the novice programmer. Commun. ACM **26**(12), 1058–1064 (1983)
84. EPSRC: Computer Science Inside... enthusing and informing potential computer science students (2005). URL gow.epsrc.ukri.org/NGBOViewGrant.aspx?GrantRef=EP/D507219/1. Publisher: Engineering and Physical Sciences Research Council, Polaris House, North Star Avenue, Swindon, SN2 1ET
85. EPSRC: Securing the future: Expanding the cs4fn (Computer Science for Fun) Project (2007). URL gow.epsrc.ukri.org/NGBOViewGrant.aspx?GrantRef=EP/F032641/1. Publisher: Engineering and Physical Sciences Research Council, Polaris House, North Star Avenue, Swindon, SN2 1ET
86. Erskine, S., Harmon, D.: Eurostudent Survey VII report on the social and living conditions of higher education students in Ireland (2019)
87. Evershed, D., Rippon, G.: High level languages for low level users. The Computer Journal **14**(1), 87–90 (1971)
88. Faherty, R., Nolan, K., Quille, K.: A collaborative online micro:bit K-12 teacher PD workshop. In: Proceedings of the 2020 ACM Conference on International Computing Education Research, ICER '20, p. 307. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3372782.3408113
89. Faherty, R., Quille, K., Becker, B.A.: Comparing the programming self-efficacy of teachers using CSLINC to those teaching the formal national curriculum. In: Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 2, ITiCSE '22, p. 619. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3502717.3532130
90. Faherty, R., Quille, K., Vivian, R., McGill, M.M., Becker, B.A., Nolan, K.: Comparing programming self-esteem of upper secondary school teachers to CS1 students. In: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE '21, p. 554–560. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3430665.3456372
91. Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F., McGill, M.M., Quille, K.: An international benchmark study of K-12 computer science education in schools. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '19, p. 257–258. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3304221.3325535
92. Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F., McGill, M.M., Quille, K.: An international comparison of K-12 computer science education intended and enacted curricula. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research, Koli Calling '19. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3364510.3364517
93. Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F., McGill, M.M., Quille, K.: An international study piloting the measuring teacher enacted computing curriculum (METRECC) instrument. In: Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE-WGR '19, p. 111–142. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3344429.3372505
94. Fincher, S., Ben-David Kolikant, Y., Falkner, K.: Teacher learning and professional development (2019)
95. Fincher, S.A., Robins, A.V. (eds.): The Cambridge Handbook of Computing Education Research. Cambridge Handbooks in Psychology. Cambridge University Press, Cambridge (2019). https://doi.org/10.1017/9781108654555
96. Finnie-Ansley, J., Denny, P., Becker, B.A., Luxton-Reilly, A., Prather, J.: The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. In: Australasian Computing Education Conference, ACE '22, p. 10–19. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3511861.3511863
97. Fisher, L., Byrne, J.R., Tangney, B.: Teacher experiences of learning computing using a 21st century model of computer science continuing professional development. In: Proceedings of the 8th International Conference on Computer Supported Education, pp. 273–280 (2016)

98. Fisher, L., Oldham, E., Millwood, R., FitzGibbon, A., Cowan, P.: Recognising and addressing inertia affecting teacher education: A case study considering computer science in the Republic of Ireland. Journal of the World Federation of Associations of Teacher Education **1**(3a), 81–102 (2016)
99. Fothergill, R.: The director's view. British Journal of Educational Technology **18**(3), 181–93 (1987)
100. Fothergill, R., Anderson, J.: Strategy for the microelectronics education programme (MEP). Programmed Learning and Educational Technology **18**(3), 120–129 (1981)
101. Furlong, J., Lunt, I.: Education in a federal UK. Oxford Review of Education **42**(3) (2016). https://doi.org/10.1080/03054985.2016.1184867
102. Gallacher, J., Raffe, D.: Higher education policy in post-devolution UK: More convergence than divergence? Journal of Education Policy **27**(4), 467–490 (2012)
103. Gardner, J., Fulton, J., Megarity, M.: The in-service education of teachers (INSET) in information technology (IT). Tagg (Eds.), Computers in Education. ECCE **88** (1988)
104. Gilbert, L.: Microelectronics in education: Two types of innovation, two strategies. International Journal of Man-Machine Studies **17**(1), 3–14 (1982)
105. Glanville, G., McDonagh, P., Becker, B.A.: Efforts in outreach programmes to inform secondary students on studying ICT at third level: Providing a realistic experience in coursework and assessment. In: Proceedings of the 6th International Conference on Engaging Pedagogy, ICEP. Dublin, Ireland (2013). URL icep.ie/paper-template/?pid=98
106. Grier, D.A.: When computers were human. In: When Computers Were Human. Princeton University Press (2013)
107. Groeneveld, W., Becker, B.A., Vennekens, J.: Soft skills: What do computing program syllabi reveal about non-technical expectations of undergraduate students? In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, p. 287–293. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3341525.3387396
108. Groeneveld, W., Becker, B.A., Vennekens, J.: How creatively are we teaching and assessing creativity in computing education: A systematic literature review. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2022, p. 934–940. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3478431.3499360
109. Guzdial, M., du Boulay, B.: The history of computing. The Cambridge Handbook of Computing Education Research **11** (2019)
110. Hammerman, R., Russell, A.L.: Charles Babbage, Ada Lovelace, and the Bernoulli numbers. In: Ada's Legacy: Cultures of Computing from the Victorian to the Digital Age. ACM and Morgan & Claypool (2015). URL doi.org/10.1145/2809523.2809527
111. Hanna, N., Guy, K., Arnold, E.: The diffusion of information technology: Experience of industrial countries and lessons for developing countries, vol. 281 (1995)
112. Hartree, D.R., Newman, M., Wilkes, M.V., Williams, F.C., Wilkinson, J., Booth, A.D.: A discussion on computing machines. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences pp. 265–287 (1948)
113. Heavin, C., Fitzgerald, B., Trauth, E.: Factors influencing Ireland's software industry, pp. 235–252 (2003). https://doi.org/10.1007/978-0-387-35695-2_15
114. Her Majesty's Stationery Office: Information technology from 5 to 16 (1989). URL www.educationengland.org.uk/documents/hmi-curricmatters/infotech.html
115. Hijón-Neira, R., Connolly, C., Palacios-Alonso, D., Borrás-Gené, O.: A guided Scratch visual execution environment to introduce programming concepts to CS1 students. Information **12**(9) (2021). https://doi.org/10.3390/info12090378. URL www.mdpi.com/2078-2489/12/9/378
116. Hijón Neira, R., Garcia-Iruela, M., Connolly, C.: Developing and assessing computational thinking in secondary education using a TPACK guided Scratch visual execution environment. International Journal of Computer Science Education in Schools **4**(4), 3–23 (2021). https://doi.org/10.21585/ijcses.v4i4.98. URL www.ijcses.org/index.php/ijcses/article/view/98

117. Hooper, R.: Two years on: The national development programme in computer assisted learning: Report of the director. Council for Educational Technology for the United Kingdom (1975)
118. Horn, C.: Professor John Byrne: Reminiscences: The father of computing in Ireland. Independently published (2017)
119. Howe, J.A., Du Boulay, B.: Microprocessor assisted learning: Turning the clock back? Programmed Learning and Educational Technology **16**(3), 240–246 (1979)
120. Huggins, J.K.: Engaging computer science students through cooperative education. ACM SIGCSE Bull. **41**(4), 90–94 (2010). https://doi.org/10.1145/1709424.1709454
121. Irish Department of Education and Science: Blueprint for the future of ICT in Irish education: Three year strategic action plan 2001 to 2003. An Roinn Oideachais agus Eolaíochta / Department of Education and Science (2001). URL books.google.ie/books?id=a1n2MgEACAAJ
122. James, M.: National curriculum in England: The first 30 years, part 1 (2018). URL www.bera.ac.uk/blog/national-curriculum-in-england-the-first-30-years-part-1
123. Jones, S.P., Bell, T., Cutts, Q., Iyer, S., Schulte, C., Vahrenhold, J., Han, B.: Computing at school. International comparisons. Retrieved May **7**, 2013 (2011)
124. Kallia, M., Cutts, Q.: Re-examining inequalities in computer science participation from a Bourdieusian sociological perspective. In: Proceedings of the 17th ACM Conference on International Computing Education Research, pp. 379–392 (2021)
125. Kallia, M., Cutts, Q., Looker, N.: When rhetorical logic meets programming: Collective argumentative reasoning in problem-solving in programming. In: Proceedings of the 2022 ACM Conference on International Computing Education Research V. 1, pp. 120–134 (2022)
126. Karvelas, I., Becker, B.A.: Sympathy for the (novice) developer: Programming activity when compilation mechanism varies. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2022, p. 962–968. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3478431.3499347
127. Karvelas, I., Dillane, J., Becker, B.A.: Compile much? A closer look at the programming behavior of novices in different compilation and error message presentation contexts. In: United Kingdom & Ireland Computing Education Research Conference., UKICER '20, p. 59–65. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3416465.3416471
128. Karvelas, I., Li, A., Becker, B.A.: The effects of compilation mechanisms and error message presentation on novice programmer behavior. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20, p. 759–765. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3328778.3366882
129. Kemp, N., Lawton, W.: A strategic analysis of the Scottish higher education sector's distinctive assets. Edinburgh: British Council Scotland (2013)
130. Khan, T.M., Nabi, S.W.: English versus native language for higher education in computer science: A pilot study. In: 21st Koli Calling International Conference on Computing Education Research, pp. 1–5 (2021)
131. Kirwan, C.: The machine in the ghost: An educational design research study that explores the teaching of computational thinking to Irish second-level students. Ph.D. thesis, Dublin City University (2021)
132. Kirwan, C., Connolly, C.: Computer science education in Ireland: Capacity, access and participation. In: Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 2, ITiCSE '22, p. 610. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3502717.3532127
133. Knuth, D.E., Pardo, L.T.: The early development of programming languages. A history of computing in the twentieth century pp. 197–273 (1980)
134. Kölling, M., Quig, B., Patterson, A., Rosenberg, J.: The BlueJ system and its pedagogy. Computer Science Education **13**(4), 249–268 (2003). https://doi.org/10.1076/csed.13.4.249.17496
135. Land, F.: The first business computer: A case study in user-driven innovation. IEEE Annals of the History of Computing **22**(3), 16–26 (2000)

136. Land, F.: Early history of the information systems discipline in the UK: An account based on living through the period. Communications of the Association for Information Systems **36**(1), 26 (2015)
137. Larke, L.R.: Agentic neglect: Teachers as gatekeepers of England's national computing curriculum. British Journal of Educational Technology **50**(3), 1137–1150 (2019)
138. Lavington, S.: Early computing in Britain. Springer (2019)
139. Lavington, S.H.: Early British computers: The story of vintage computers and the people who built them. Manchester University Press (1980)
140. Lean, T.: Electronic dreams: How 1980s Britain learned to love the computer. Bloomsbury Publishing (2016)
141. Lehtimäki, T., Monahan, R., Mooney, A., Casey, K., Naughton, T.J.: Bebras-inspired computational thinking primary school resources co-created by computer science academics and teachers. In: Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1, ITiCSE '22, p. 207–213. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3502718.3524804
142. Lillington, K.: Intel turned Leixlip into Ireland's Silicon Valley. The Irish Times (2013). URL www.irishtimes.com/business/intel-turned-leixlip-into-ireland-s-silicon-valley-1.1593495
143. Loksa, D., Margulieux, L., Becker, B.A., Craig, M., Denny, P., Pettit, R., Prather, J.: Metacognition and self-regulation in programming education: Theories and exemplars of use. ACM Trans. Comput. Educ. (2021). https://doi.org/10.1145/3487050. Just Accepted
144. Luxton-Reilly, A., Simon, Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C.: Introductory programming: A systematic literature review. In: Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2018 Companion, p. 55–106. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3293881.3295779
145. López-Pernas, S., Saqr, M., Apiola, M.: Scientometrics: A Concise Introduction and a Detailed Methodology for the Mapping of the Scientific Field of Computing Education. In: Past, Present and Future of Computing Education Research. Springer, Rochester, NY (2023)
146. Maguire, J., Cutts, Q.: Back to the future: Shaping software engineering education with lessons from the past. ACM Inroads **10**(4), 30–42 (2019)
147. Mahon, J., Quille, K., Mac Namee, B., Becker, B.A.: A novel machine learning and artificial intelligence course for secondary school students. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2, SIGCSE 2022, p. 1155. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3478432.3499073
148. Manches, A., Plowman, L.: Computing education in children's early years: A call for debate. British Journal of Educational Technology **48**(1), 191–201 (2017)
149. Matula, D.: Who's in SIGCSE? ACM SIGCSE Bull. **2**(5) (1970)
150. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., Laxer, C., Thomas, L., Utting, I., Wilusz, T.: A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In: Working group reports from ITiCSE, Innovation and Technology in computer Science Education, pp. 125–180 (2001)
151. McGarr, O.: The development of ict across the curriculum in irish schools: A historical perspective. British Journal of Educational Technology **40**(6), 1094–1108 (2009)
152. McGarr, O., Exton, C., Power, J., McInerney, C.: What about the gatekeepers? School principals' and school guidance counsellors' attitudes towards computer science in secondary schools. Computer Science Education **0**(0), 1–18 (2021). https://doi.org/10.1080/08993408.2021.1953296
153. McGarr, O., McInerney, C., Exton, C., Power, J.: Exploring teachers' professional development to support the roll-out of computer science in Irish second-level schools (2020)
154. McGregor, N.: Business growth, the internet and risk management in the computer games industry. In: Changing the Rules of the Game, pp. 65–81. Springer (2013)
155. McInerney, C.: Second level computer science teacher self-efficacy and how it influences the use of teaching and assessment strategies. Ph.D. thesis (2021)

156. McInerney, C., Exton, C., Hinchey, M.: A study of high school computer science teacher confidence levels. In: Proceedings of the 15th Workshop on Primary and Secondary Computing Education, WiPSCE '20. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3421590.3421614
157. McInerney, C., Lamprecht, A.L., Margaria, T.: Computing camps for girls – A first-time experience at the University of Limerick. In: A. Tatnall, M. Webb (eds.) Tomorrow's Learning: Involving Everyone. Learning with and about Technologies and Computing, pp. 494–505. Springer International Publishing, Cham (2017)
158. McLoughlin, H., Hely, K.: Teaching formal programming to first year computer science students. ACM SIGCSE Bull. **28**(1), 155–159 (1996). https://doi.org/10.1145/236462.236530
159. Michaelson, G.: Teaching programming with computational and informational thinking (2015)
160. Miller, J.E.: Notes from the editor. ACM SIGCSE Bull. **15**(2) (1983)
161. Millwood, R., Bresnihan, N., Walsh, D., Hooper, J.: Primary coding: Review of literature on computational thinking (2018). URL ncca.ie/en/resources/primary-coding_review-of-literature-on-computational-thinking/
162. Millwood, R., Oldham, E.: Computer science in schools in England and Ireland–Context and current developments in 2017. Redin-Revista Educacional Interdisciplinar **6**(1) (2017)
163. Millwood, R., Strong, G., Bresnihan, N., Cowan, P.: Ctwins: Improving computational thinking confidence in educators through paired activities. In: Proceedings of the 11th Workshop in Primary and Secondary Computing Education, WiPSCE '16, p. 106–107. ACM, NY, NY, USA (2016). https://doi.org/10.1145/2978249.2978269
164. Mooney, C., Becker, B.A.: Sense of belonging: The intersectionality of self-identified minority status and gender in undergraduate computer science students. In: United Kingdom & Ireland Computing Education Research Conference., UKICER '20, p. 24–30. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3416465.3416476
165. Mooney, C., Becker, B.A.: Investigating the impact of the COVID-19 pandemic on computing students' sense of belonging. In: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education, SIGCSE '21, p. 612–618. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3408877.3432407
166. Mooney, C., Becker, B.A., Salmon, L., Mangina, E.: Computer science identity and sense of belonging: A case study in Ireland. In: Proceedings of the 1st International Workshop on Gender Equality in Software Engineering, GE '18, p. 1–4. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3195570.3195575
167. Moynihan: Computer education: Ireland - a case study (1986). URL hdl.handle.net/2134/10837. Online; accessed 31 August 2020
168. Moynihan, C.: The Irish software industry 1989–2008: An overview of its development (2008)
169. Mulholland, P., Eisenstadt, M.: Using software to teach computer programming: Past, present and future (1998)
170. Nolan, K., Bergin, S.: The role of anxiety when learning to program: A systematic review of the literature. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli Calling '16, p. 61–70. ACM, NY, NY, USA (2016). https://doi.org/10.1145/2999541.2999557
171. Nolan, K., Faherty, R., Quille, K., Becker, B.A., Bergin, S.: CSinc: An inclusive K-12 outreach model. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research, Koli Calling '19. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3364510.3366156
172. Nolan, K., Mooney, A., Bergin, S.: An investigation of gender differences in computer science using physiological, psychological and behavioural metrics. In: Proceedings of the Twenty-First Australasian Computing Education Conference, ACE '19, p. 47–55. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3286960.3286966
173. Nolan, K., Quille, K., Becker, B.A.: CSLINC a nationwide CS MOOC for second-level students. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science

Education V. 2, SIGCSE 2022, p. 1100. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3478432.3499069
174. O'Callaghan, G., Connolly, C.: Developing creativity in computer science initial teacher education through design thinking. In: United Kingdom & Ireland Computing Education Research Conference., UKICER '20, p. 45–50. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3416465.3416469
175. Oldham, E., Cowan, P., Millwood, R., Strong, G., Bresnihan, N., Amond, M., Hegarty, L.: Developing confident computational thinking through teacher twinning online. International Journal of Smart Education and Urban Society (IJSEUS) **9**(1), 61–75 (2018)
176. O'Shea, T., Self, J.: Learning and teaching with computers: The artificial intelligence revolution. Prentice Hall Professional Technical Reference (1986)
177. Parkinson, J., Cutts, Q.: Investigating the relationship between spatial skills and computer science. In: Proceedings of the 2018 ACM Conference on International Computing Education Research, pp. 106–114 (2018)
178. Parkinson, J., Cutts, Q.: Relationships between an early-stage spatial skills test and final CS degree outcomes. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, pp. 293–299 (2022)
179. Parslow, T.: CS departments. email correspondence (2022)
180. Passey, D.: Early uses of computers in schools in the United Kingdom: shaping factors and influencing directions. In: Reflections on the History of Computers in Education, pp. 131–149. Springer (2014)
181. Perry, C.: Coding in schools (2015). URL www.niassembly.gov.uk/globalassets/documents/raise/publications/2015/education/3715.pdf
182. Porter, L., Bouvier, D., Cutts, Q., Grissom, S., Lee, C., McCartney, R., Zingaro, D., Simon, B.: A multi-institutional study of peer instruction in introductory computing. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, pp. 358–363 (2016)
183. Prather, J., Becker, B.A., Craig, M., Denny, P., Loksa, D., Margulieux, L.: What do we think we think we are doing? Metacognition and self-regulation in programming. In: Proceedings of the 2020 ACM Conference on International Computing Education Research, ICER '20, p. 2–13. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3372782.3406263
184. Prather, J., Margulieux, L., Whalley, J., Denny, P., Reeves, B.N., Becker, B.A., Singh, P., Powell, G., Bosch, N.: Getting by with help from my friends: Group study in introductory programming understood as socially shared regulation. In: Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1, ICER '22, p. 164–176. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3501385.3543970
185. Prather, J., Pettit, R., Becker, B.A., Denny, P., Loksa, D., Peters, A., Albrecht, Z., Masci, K.: First things first: Providing metacognitive scaffolding for interpreting problem prompts. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, p. 531–537. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3287324.3287374
186. Quille, K., Bergin, S.: Programming: Predicting student success early in CS1. a re-validation and replication study. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE 2018, p. 15–20. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3197091.3197101
187. Quille, K., Bergin, S.: CS1: How will they do? How can we help? A decade of research and practice. Computer Science Education **29**(2-3), 254–282 (2019). https://doi.org/10.1080/08993408.2019.1612679
188. Quille, K., Bergin, S.: Promoting a growth mindset in CS1: Does one size fit all? A pilot study. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, p. 12–18. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3341525.3387361
189. Quille, K., Bergin, S., Mooney, A.: Press#, a web-based educational system to predict programming performance. International Journal of Computer Science and Software Engineering (IJCSSE) **4**(7), 178–189 (2015). URL mural.maynoothuniversity.ie/6503/

190. Quille, K., Culligan, N., Bergin, S.: Insights on gender differences in CS1: A multi-institutional, multi-variate study. In: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '17, p. 263–268. ACM, NY, NY, USA (2017). https://doi.org/10.1145/3059009.3059048
191. Quille, K., Faherty, R., Becker, B.A.: Building K-12 teacher capacity to expand uptake in a national CS curriculum. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2, SIGCSE 2022, p. 1086. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3478432.3499063
192. Quille, K., Faherty, R., Bergin, S., Becker, B.A.: Second level computer science: The Irish K-12 journey begins. In: Proceedings of the 18th Koli Calling International Conference on Computing Education Research, Koli Calling '18. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3279720.3279742
193. Quille, K., Nam Liao, S., Costelloe, E., Nolan, K., Mooney, A., Shah, K.: PreSS: Predicting student success early in CS1: A pilot international replication and generalization study. In: Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1, ITiCSE '22, p. 54–60. ACM, NY, NY, USA (2022). https://doi.org/10.1145/3502718.3524755
194. Quille, K., Nolan, K., Becker, B.A., McHugh, S.: Developing an open-book online exam for final year students. In: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE '21, p. 338–344. ACM, NY, NY, USA (2021). https://doi.org/10.1145/3430665.3456373
195. Robertson, J.: Cheerful confusion and a thirst for knowledge: tales from the primary school computing classrooms. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education, pp. 1–1 (2018)
196. Russell, B.: Computer science course a stimulating experience for students and staff. Irish Times p. 17–17 (1972)
197. Scanlon, D., Connolly, C.: Teacher agency and learner agency in teaching and learning a new school subject, leaving certificate computer science, in Ireland: Considerations for teacher education. Computers & Education **174**, 104291 (2021). https://doi.org/doi.org/10.1016/j.compedu.2021.104291
198. Selinger, M., Austin, R.: A comparison of the influence of government policy on information and communications technology for teacher training in England and Northern Ireland. Technology, Pedagogy and Education **12**(1), 19–38 (2003). https://doi.org/10.1080/14759390300200144
199. Sentance, S.: Moving to mainstream: Developing computing for all. In: Proceedings of the 14th Workshop in Primary and Secondary Computing Education, pp. 1–2 (2019)
200. Sentance, S., Humphreys, S., Dorling, M.: The network of teaching excellence in computer science and master teachers. In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education, pp. 80–88 (2014)
201. Shadbolt, N.: Shadbolt review of computer sciences degree accreditation and graduate employability. London: BIS (2016)
202. Shulman, L.S.: Those who understand: Knowledge growth in teaching. Educational Researcher **15**(2), 4–14 (1986)
203. Sime, M.E., Green, T.R., Guest, D.: Psychological evaluation of two conditional constructions used in computer languages. International Journal of Human-Computer Studies **51**(2), 125–133 (1972)
204. Strong, G., Higgins, C., Bresnihan, N., Millwood, R.: A survey of the prior programming experience of undergraduate computing and engineering students in Ireland. In: IFIP World Conference on Computers in Education, pp. 473–483. Springer (2017)
205. Sullivan, K., Byrne, J.R., Bresnihan, N., O'Sullivan, K., Tangney, B.: Codeplus – designing an after school computing programme for girls. In: 2015 IEEE Frontiers in Education Conference (FIE), pp. 1–5 (2015). https://doi.org/10.1109/FIE.2015.7344113
206. Szabo, C., Falkner, N., Petersen, A., Bort, H., Connolly, C., Cunningham, K., Donaldson, P., Hellas, A., Robinson, J., Sheard, J.: A periodic table of computing education learning

theories. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '19, p. 269–270. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3304221.3325534
207. Szabo, C., Sheard, J., Luxton-Reilly, A., Simon, Becker, B.A., Ott, L.: Fifteen years of introductory programming in schools: A global overview of K-12 initiatives. In: Proceedings of the 19th Koli Calling International Conference on Computing Education Research, Koli Calling '19. ACM, NY, NY, USA (2019). https://doi.org/10.1145/3364510.3364513
208. Tedre, M., Denning, P.J.: The long quest for computational thinking. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli Calling '16, pp. 120–129. ACM, NY, NY, USA (2016). https://doi.org/10.1145/2999541.2999542
209. Tenenberg, J., Fincher, S.: Opening the door of the computer science classroom: The disciplinary commons. In: Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE '07, pp. 514–518. ACM, NY, NY, USA (2007). https://doi.org/10.1145/1227310.1227484
210. The Royal Society: Shut down or restart? The way forward for computing in UK schools. The Royal Society, London (2012)
211. The Royal Society: After the reboot: Computing education in UK schools (2017). URL royalsociety.org/topics-policy/projects/computing-education/
212. Thorne, M.: The legacy of the microelectronics education programme. British Journal of Educational Technology **18**(3), 165–81 (1987)
213. Tshukudu, E., Cutts, Q., Foster, M.E.: Evaluating a pedagogy for improving conceptual transfer and understanding in a second programming language learning context. In: 21st Koli Calling International Conference on Computing Education Research, pp. 1–10 (2021)
214. Tshukudu, E., Cutts, Q., Goletti, O., Swidan, A., Hermans, F.: Teachers' views and experiences on teaching second and subsequent programming languages. In: Proceedings of the 17th ACM Conference on International Computing Education Research, pp. 294–305 (2021)
215. Tshukudu, E., Sentance, S., Adelakun-Adeyemo, O., Nyaringita, B., Quille, K., Zhong, Z.: Investigating K-12 computing education in four African countries (Botswana, Kenya, Nigeria and Uganda). ACM Trans. Comput. Educ. (2022). https://doi.org/10.1145/3554924
216. Tsouroufli, M.: An examination of the Athena SWAN initiatives in the UK: Critical reflections. Palgrave Studies in Gender and Education pp. 35–54 (2019). https://doi.org/10.1007/978-3-030-04852-5_3
217. Turing, A.M.: Computing machinery and intelligence. In: Parsing the Turing Test, pp. 23–65. Springer (2009)
218. Turing, S.: Alan M. Turing: Centenary edition. Cambridge University Press (2012)
219. Uhomoibhi, J.O.: Implementing e-learning in Northern Ireland: prospects and challenges. Campus-Wide Information Systems **23**(1), 4–14 (2006). https://doi.org/10.1108/10650740610639697. Publisher: Emerald Group Publishing Limited
220. University College Dublin: UCD News (1976). https://doi.org/10.7925/drs1.ucdlib_49485
221. University College Dublin College of Science: From early scientific endeavours to today's UCD Science: Towards a history of the UCD College of Science (2015). URL https://www.yumpu.com/en/document/read/55057153/from-early-scientific-endeavours-to-todays-ucd-science
222. Vegas, E., Hansen, M., Fowler, B.: Building skills for life: how to expand and improve computer science education around the world (2021). Available at: www.brookings.edu/essay/building-skills-for-life-how-to-expand-and-improve-computer-science-education-around-the-world/
223. Vivian, R., Quille, K., McGill, M.M., Falkner, K., Sentance, S., Barksdale, S., Busuttil, L., Cole, E., Liebe, C., Maiorana, F.: An international pilot study of K-12 teachers' computer science self-esteem. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, p. 117–123. ACM, NY, NY, USA (2020). https://doi.org/10.1145/3341525.3387418
224. Walker, D.D., Megarry, J.: The Scottish microelectronics development programme. Programmed Learning and Educational Technology **18**(3), 130–135 (1981)

225. Walker, H.M.: Message from the SIGCSE secretary/treasurer. ACM SIGCSE Bull. **27**(4), 1–4 (1995). https://doi.org/10.1145/216511.571912
226. Whetton, C.: A brief history of a testing time: National curriculum assessment in England 1989–2008. Educational Research **51**(2), 137–159 (2009)
227. Yadav, A., Gretter, S., Hambrusch, S.: Challenges of a computer science classroom: Initial perspectives from teachers. In: Proceedings of the Workshop in Primary and Secondary Computing Education, pp. 136–137 (2015)
228. Zingaro, D., Craig, M., Porter, L., Becker, B.A., Cao, Y., Conrad, P., Cukierman, D., Hellas, A., Loksa, D., Thota, N.: Achievement goals in CS1: Replication and extension. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18, p. 687–692. ACM, NY, NY, USA (2018). https://doi.org/10.1145/3159450.3159452

# Computing Education Research in Schools


**Valentina Dagienė, Yasemin Gülbahar, Natasa Grgurina, Sonsoles López-Pernas, Mohammed Saqr, Mikko Apiola, and Gabrielė Stupurienė**

## 1 Introduction

Looking at the last decades, K-12 computing education and computational thinking (CT) have become increasingly popular areas of research in education at large and within computing education research (CER) in general [1, 2]. CT and K-12 are among the top keywords in CER [3], as well as the top discussed computing education topic on social media [4]. While a high trend means a wide repertoire of approaches for CT in K-12, there is still no consensus on what exactly should be taught in classes [5]. Yet, CT may be defined as the thought processes involved in computing [6]. Alternative labels for CT include computational making, and computational participation [7]. Initially, CT was advocated as a general concept that fits education at all levels, nowadays, CT is more and more strongly associated with K-12 computing education [5, 8].

The roots of CT can be traced back to the works of Edsger W. Dijkstra, Donald Ervin Knuth, Seymour Papert's groundbreaking Mindstorms [9], and many others (see e.g., [10]). The new wave of CT was made popular by Jeannette M. Wing (2006) [11], who proposed that learning CT would be beneficial for everyone.

V. Dagienė (✉) · G. Stupurienė
Vilnius University, Vilnius, Lithuania
e-mail: valentina.dagiene@mif.vu.lt

Y. Gülbahar
Ankara University, Ankara, Turkey

N. Grgurina
University of Groningen, Groningen, The Netherlands

S. López-Pernas · M. Saqr · M. Apiola
University of Eastern Finland, Joensuu, Finland
e-mail: mikko.apiola@uef.fi



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_20

After Wing's introduced the term to parlance of CER in 2006, the concept and the number of publications have grown slowly [2]. Nevertheless, Wing's strategic position as a head of US National Science Foundation (NSF) for Computer and Information Science and Engineering has helped initiate CT targeted funding that played a pivotal role in bringing CT to the forefront of CER. Other influential works include those of [10, 12–14].

Many outstanding reviews of K-12 computing education [1, 15] and CT [16] have revealed a rich repertoire of concrete examples and practical teaching ideas, ranging from maker-spaces, constructing of artifacts to informal hacker-events, and various educational technologies, such as block-based programming, educational robotics and educational challenges such as Bebras [17, 18].

Previous analyses have revealed that, out of numerous approaches, the greater part of them focus on rule-driven programming, with some 27.2% of CT articles including programming and coding-related terms [2]. Many researchers have recommended a broader approach than focusing solely on programming skills [19], and modern approaches include design-driven pedagogies to teach machine learning in K-12 [20], well-established pedagogical toolkits such as the CS Unplugged [21, 22], and recent trends also include STEAM integration [23, 24], with focus on computational practices within the strands of data practices, modeling and simulation, computational problem solving, and systems thinking practices [25].

In this chapter, we complement previous reviews by presenting a scientometric view into research on CT in K-12 education. We discuss existing research especially from the following viewpoints. First, we look into curricular issues in K-12 computing education. Second, we turn our attention to teaching programming and coding, with focus on learning tools and environments. Third, we put our focus especially on pedagogy and teaching praxis. Fourth, we concentrate on assessment and evaluation. Fifth, we discuss teacher education. Sixth, we discuss extracurricular activities.

It is important to note that the terminology varies when talking about computing, computer science, informatics, programming, and coding. Informatics is widely used in Europe with reference to computer science. In this report, we mostly use the terms computer science (CS) and computing education research (CER). Also, while coding is a more technical term, programming is a broader concept, and it refers to teaching students how to communicate clearly, plan well-articulated and efficient processes, and work within the parameters set by any problem.

## 2 A Scientometric Overview of Research on Computing Education in Schools

To understand how the field of CER in schools has evolved, we combine bibliometric analysis with a qualitative overview of the main themes and lines of research in

the field. The data behind this chapter was extracted from the dataset described in [26]. The original dataset was downloaded from the Scopus database and contained 16,863 articles about CER broadly. From this dataset, we extracted those articles that contained terms relevant to CER in schools in their title or author keywords; we did not use abstracts to avoid unnecessary noise in our data. After an iterative process, the authors reached a consensus upon the following set of terms: *K-12, child, primary, school, high school, middle school, kid, pre-university, pre-college, pre-K*. The resulting dataset contained 1650 articles.

| MAIN INFORMATION ABOUT DATA | | AUTHORS | |
|---|---|---|---|
| Timespan | 1970: 2021 | Authors | 3011 |
| Sources (journals, books, etc.) | 233 | Author appearances | 5623 |
| Documents | 1650 | Authors of single-authored documents | 217 |
| Average years from publication | 6.97 | Authors of multi-authored documents | 2794 |
| Average citations per documents | 9.933 | AUTHORS COLLABORATION | |
| Average citations per year per doc | 1.247 | Single-authored documents | 246 |
| References | 37617 | Documents per author | 0.548 |
| DOCUMENT TYPES | | Authors per document | 1.82 |
| Article | 372 | Co-Authors per documents | 3.41 |
| Book chapter | 6 | Collaboration index | 1.9 |
| Conference paper | 1272 | | |

## *2.1 Venues*

There were 233 dissemination venues (journals, conferences, etc.). The venues were largely fragmented where 142 (60.94%) of the venues had a single article, 29 (12.45%) had two articles and only 62 (26.61%) had more than two articles. The average age of an article in the dataset was 7 compared to 15 years in computing education in general, indicating that most of the research about computing in schools in our dataset is recent due to the accelerated interest in the topic. The average number of citations for each article was 9.9 compared to 7.8 in CER in general. A two-sample t-test revealed that the difference between the two is statistically significant $t(1756.17) = 2.65$, $p = 0.008$. The main source of articles comes from conference proceedings, with 1272 (77%) articles, followed by 372 (22.5%) journal articles and a small fraction from book chapters. Some 3011 authors have contributed to the articles: most of them were collaborative while only 217 (13.2%)

were single-authored. The average number of collaborators in co-authored articles was 3.4 which is higher than CER in general which was 2.7. Some 2109 (70.02%) authors had only a single publication, and 419 (13.91%) had two publications. Only 484 (16.07%) had more than two publications. The annual average growth rate[1] in number of publications was 10.8 compared to 11.4 in CER in general, which shows a slightly slower year-to-year growth in K-12 research.

## 2.2 Countries

Only 65 countries had articles about K-12 CER in our dataset: the majority of them (60%) had fewer than 10 articles, 33 countries (50.8%) had five or less articles, twenty countries (33.4%) had two articles or less and twelve countries (18.5%) had a single article. The top productive countries included the US, Germany, UK, Israel, Italy, Finland, Netherlands, Australia, Greece, and Canada (Table 1).

The largest number of articles was produced by US authors which amounts to just below half of all articles in the dataset (883, 47.4%) with the largest number of citations among all countries and an average of 17.9 citations per article. Germany came as a distant second with 133 articles (7.1%) of all articles, 400 citations and an average 8 citations per article. The United Kingdom followed with 96 articles (5.2%) of citations, and an average of citation per article of 14. The remainder of the list contained European countries including Italy, Finland, Netherlands, Greece and a group of developed countries Israel, Australia, and Canada.

**Table 1** The top productive countries

| Country | Count | Percent | Total citations | Avg. article citations |
|---|---|---|---|---|
| US | 883 | 47.4% | 5281 | 17.9 |
| Germany | 133 | 7.1% | 400 | 8.0 |
| United Kingdom | 96 | 5.2% | 449 | 14.0 |
| Israel | 69 | 3.7% | 752 | 18.3 |
| Italy | 52 | 2.8% | 49 | 2.1 |
| Finland | 39 | 2.1% | 128 | 8.0 |
| Netherlands | 37 | 2.0% | 105 | 8.8 |
| Australia | 33 | 1.8% | 107 | 11.8 |
| Greece | 33 | 1.8% | 1139 | 75.9 |
| Canada | 31 | 1.7% | 93 | 10.3 |

[1] $Average\ growth\ rate = \frac{1}{Years-1} \cdot \sum_{y=2}^{Years} \frac{Publications_y - Publications_{y-1}}{Publications_{y-1}}$

## 2.3 Research Themes

In addition to the cleaning process reported in chapter "Scientometrics: A Concise Introduction and a Detailed Methodology for Mapping the Scientific Field of Computing Education Research" to clean authors, affiliations, keywords, and venues, we conducted an additional analysis of the keywords on our subsetted dataset to identify the main research themes in K-12 CER by grouping together keywords that refer to the same concept [26]. Four researchers grouped similar keywords into themes recursively until consensus was reached. For example, the theme "block-based programming" included "Scratch", "Alice", and "block programming", among others. Table 2 shows the most recurring themes identified and the top five most frequent keywords for each theme. The remainder of the chapter is structured based on these research themes and their most representative papers.

**Table 2** Research themes and keywords that are included in each theme

| Theme | Keywords |
|---|---|
| Artificial Intelligence | AI, AI Literacy, Artificial Intelligence, Image Processing, Machine Learning, Natural Language Processing |
| Algorithmic thinking | Algorithm, Algorithmic Patterns, Algorithmic Thinking, Computational and Algorithmic Thinking |
| Assessment and evaluation | Assessment, Advanced Placement, Evaluation, Measurement, Rubric |
| Block-based Programming | Alice, App Inventor, Block-Based Programming, Block Programming, Scratch |
| Computational Thinking | Abstraction, Bebras, CT, Modeling and Simulation, Modelling, |
| Curricular issues | Computer Science Curriculum, CS Curriculum, Computing Curriculum, Curriculum, Educational Standards, Informatics Curriculum |
| Data literacy | Big Data, Data Literacy, Data Management, Data Mining, Data Science |
| Game Development | Game Design/Development, Kodu |
| Games/Gamification | Escape Room, Fantasy, Game-Based Learning, Gamification, Play |
| Extracurricular activities | After-School Programs, Broadening Participation, Informal Education, Outreach, Summer Camp |
| Pedagogical approaches and techniques | Collaborative Learning, Constructionism, Didactics, Pedagogy, Methods/Strategies, Teaching |
| Physical Computing | Arduino, Embedded Systems, E-Textiles, Hardware, Physical Computing |
| Programming | Coding, Debugging, Novice Programmers, Pair Programming, Programming Education |
| Robotics | Robotics, Robots |
| Teacher education and training | Pedagogical Content Knowledge, Professional Development, Teacher, Teacher Education, Teacher Professional Development |

## 3 Curricular Issues

In the last few years, countries from all over the world have started to modify their national curricula to introduce CT [27]. A review of policy initiatives for integrating CT in compulsory education in European countries reveals two reasons behind this movement: (1) to prepare for future employment and fill ICT job vacancies; and (2) to enable students to think in a different way, express themselves using new media and solve real-world problems [28].

The newly launched JRC (Joint Research Council at the European Commission) report "Reviewing Computational Thinking in Compulsory Education" [27] surveyed the European Union countries and found that 18 of the 25 countries have renewed their computing curricula in schools between 2016 and 2021. All these countries refer to programming/coding or algorithmic thinking (or both) implementation in their computing curricula, and so pave the way for developing CT skills. However, not all countries in the world have a computing curriculum in schools.

A short historical overview of the most important dates and contributions to computing curriculum development is presented in Table 3.

**Table 3** Influential contribution to computing education in schools: curricula and standards

| Computing curriculum models | Year | Related publications |
|---|---|---|
| Logo – teaching computing "without curriculum" | 1970 | Solomon et al., 2020 [29] |
| Mindstorms: computers, children, and powerful ideas | 1980 | Papert, 1980 [9] |
| Basics of computing in all schools of Soviet Union | 1985 | Kerr, 1991 [30] |
| First official computing (informatics) curriculum in upper secondary school in Lithuania | 1986 | Dagiene, 1999 [31] |
| First CS curriculum for high school in Israel | 1995 | Gal-Ezer & Harel, 1999 [32] |
| CS and computer engineering curriculum for all secondary schools in Ontario, Canada (grades 9 to 12) | 1999 | Province of Ontario Ministry of Education, 1999; 2000 [33, 34] |
| First edition of ACM's Model Curriculum for K-12 CS, published in 2003 and revised in 2006 | 2003 | Tucker, 2003 [35]; Tucker et al., 2006 [36] |
| CSTA (Computer Science Teachers Association) K-12 CS Standards (revised) | 2011 | Seehorn et al., 2011 [37] |
| CS in K–8: Building a Strong Foundation | 2012 | CSTA, 2012 [38] |
| Computing curriculum in the UK (Shut down or restart?) | 2012 | Britain, 2012 [39] |
| Informatics Education: Europe Cannot Afford to Miss the Boat | 2013 | Informatics Europe and ACM, 2013 [40] |
| Australian Curriculum | 2013 | ACARA, 2022 [41] |
| England National Curriculum in computing | 2013 | Greaves, 2017 [42] |
| CS in UK schools | 2014 | Brown et al., 2014 [43] |
| Informatics Reference Framework for School | 2022 | Caspersen et al., 2022 [44] |

Compared with other computing education at school research initiatives, not so many papers are dedicated to exploring curriculum and its related issues [45]. The scientometric analysis revealed 157 papers within the theme of "curricular issues". Among the top cited of those papers, the first one focuses on designing for deeper learning of computing courses for middle school students [46], while the second one discusses challenges and strategies in the computing curriculum [47]. Next highly cited paper [48] examined vignettes from five countries (Australia, UK, New Zealand, Israel, and Poland) and discussed key issues of curriculum development.

In England, in 2013, the national curriculum of ICT was replaced with Computing in all primary and secondary schools. The new subject was implemented in schools as of September 2014. Greaves [42] examined issues such as insufficient training of teachers, a lack of content and pedagogical knowledge and a lack of time and support of teachers to teach the new computing curriculum. Anwar et al. [49] explored the context of K-12 computing education research in high, middle, and low income countries with focus on four countries in South Asia (Bangladesh, Nepal, Pakistan, and Sri Lanka), and found that research mostly targeted other than lower or middle income countries.

### *3.1 Computational Thinking*

In recent years, Computational thinking (CT) in education has drawn a lot of attention. CT skills are being developed by introducing a computing curriculum in schools. We see more and more countries adding computing education to K-12 curricula [27]. As Denning and Tedre [10] pointed out, CT is one of the strongest movements to bring computing into K-12 schools.

Our scientometric analysis yielded 313 papers within the theme of CT, and eight out of them were cited more than a hundred times. Barr and Stephenson [50] wrote a seminal paper about bringing CT into K-12 education, and identified nine core CT concepts: data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation. Furthermore, the paper discusses students' capabilities, dispositions and pre-dispositions, and classroom culture that are beneficial for the CT learning experience. Román-González et al. [51] identify three cognitive abilities (spatial, reasoning, and problem-solving) that contribute to the idea of CT as problem-solving ability. Yadav et al. [52] discuss the key CT constructs and how these ideas are related to educational reforms in the US and provide recommendations for teachers to embed CT in the K-12 classrooms, including how to infuse CT into other disciplines. Using games to support CS education is described in many papers. Repenning et al. [53] suggest to use games to scale up from after school programs into school curriculum. A tool of low threshold and high ceiling, which is based on systemic strategy, scaffold flow, enable transfer, and support equity was developed.

### 3.2 Algorithmic Thinking

Algorithms are one of the oldest and biggest parts of computing education. CT skills are closely intertwined with algorithms and algorithmic thinking. Our scientometric analysis found 158 papers that focus on teaching algorithms to school students with different ages for them to develop algorithmic thinking. Sixteen of these papers focus on algorithmic thinking itself, and seventeen papers mention algorithmic thinking as an aspect of CT, programming, or CS education in general. Other papers deal with teaching and learning algorithms at the primary and secondary school levels.

Futschek and Moschitz [54] present a learning scenario with tangible objects for primary school children to learn the basic concepts of algorithmic thinking. Thomas [55] focuses on the "ability to design, implement, and assess the implementation of algorithms to solve a range of problems" in a project involving African-Americal middle school girls. Ragonis [56] is concerned with algorithmic patterns in the context of teacher education. Finally, a recent paper by Clarke-Midura et al. [57] focuses on the development of the assessment of algorithmic thinking of kindergarten-aged children.

Three areas are of algorithmic thinking research can be distinguished: (1) research based upon Logo, which offers a historical perspective on efforts to teach computing (e.g., [29]), (2) introducing concepts of CT in the early years, and (3) integration with STEM and other disciplines. Observations from research show that by engaging children in thinking about real-world examples such as driverless cars and self-service shop tills, there are opportunities for them to explore, experiment and reflect upon different ideas collaboratively. Tangible objects that use physical interfaces rather than a screen are becoming more widespread in the early years [58]. Other attractive approaches found in the literature use e.g., floor robots (such as Beebot, Dash & Dot, Cubetto, Pixie, Root) that enable children from 4 years old to explore simple commands.

### 3.3 Data Literacy and Artificial Intelligence

The relatively new discipline of data science is concerned with generating information and knowledge from various forms of data, which is important given the increasing role of big data. Our scientometric analysis yielded 27 papers mentioning keywords that are linked in one way or another to implementing data science or databases in K-12 computing curriculum.

Out of the most cited works [59], deals with the development of one of the first curricular modules about the big data for middle school. This module was a part of a CS Principles-based middle school curriculum in the US. In Germany, one of the first curricula for data science in secondary education was introduced in 2018 [60]. This curriculum was constructed from scratch and contained four modules: (1) from

data to information, (2) big data and artificial intelligence, (3) data projects, and (4) data science and society.

Few years later, Mike et al. [61] proposed a data science curriculum for high schools in Israel involving introductory topics to data science, machine learning and artificial neural networks, examining images as data, data and tables, with in-depth analyses of fundamental algorithms like the perceptron algorithm, the K-nearest neighbors' algorithm, and the support vector machine algorithm.

Artificial intelligence (AI) is also increasingly finding its way into K-12 education. Our scientometric analysis identified 37 papers concerned with AI and machine learning (ML). We see a clear distinction between the papers, which report learning with AI and those concerned with learning about AI. In the former category, two highly cited papers cite using chatbots to enhance students' learning.

Bigham et al. [62] describe a project where blind students design instant messaging chatbots and provide general design principles for such projects. Benotti et al. [63] use chatbots to foster engagement of students learning basic CS concepts. Further papers of employing AI include Vachovsky et al. [64] who describe an AI summer program aimed at fostering diversity in CS, and Zimmermann-Niefield et al. [65] who report students making ML models for gesture-controlled interactive media.

The most cited paper listed in our dataset on AI and ML provides in-depth analysis on definitions of AI literacy through a framework for teaching learner-centered AI [66]. The presented framework contains 17 competences related to AI and ML: recognizing AI, understanding intelligence, interdisciplinarity, general vs. narrow, AI's strengths and weaknesses, imagine future AI, representations, decision-making, ML steps, human role in AI, data literacy, learning from data, critically interpreting data, action and reaction, sensors, ethics and programmability. The framework contains 15 design considerations relevant to teaching AI, ML and robotics.

Kandlhofer et al. [67] developed an education concept to foster AI literacy for different age groups on different education levels in Austria. Sabuncuoglu [68] provides a curriculum to teach AI in middle school in Turkey. In Finland, Vartiainen et al. [20] present a pedagogical framework for teaching ML in middle school.

Arguably the most well-known initiative regarding AI education is described by Touretzky et al. [69] where a joint working group was formed from the Association for the Advancement of AI and the CSTA develop guidelines to teach AI in K-12 in the US, the initiative AI4K12 (https://ai4k12.org/) was emerged. In the Netherlands, AI-related learning occurs as an elective theme within CS in the upper grades of secondary education [70]. In 2019, this theme was renamed as cognitive science, and included basics of intelligent behavior, features of cognitive computing, and application of cognitive computing.

## 4 Programming Tools, Languages and Environments

A variety of tools and environments are being developed for teaching and learning computing in schools and even in preschool or kindergarten education. Both tools and environments can be tangible or virtual (using a screen). Our scientometric analysis of the research on programming education yielded 300 papers revealing a diverse use of tools, programming languages and environments for different age groups. It is important to note that separating tools, programming languages, and learning environments is not always straightforward. For example, unplugged activities and physical manipulatives can count both as tools and environments [71].

### *4.1 Short Overview of Programming Tools and Environments*

Since educational content must be simplified for younger learners, teaching often starts with tangible objects and manipulatives in pre-school and first years of primary school, then followed by block-based programming in the primary and finally text-based programming in secondary school. Many tools, especially physical devices (robotics, computing devices, board games, and programming toys), are intended for the development of CT in young learners.

Ching et al. [72] presented a review of empirical studies on the educational technologies and the analyses of selected technologies for CT skills in this age group, where they classified educational tools and technologies as either being electronic physical enacting agents or as programming with or without manipulatives. The programming manipulatives (robots, electronic blocks, storybooks with stickers, and devices controlled by buttons) are designed for the youngest learners, starting from the age of 3–4 years old to lower primary grades. Coding with fairy tales, game-based learning, paper-and-pencil activities, creative drama activities, maze games and tinkering activities come to the fore in the pre-school and first years of primary school.

Teaching programming brings new pedagogical approaches along with new tools and environments. Table 4 shows a selection of educational technologies classified by the existence of manipulatives for programming, and the existence of programming tools on screen.

Environments that allow instant application integrated into the content narrative, editors that remind and hint while writing code, and integrated development environments that enable us to observe the values of variables, output and errors, if any, actually support the learning and reinforcement process by containing different learning techniques.

Pedagogical approaches appear not only in the context of instructional design, but also in a way that is integrated into the tools and environments in which we teach coding and programming. The structure of Blockly and code.org based on goal-oriented and problem-based games, or the structure of Scratch software that allows

**Table 4** Programming with physical manipulatives and programming on screen

| Tangible programming or Programming with manipulatives | Block-based programming | Text-based programming |
|---|---|---|
| Turtle geometry robot [29] | ScratchJr [82] | Logo [29, 93] |
| Electronic textiles, and wearables [73–76] | Scratch [83–86] | Java [83, 94] |
| Robotics [76, 77, 78] | App Inventor [16, 87–89] | Python [95–97] |
| Arduino [79, 80] | Net Gadgeteer [90] | BlueJ [94] |
| Board games [81] | Alice [91, 92] | |

for open-ended scenario-based story or game design are examples of pedagogy-injected tools and environments.

Moreover, there are tools and environments for switching between block-based coding and text-based programming, so in case the student is competent in coding it is easier to jump in and understand the program [98].

## *4.2 Block-Based Programming*

The introduction of block-based programming environments changes the landscape of introductory tools, replacing questions of syntactic features of textual languages with the larger question of whether text-based programming altogether is the best way to introduce novices to programming. Our scientometric analysis revealed data from 143 papers related to "block-based programming". Among the top cited of those papers is a paper with 283 citations about the three-dimensional block programming environment Alice used for creating animations for generic and storytelling purposes [91].

The findings by Weintrop & Wilensky [99] show that students in the block's condition had shown greater learning gains and a higher level of interest in future computing courses. Students in the text condition viewed their programming experience as more similar to what professional programmers do and as more effective at improving their programming ability.

Block-based programming environments are becoming increasingly common in introductory programming courses, but to date, little comparative work has been done to understand if and how this approach affects students' emerging understanding of fundamental programming concepts. Block-based programming tools like ScratchJr [100], *Scratch* [101], and Blockly are becoming commonplace in introductory programming contexts, with a growing number of new curricula utilizing blocks-based programming tools in their materials.

The scientometric data also reveals numerous attempts to compare blocks-based and text-based programming environments in an introductory high school programming class [84, 98, 102] where the second most cited paper was by Weintrop and Wilensky [103] with 193 citations. This paper investigated perceptions of high

school students about blocks-based programming tools in terms of perceived ease-of-use, and perceived differences between block-based programming and text-based programming. Numerous factors contribute to making block-based programming easy, including the natural language description of blocks, the drag-and-drop composition interaction, and the ease of browsing the language. However, a perceived lack of authenticity and being less powerful were reported as disadvantages when compared with text-based environments.

The third most cited paper (181 citations) with the keyword "block-programming" was that of teaching a summer camp for high school students using *App Inventor*, and two separate methods of visual block programming and Java to create Android applications [92].

Meerbaum-Salant, the fourth most cited author with 177 citations, and colleagues also used block programming and designed learning materials according to the constructionist philosophy of *Scratch* [104]. The evaluation lasted 2 years with several schools, where two taxonomies, namely revised Bloom taxonomy and the structure of the observed learning outcome (SOLO) taxonomy were used. The results showed positive outcomes for teaching CS concepts except several challenges for repeated execution, variables, and concurrency.

The data shows that block-based programming environments are used most intensively at primary education level. Many studies have shown that these environments are successful in the process of acquiring CT skills [86, 105], such as in acquiring mathematics achievement, programming self-efficacy perception, CT skills, and spatial thinking skills [106], while other studies have shown gains in mathematics, language learning, motivation, self-confidence, and critical thinking [107, 108]. Furthermore, it is seen that 3D block-based programming environments (like Alice) can also be used successfully [91]. Three-dimensional design and coding improve students' spatial thinking and spatial intelligence skills [51].

### *4.3 Text-Based Programming*

Our data shows some 300 papers within the broad theme of "programming". Very few of those papers focus on text-based programming, but rather on either design or pedagogy, or a comparison between text-based and another kind of teaching approach. Moritz et al. [109] conducted a study and proposed a design-first approach instead of an object-first approach, and they developed an intelligent tutoring system based on design-first approach to help students in CS courses. Mentioning that the curriculum requires the teaching of text-based programming from age 11 onwards, Sentance and Waite [110] described an approach to teaching programming that they call PRIMM (predict-run-investigate-modify-make). Being a methodology, this approach is also found to be useful for dealing with levels of abstraction, and tracing and code comprehension.

Yet in another study in our analysis, Blanchard et al. [111] mentioned that students who traditionally learn to program using text-based languages should

concurrently master both syntax and semantics and switching from block-based to text-based may be a challenge for some learners. Students' performance increased due to the use of dual-modality programming environments which provide both block and text representation to make transition between environments easily. On the other hand, it is known that Python is the most widely used as an advanced text-based programming language [97]. Python, which is preferred due to its proximity to the spoken language, enables the development of very advanced applications such as data analytics and machine learning with its rich library options, and allows learners to learn without fear of programming and without developing negative attitudes.

Due to their old history, the languages used in text-based programming teaching also ended up in object-oriented programming languages where both the pedagogical approach as well as the tools and environments changed accordingly. Hence, integrated development environments, and dual-mode programming became accessible and usable for not only transition from block-based to text-based programming but also for physical programming.

### *4.4 Physical Computing and Robotics*

Our scientometric analysis revealed data from 58 papers within the theme of "physical computing" and 41 papers within "robotics'". Regarding physical programming, the most cited research was about the use of electronic textiles to introduce key concepts and practices of CT where they developed a curriculum using Lilypad Arduino as a tool [112]. The results revealed that this approach can broaden participation in computing within the context of scaffolded challenges and using crafts materials and activities.

Another article by Kafai et al. [113], which is focused on ethnocomputing, was again about the use of electronic textiles where the focus is on culturally responsive computing. Their purpose was to leverage traditional crafting and sewing practices while teaching engineering and computing concepts in line with local indigenous knowledge. The tools used here were sewable microcontrollers that can be connected to sensors and actuators by stitching circuits with conductive thread.

Hence, among the top 5 cited research there were four papers about the implementation of electronic textiles unit for CS curriculum, where students designed wearable electronic textile projects with microcontrollers, sensors, and leds [79, 114]. On the other hand, Sentance et al. [115] reported the first study about the usability and affordances on BBC micro:bit, which is a portable and low-cost programmable device delivered to all 11–12 year-old students in the UK Their findings revealed the benefits of physical computing in terms of usability, creativity, tangible structure and learning gains.

Among 41 papers with the keyword "robotics", the top cited research by Ludi and Reichlmayr [77] explored the robotics programming tools and materials which are designed with an accessibility focus for visually impaired participants based on the availability of Lego Mindstorm NXT Kits. They indicated that regardless

of having CS courses, the students' interest and confidence increased via robotics activities. Sapounidis et al. [116] compared the performance of one tangible and one isomorphic graphical system for robot introductory programming activities on learning outcomes. The results were in favor of a tangible system where the children were reported to produce fewer errors, made more effective debugging, and needed less time to accomplish the robot programming tasks. Moreover, children were also found to be more engaged, created more complicated programs and explored different commands and parameters more actively.

To summarize, many studies reveal that robotic kits and physical programming help achieve a wider range of learning outcomes, especially since robotics education include authentic, project and problem-based teaching activities with an interdisciplinary approach [74, 117]. Some findings such as increasing motivation in students, concretizing the teaching content, and providing an interdisciplinary perspective are just some of the findings reached in such studies [75]. Thus, comprehending programs is key to learning programming, it can be done within physical computing in high school [73, 79].

## 5 Pedagogical Approaches and Techniques

Teaching computing in schools brings new pedagogical approaches along with new tools and environments. Analyses of the selected literature have shown that most approaches used for different disciplines are also applied for teaching programming and coding. New methods and approaches suitable for the automation processes in computing have emerged. Innovative and sound practices have been added to this wide range of pedagogical approaches, which allow a rich learning environment to be offered for learners with different personal characteristics and learning preferences, in the context of a variety of tools and environments. The scientometric analysis revealed data from 231 papers that have "pedagogy" as a theme. The most frequently conducted research is about game development [118] and collaborative problem-solving strategies, e.g., pair programming [119].

Another approach that is frequently used at intermediate and advanced levels is "modeling". In particular, modeling provides important contributions to concretize abstract concepts or to make complex processes understandable [120]. Processes that become more complex as programming becomes advanced are supported by techniques such as PRIMM (Predict-Run-Investigate-Modify-Make) or Use-Modify-Create [121]. As the problem to be solved becomes more complex, the cognitive load increases, so it becomes difficult to teach new concepts and processes.

### *5.1 Pair Programming and Collaboration*

Pair programming is one of the most frequently used and successful approaches in computing education [119], where multimodal student behavior and performance

data are modeled. Peer instruction encourages students at intermediate and advanced levels to work in groups, allows students to learn by employing more than one pedagogical approach and build meaningful knowledge [122]. In long-term projects, many different techniques make it easier for students to learn, such as assigning different responsibilities to group members, keeping an engineering notebook, and revealing differences between versions in software improvements [123]. Research on pair programming [124] has found that compared to students who work alone, students who work in pairs have shown increased confidence in their work.

The results of the study focusing on collaborative pair programming approach conducted by Tsan et al. [125] have shown that pairs vary in terms of their collaboration process. While some student pairs naturally made suggestions in the dialogue during swapping control, some other pairs needed substantial scaffolding during the transition from "driver" to "navigator". Based on this few and many other research studies, it is obvious that either with the infusion of collaboration or use of teaching methods and techniques, the student engagement can be increased. Increased engagement will change the status of many other variables during the learning process and possibly will result in positive outcomes and more learning gains [126]. Hence, more research should be conducted to reveal the effects of various pedagogies for teaching programming.

Tsan et al. [127] have investigated collaborative problem solving by examining the gender composition of collaborative groups in computing classes. The findings of this study reported significant differences in the quality of artifacts produced by learner groups depending upon their gender composition. They stressed the importance of future study of such factors for providing equitable computer science education to learners of all ages.

## *5.2 Inquiry Based Learning*

Successfully enacting an inquiry-based approach to teaching requires extensive pedagogical content knowledge and understanding of student culture to create, facilitate, and assess the opportunities that activate learning for students. Inquiry requires well-designed and carefully planned classroom practices in order for students to have the necessary opportunities to make connections between key concepts across a discipline by exploring computing principles based on an inquiry-instruction approach [128].

The family of approaches that can be considered inquiry-based includes project-based learning, problem-based learning, and design-based learning [129]. Though each approach has its own unique characteristics, they all share a common premise that students should be actively engaged in building their own learning through meaningful long-term learning experiences. Learning scientists and educational researchers point to a set of studies that conclude that students learn more deeply and perform better on complex tasks if they can engage in more authentic learning [130].

Students who learn subject matter through an inquiry-based approach demonstrate increased performance on intellectually challenging performance tasks when compared to their direct-instruction peers [129].

Inquiry-based learning also leads to significant gains in problem-solving abilities, curiosity, creativity, independence, and positive feelings about school. Furthermore, educators note that engaging students in the context of complex, meaningful projects that require sustained engagement, critical thinking, collaboration, communication, and management of resources to develop final products or presentations builds the types of twenty-first century skills.

## *5.3 Games and Gamification and Game Development*

The scientometric analysis revealed data from 35 papers that have "game development" as a theme and 45 papers that have "games" or "gamification". The most cited paper was written by Papastergiou [118] (984 citations), and it is devoted to the use of game-based learning in computing teaching. In his study, Papastergiou compared two groups as gaming and non-gaming and found that the gaming approach was both more effective in students' learning of computer memory concepts and more motivational than the non-gaming approach.

Repenning et al. [131], being among the top cited researchers, developed a scalable game design curriculum for effective integration of computing education into the regular school curriculum. The approach called "computational thinking pattern analysis" for assessing and correlating CT skills relevant to game design and simulations was designed and tested. Findings revealed positive outcomes like rapid adoption of the curriculum by teachers from different disciplines, high student motivation, and high levels of participation and interest.

Many researchers worked on correlation of game design concepts and CT aspects. Basawapatna et al. [132] examined the effect of the end-user game design tools. Their study explored if students recognize CT patterns from their game programming experience by applying the acquired "know-how" when creating science simulations and reaching positive outcomes.

Werner et al. [133] investigated the relationship of computer game programming to computational learning by analyzing 231 games programmed by 325 school students (11–12 years old).

An extra-curricular CT training approach for primary school students based on the integration of both unplugged and plugged activities in a gamified environment was elaborated by Tsarava et al. [134]. In both types of activities, the researchers tried to clarify the association between CT-based solving of real-life problems and aspects of different STEM disciplines. This approach incorporated principles of constructionism in combination with game-based and project-based learning, and students were guided to construct their knowledge through playing and interacting with interdisciplinary educational scenarios.

### *5.4 Problem-Based Learning and Project-Based Learning*

Problem solving and fruitful collaboration are two important skills in K-12 education in the twenty-first century. The project-based learning among the 231 selected papers, 76 papers contain "project" ("projects" or "project-based") as learning approach, 26 papers contain "collaboration", and 22 papers contain "problem solving" as keywords. Problem-based learning as well as project-based learning and exploratory learning processes are widely preferred. Project-based learning and project as a form of instruction has clear connections with problem-based learning among others [135]. Both approaches focus on participants' collaboration and achieve a shared goal. Students are engaged with a project and can encounter problems which need to be addressed in order to construct and present the results in response to the estimated goal [136].

Project-based learning is a widely used approach in computing education. However, the way students develop their projects may differ in terms of the process and the product. Romeike and Göttel [137] stated that the majority of models and examples for project-based lessons rely on a traditional software development approach like the waterfall model and suggested a new approach in secondary computing education which uses the concept of didactic transposition to adapt agile software development methods for all stages of project development and implementation.

The main difference between the two approaches is that students in problem-based learning are primarily focused on the process of learning, while students in project-based learning are concentrating on the achievement of the end product [138]. Project-based learning has also been compared with collaborative learning, and the importance of student collaboration, reflection, redrafting and presentations are emphasized [139]. As Helle et al. [135] argue, project work is a collaborative form of learning, as all participants need to contribute to the results, and it also has elements of inquiry-based learning with active engagement rather than passive experiences. In addition, project-based activities such as digital game design, algorithm development and debugging, which support the construction of knowledge, also play an important role in the process.

## 6 Assessment and Evaluation

Computing education researchers studying curriculum implementation in schools and student learning want to evaluate how students apply their CS knowledge. Focus on assessment tools and methods have enormously increased during the last decade when European and other countries started to promote CS and CT in schools. According to the scientometric search, there are 126 papers focusing on computing performance assessment issues in schools. The most cited papers are

from conferences: ACM Technical Symposium on CS Education, SIGCSE and ITiCSE - Innovation and Technology in CS Education.

### *6.1 Assessment Approaches*

For computing education, the evaluation of the process is as important as the evaluation of the end result. There are various methods used to assess students' computing knowledge and skills. Researchers and practitioners work on finding the most effective assessment approaches and try to discover innovative approaches. In computing education, it is essential that the measurement and evaluation of students' performance in the class be learner-centered. Therefore, evaluation approaches lead to an authentic evaluation of the process and product [140].

In design-oriented learning environments based on constructivist approach, peer assessment requires frequent assessment in various forms, including self-assessment, authentic assessment, portfolio, and reflection. During computer education applications, students take an active role in problem solving processes and learn by exploring as well as by collaborating with their peers [141]. Hence, Dr. Scratch or Scratch can be used as a tool for assessing CT skills [41, 105, 142].

Furthermore, assessment is applied in the context of activities that reflect a real-life situation. Commonly used authentic assessment methods and techniques are as follows: discussions, observations, classification questions, paper-and-pencil tests, portfolios, performance tasks, checklists, and rating scales [143, 144]. Werner et al. [92] developed a problem-solving-based performance measurement tool to measure the following CT skills: algorithmic thinking, abstraction, and modeling with the help of a grading scale while students' complete tasks such as coding, debugging, and re-editing the code.

Teachers use different assessment methods in schools to measure students' understanding of a particular subject and to determine the extent to which learning objectives have been met. Evaluation in computing education is a complex process due to the nature of the CT understanding and the use of different tools and environments and includes the problem solving and design skills of CT. Due to the difficulty of accurately measuring problems, project-based and design-based learning processes with traditional assessment methods, performance-based and open-ended assessments are preferred.

A conceptual model for CS and CT assessment was elaborated by Tikva & Tambouris [145]; it includes the following approaches: (1) self-report methods; (2) tests; (3) artifact analysis; (4) observations; and (5) frameworks. Tang et al. [146], while doing a literature review, arranged the CT assessment approaches into four groups: (1) the traditional test composed of selected or constructed items-questions; (2) portfolio assessment; (3) interviews; and (4) surveys.

Different approaches and model suggestions have been discussed in the literature to evaluate CT skills. Studies on evaluating CT mostly focus on analyzing the final product (game or project) developed by students using this skill [147, 148].

Another evaluation strategy for CT skills is to reorganize the existing program's code to complete a specific task [92]. Scenarios for detecting problems and debugging the current program are also among the effective methods used to measure students' fluency in computer programming and computer-based problem solving. Some researchers have stated that multiple-choice questions and grading scale and assessment methods can be used to evaluate school students' CT skills [149]. In addition, the design-based approach (such as programming interactive media) is presented as an indispensable element of evaluation systems.

## *6.2 Assessment Tools: Tests and Scales*

During the last decade a lot of computing education researchers have designed and analyzed various tools for assessment skills in CS and especially in CT education. Román-González et al. [149] have classified the assessment tools into several groups. Later Djambong et al. [150] reported on a three-year study of innovative practices targeting the development of a visual data flow programming language for the introduction of CT skills, and the development of a testing method based on a selection of tasks and its application to measuring CT skills in K-12 education. We provide a detailed analysis of the most frequently used assessment tool groups complemented by examples (Table 5).

**Table 5** Classification of computing and CT assessment tools used in literature

| Tool group | Examples from literature |
|---|---|
| Diagnostic tools | Question based tests [51, 151]<br>Systems of assessments [46] |
| Summative assessment | Aptitude-based CT Test - CTt [51]<br>Test for Measuring Basic Programming Abilities [151]<br>Commutative Assessment Test [152]<br>e-portfolio [153] |
| Formative assessment | Dr. Scratch [154]<br>CT pattern graph [155] |
| Tools targeting ability to transfer CT skills into the real-life context | Cognitive abilities-based [104]<br>Tasks of the Bebras challenge [17, 156]<br>CT Pattern Quiz [132, 156] |
| CT data-mining tools | Using educational games [148]<br>CT pattern analysis [131] |
| Measurement of perceptions and attitudes | CT Scales – CTS [157]<br>Automatic recognition of CT [155]<br>Self-efficacy perception scale [158] |
| Assessment of the vocabulary | Measuring students' verbal skills when coding [159] |

Román-González et al. [51] developed an assessment tool, so called CTt (CT test) which includes the following components: sequences, loops, events, parallelism, conditionals, operators, data computational practices, problem-solving practices that occur in the process of programming, experimenting and iterating required task, testing and debugging, reusing and remixing, abstracting and modularizing. In addition to the CTt, Román-González et al. [51] mentioned two other skills tests, one measuring basic programming abilities and the other one measuring CT concepts. The first test is aimed at measuring the students' ability to execute a given program based on the so-called "flow control structures" which are considered at the core of the CT for this age group: sequencing, selection, and repetition [151]. The second test, called by Commutative Assessment, aims to measure students' understanding of different computational concepts, depending on whether they occur through scripts written in visual (block-based) or textual programming languages, which is a key transition to reach higher levels of code literacy [152].

So called "Bebras tasks" can be used as another assessment tool for CT skills [41]. The Bebras challenge refers to the analytic and apply levels of the taxonomy, i.e., general analytical thinking.

When the studies focus on evaluation of coding and programming as the core skills of CT, it is observed that assessment approaches show variety according to the measurement of the skill, measurement of the related activities or projects, or measurement of both [25, 160, 161]. Although formative evaluations that focus more on the process are made as a result of these measurements, product-oriented summative evaluations are also necessary.

## *6.3 Suggestions to be Taken in Mind*

Considering the fact that raising computational thinkers develops rapidly in different contexts and these differences lead to different approaches in terms of evaluation dimension, also taking into account increasing attention to computing assessment in schools and booming publications, we would like to share some points that can be noted about evaluation processes.

- The best forms of assessment are those that are useful to learners. For this reason, approaches that are suitable for the level of the student and that support the learning process should be preferred. The evaluation process should include the creation and critical review of comprehensive projects.
- Using a project collection or portfolio enriches the assessment, sharing assessment criteria with students beforehand will help draw students' attention to specific points.
- Focusing on the process provides an opportunity to explore CT that cannot be fully represented by blocks or codes. Students can talk about their experiences in presentations, incorporate audio-recorded annotations into their projects, screen-record the development processes, teach others what they know, or participate in retrospective or real-time interviews.

– Taking a formative approach to evaluation may require evaluating CT, coding and programming skills from different perspectives, at different times, with different criteria and with different approaches during the design process.
– Teachers should use level-determining and formative assessment methods in schools together. In short, both the process and the product should be evaluated, and sufficient, instructive and guiding feedback should be provided to the students in both processes.

CT skills, by their nature, bring some difficulties and challenges related to evaluation [160]. The uniqueness of computer programs, the time-consuming evaluation process and the fact that computer programs are one of the most difficult end products to evaluate are among the difficulties experienced in the evaluation process. Other potential difficulties with assessment are the lack of variety of resources needed to assess, the lack of online access to resources, the need for automated assessment tools, computer-based simulations, and lack of professional development seminars for teachers to learn about assessment.

## 7 Teacher Education and Training

Living in the knowledge age and facing rapid changes in technological developments, teachers in particular are expected to be life-long learners to be provided with up-to-date knowledge and skills as well as the abilities to transform education with emerging and innovative tools and technologies. Having this in mind, our scientometric analysis revealed data from 257 papers that have "teacher education" as a theme: 210 of them are conference papers, 45 are journal articles and two are book chapters. A significant portion of them, 116, originate from the US, while Germany, United Kingdom, Israel, and the Netherlands contribute about ten each. The rest of the papers are contributions from another 30 countries, mostly from Europe, Latin America, Middle Eastern, and Far Eastern countries.

When the top cited and recent research were explored, it was seen that different models are considered for providing professional development to teachers and support K-12 teachers in building knowledge needed for sustainable CS teaching.

The top cited paper on this topic with 191 citations was written by Yadav and his colleagues [162] in the US. They mentioned the importance of providing teachers with an adequate knowledge base about computational thinking and how to incorporate it into their teaching. For this reason, they developed CT modules and assessed their impact on preservice teachers' understanding of CT concepts, and attitude towards computing. The findings revealed that this kind of intervention has a positive impact on preservice teachers' understanding of CT concepts.

The paper by Brown et al. [43] from the UK, as the second most cited research with 142 citations, also regards the teacher training issue as a significant forthcoming challenge and described their efforts for a national network of teaching excellence which is provided as a solution to this problem. Similarly, in a study

conducted by Yadav et al. [52] computing teachers' perspectives on the demands of teaching CS and support needed to ensure quality teaching were investigated. Among a number of challenges reported by teachers were isolation, lack of adequate CS background, and limited professional development resources which points out the need for collaboration between teachers and experts and the need for continuous training.

Mouza et al. [163] proposed a model in which during school-university partnerships undergraduates assist teachers in computing in their classrooms as part of a university service-learning course. Findings indicated that undergraduates were able to connect knowledge of computing to pedagogy and technology to assist teachers in the implementation of CS instruction.

Hubbard and D'Silva [164] propose a model that computing teachers could be taken from a pool of in-service teachers trained in other disciplines. While these transitioning teachers can learn about computing pedagogy and subject matter at professional learning workshops, daily teaching experiences would also be a source of their learning. In order to find out how specific teaching practices support teachers' learning, with a focus on pedagogical content knowledge, teachers and volunteers from the technology sector were assigned teaching responsibilities. However, teachers' engagement and learning on integration of CT is affected not just by the design of the professional development course, but also by the knowledge and beliefs teachers brought with them.

Recently, Jocius et al. [165] proposed 3C (Code, Connect, Create) professional development model which was designed to support teachers while infusing CT into their classrooms. The model is evaluated by the participation of teachers from different disciplines, and results showed that the 3C professional development model supported integration process, and increased teachers' self-efficacy and beliefs regarding CT integration into disciplinary content.

Goode et al. [128] proposed a professional development program (Exploring CS) which focused on immersion into inquiry, teacher-learner-observer model, professional learning community, equitable practices and professional development into the classroom and throughout the year. The researchers also urged the importance of research about the effectiveness of professional development models and provided a research methodology, so that we can fulfill the premise of broadening participation and engagement in computing.

## 8 Extracurricular Activities

In many countries, computing education is not a (compulsory) subject in a school curriculum. Thus, various extracurricular ways to introduce computing to school students starting from early ages are used. Our scientometric analysis found 155 articles concerned with extracurricular activities by looking at keywords "outreach", "after-school programs", "summer camp", "code camp", "informal education", "summer workshop", "code club" or similar variants. Majority of the articles are

published in the proceedings of the two ACM conferences, SIGCSE and ITiCSE. One hundred and seventeen of the papers originate from the US, six from Finland, four from the United Kingdom, three from Argentina, Greece, Italy, and Saudi Arabia each, and the others from another 12 countries, mostly European and/or English speaking.

The activities described concern summer camps – both for students and teachers, additional extracurricular courses for students, and competitions. The most prominently stated purpose of the outreach activities is broadening participation - mentioned in 44 papers, with specific focus on female students in 22 papers. Sixteen papers focus on teachers, and four on competitions.

The top cited articles [128, 166–168] equilibrate between computing education curriculum and outreach activities. Exploring CS educational reform programs are presented and analyzed. The core of the Exploring CS activities is devoted to the problem solving, computational practices, and modes of inquiry associated with doing computing, rather than just a narrow focus on coding, navigating particular syntax or tools. Computing and data analysis, robotics, web design, introductory programming, problem solving, and human computer interaction are among the most dedicated topics [169].

The majority of the top cited ACM conference papers are dedicated to summer camps or summer schools on various computing activities. Annually organized olympiads and contests in informatics are a huge movement in computing education involving many school students and creating all kinds of training activities. Olympiads and contests in informatics can be taken as a serious didactical approach of CER. Involving school students in recognition of informatics as a science discipline should be one of important targets. However, olympiads in informatics are not well presented from a research viewpoint.

## *8.1 Summer Camps on Programming*

Programming is a popular extracurricular activity for school-age children and can be taught as an extracurricular activity, for example at summer coding camps, online courses that students can follow at home, after-school clubhouses, Code Clubs or CoderDojos. Summer schools or camps are mostly designed with an exploratory research orientation. Majority of articles published in ACM conferences SIGCSE and ITiCSE focus on involving girls.

Many researchers investigated introducing programming activities during summer camps or workshops (in our scientometric analysis 40 papers explore summer camps). The Imaginary Worlds Girls Camp focuses on providing girls with a positive computing experience and introducing them to the basic ideas of programming [170]. Expectation was to undermine the stereotype of CS as “geeky” and introduce young women to the concepts of object-based programming. Among presented results, the most important were that students found the campus highly motivating and enjoyable to be able to create their own animated movies, also that the Alice IDE

essentially eliminates the syntax errors that make programming such a headache for novices. Both boys and girls enjoyed using Alice. However, by holding single-sex camps, a social environment in which the girls could support and encourage one another is more suitable.

Educators are often seeking new ways to motivate or inspire school students to learn. Robotics and media computation are the most popular contexts for teaching CS in K-12 schools. With the deep interest in mobile technologies among teenagers, some researchers have focused on using smartphones as a new context [87] and suggested applying App Inventor on summer campus for 1 week. App Inventor also integrates with Android smartphones and tablets, which enables the user-made applications to be tested on a physical device.

The research team [171] has run 3 years projects and explored the effect of summer code camps on student interest and self-efficacy. They emphasized meaningful uses of computing, focusing primarily on issues of computing for social good. Focus was on a pair of middle-school camps offered in summer 2017 and 2018 in which school students explored computational techniques related to data science and applied their newly-developed skills to explore data sets relating to issues of social good. As the research stated, the camp was generally successful "in making a positive short-term difference in students' sense of who can "do" computing and in their own self-efficacy and interest".

Nite et al. [172] stated that the summer camp/school activities on physical computing (i.e., coding Arduino microcontrollers) provide an opportunity for school children to engage in hands-on learning in STEM in an informal classroom setting. The results of study by Stupurienė et al. [173] showed that carefully planned activities on mini-projects with Arduino during girls' summer school with purposeful support of lecturers reduce the technological anxiety and contribute to the girls' intention to use technology in the future.

## *8.2 Olympiads in Informatics*

Non-formal and outreach computing education are exceptionally useful for motivating and involving school students in CS. The relatively long history of the International Olympiad in Informatics (IOI) proves that computer programming has a very strong attraction to school students making the IOI competition an intellectual fulfillment of the year. IOI focuses mainly on solving algorithmical tasks by using a programming language and tools. Kenderov [174] in his article "Three decades of international informatics competitions (How did IOI start?)" has studied the roots of the IOI.

Verhoeff [175] provided a deep analysis of the IOI tasks of the first 20-year history: he summarized task type and difficulty level and classified them according to concepts involved in their problem and solution domain. Difficulty level is determined based on what percentage of contestants were able to "fully" solve the task (i.e., had a submission with a score of 90% or more).

Kiryukhin and Okulov [176] published a book about all tasks of the first 18 Olympiads in informatics, including the task descriptions, analyses, solution guidance, classifications, and pseudo-code and implementations in Pascal.

The international journal "Olympiads in Informatics" is a refereed scholarly journal that provides an international forum for presenting research and development in teaching and learning computing through competition. Submissions of papers are flexible, and there are no requirements to participate in the conference while paper is accepted. Besides research articles, there are also technical papers and country reports, views, and opinions published. During the years 2007–2021, more than 200 articles were published showcasing about 300 authors from over 50 countries. All papers are open accessible on the IOI website (https://ioinformatics.org/page/ioi-journal/1).

## *8.3 Bebras – The Worldwide Challenge on CS and CT*

The Bebras challenge (www.bebras.org), as an international initiative aiming to promote computing among school students of all ages, provides a lot of data for making inquiries on how students accept CS concepts, how they develop computational and algorithmic thinking [177, 178]. Some countries have published overviews of tasks with detailed explanations on how to solve them and what CS concepts are behind [156, 179]. There are articles for promoting Bebras challenge in particular countries and also articles dealing with particular contest results [180–182].

Many other activities have been developed under the Bebras umbrella: hands-on seminars for students and teachers, discussions for deepening informatics knowledge and teamwork on developing Bebras tasks, so that the competition idea was changed to a broader Bebras challenge on informatics and CT. Combéfis and Stupurienė [183] discuss the results of a survey conducted among Bebras community aimed to identify existing activities using Bebras tasks.

The crucial point of the Bebras challenge is CS concept-based tasks [12, 184]. Bebras community (teachers and researchers) are seeking to create interesting tasks to motivate students to deal with CS and to think deeper about what makes the core of computing [185]. Agreements on task development criteria are settled [186–188]. Several types of tasks are used in the challenge: interactive (dynamic) tasks, open-ended tasks and multiple-choice tasks are the most common groups [187].

Short CS concept-based tasks solving is a powerful approach that can support a pedagogical shift in the classroom and foster students' engagement in computing. Many publications deal with problem solving methods [189, 190]. Problem solving of the short tasks can be considered as a systematic process involving pupils into deeper understanding of computing concepts. The short task solving can be one of the strategies that engage and motivate students for in-depth learning and fosters deeper thinking skills [12].

Studies of various countries on Bebras activities started to be developed year by year. The Bebras community collects publication lists and publishes on the Bebras website annually (https://www.bebras.org/publications.html). If an article is open access, the copy or link to the full article is provided.

### *8.4 Unplugged CS Activities*

There are many good ways for introducing computing in schools. CT can be seen as an innate human ability exercised daily by using computational tools and performing routine everyday procedures [10]. CS Unplugged movement (https://www.csunplugged.org), seeking to teach computing concepts and practices through games, magic tricks, and kinesthetic activities, made teaching computing in schools more alive. Established in the late 1990s by Bell et al. [21], the movement has gained a worldwide following and influenced the design of computing education in schools of many countries [191].

The unplugged approach is the only one possible for a huge number of schools around the world that do not have basic technology infrastructure [192], such as electricity. Most experiences using unplugged activities aim to foster learners' interest in CS. In addition, if evidence of the effectiveness of the unplugged approach are found, it would reinforce the theory that CT is mainly a problem-solving cognitive process/ability, which is possible to develop not only through computer programming.

The paper of Brackmann et al. [28] summarizes a quasi-experiment carried out in two primary schools in Spain, and show that students in the experimental groups, who took part in the unplugged activities, enhanced their CT skills significantly more than their peers in the control groups who did not participate during the classes. Proving that the unplugged approach may be effective for the development of this ability.

Another similar study [193] aimed to evaluate whether the inclusion of unplugged activities favors the CT skills of primary school students. The study leads to the conclusion that CS unplugged activities significantly support students CT skills acquisition, and also increase the motivation of students to learn computing topics.

## 9 Discussion and Conclusion

K-12 education is one of the most researched areas of CER, and growing rapidly, especially with the popularity and adoption of CT education in schools. To develop CT skills, education policy makers and school communities are introducing computing curricula, and an increasing number of countries are moving beyond teaching CS in the upper grades to teaching CT in primary school or even in pre-

school education. Peter J. Denning and Matti Tedre summarized in their essay "Computational Thinking" [10] that CT is one of the strongest movements bringing computers courses into all K-12 schools. CT has brought enormous changes to science, transforming research methods and tools, influencing the epistemology of science. Along with CT, recent advances CS topics, such as artificial intelligence, machine learning, and data science, are also increasingly finding their way into computing curricula in schools.

In this chapter, we have presented an overview of K-12 CER, using a scientometric lens to discover and analyze relevant research. Our analysis reveals that K-12 CER is a rising trend: articles in the K-12 domain have an average age of 7 years as compared to 15 years in all CER, K-12 articles are significantly more cited than general CER, and K-12 articles have more collaborators than general CER. The total number of articles in our dataset of K-12 CER was 1650, with top countries including: US, followed by Germany, Israel, and the United Kingdom.

Research on teaching programming and coding is one of the largest parts of computing education, accounting for nearly a third of the articles. However, contemporary innovative approaches are gaining ground that focus not only on the development of programming skills, but also include design-based pedagogy, constructionism, well-established pedagogical toolkits such as CS Unplugged or Bebras short tasks. Over the last decade, STEAM integration is receiving increasing attention focusing on computational practices in the areas of data literacy and data mining, modelling and simulation, automation of problem solving and systems thinking practices.

Our scientometric analysis revealed research on K-12 being conducted from a number of viewpoints. We looked at computing curriculum issues in the K-12 context. We highlighted the peculiarities of learning programming and coding, the different tools and techniques, discussed variety of programming learning environments. We paid special attention to pedagogy, didactic methods and teaching practices. We know that assessment and evaluation have a very important place in CER, and we have devoted a great deal of research to it, especially now that researchers are trying to find appropriate ways of assessing CT skills. We have discussed research on teacher education and training. We focused quite a lot on extracurricular activities in CS, highlighting the popular summer camps in the US to teach programming, the informatics olympiads, the CS Unplugged activities, and the Bebras approach for solving short task on CS concepts. Below are highlighted several of the most important insights.

First, a crucially important and active area of research in K-12 is that of curriculum and curriculum design. While many countries are renewing their national curricula, other countries are still lacking a decent curriculum to teach K-12 computing. Our scientometric analysis shows that some 157 papers in our dataset have included "curriculum" (or related) as their keyword, with the top cited papers focusing on discussions on pedagogies, strategies and challenges in curriculum design, and comparisons between curricula in different countries. The findings from data were reflected based on a historical overview of curriculum development. The data shows known issues that still complicate curriculum design,

including insufficient teacher training, and lack of learning content and pedagogical knowledge. There is also very little research on introducing newer topics such as data science, artificial intelligence (AI) and machine learning (ML) into computing courses in K-12 education.

Second, the analysis confirms that research on CT has gained increased popularity within the K-12 domain. A total of 313 papers (around 19% of papers in the K-12 dataset) dealt with the theme of CT, confirming the rising relevance of CT as a topic of research [2]. In addition to CT, the results show that some 158 papers were related to algorithmic thinking, a domain that is much intertwined with CT. Moreover, some 27 papers were found to deal with teaching of data science and data practices, and 37 papers on teaching AI or ML. The relatively small but rising trend of teaching data science and machine learning deserves special attention. While entire populations are nowadays living and growing in the middle of various AI and ML systems, understanding the basics of how such systems work and how they are designed, seems to have been given only limited attention in CER. While almost the entirety of CER has, over the decades, focused on how to teach classical programming, this new track of research on AI and ML in CER has fundamental importance.

Third, the analysis reveals a wide repertoire of tools. Some 300 articles report research on programming-related tools, 143 papers deal specifically with block-based programming and related tools, many papers are on text-based programming, with 58 in physical programming, and 41 papers on educational robots and robotics. A growing number of studies now show that young children (from 3 years old to primary school) learn by using programming manipulatives (robots, electronic blocks, storybooks with stickers and push-button devices). The scientometric findings on tools were reflected against a classification of tools in CT, including a range of tools on the three categories of tangible programming, block-based programming, and text-based programming. In all, a wide repertoire of educational tools are available, and research on tools within the K-12 domain is one of the most active research areas.

Fourth, a total of 231 papers cover the topic of pedagogical approaches and techniques. A total of 35 articles are about game development, and 45 involve games or gamification. Project-based learning, problem-based learning, and inquiry-based learning are receiving significant attention as research on pedagogies within CER in the K-12 domain. Computing is very different for various age groups due to its cognitive and constructive structure, especially when the curricular and cultural issues are taken into consideration. This brings variety in terms of not only pedagogical approaches but also topics and tools. Readers must keep in mind that there are many other pedagogical approaches used in teaching computing like unplugged activities, PRIMM, use-modify-create, live coding, etc., which are already known. It is still worth to mention that research reveals the changing role of CS Unplugged activities from being used as a support to existing curriculum, to being used alone which is the main reason why we started to talk about the CT curriculum recently based on research over the past 10–15 years. This is important evidence showing how CS education has changed over the years. For example, the unplugged activities were used as outreach activities once, due to lack of schools

that integrate CT or computing curricula into their courses. Similarly, there was not a sequence of computing lessons that built on one another, in other words no place for construction in most of the schools, other than informal learning opportunities. However, as the computing gained importance and the field has changed, schools and teachers started to integrate computing curricula addressing different pedagogies and tools towards more organized curricula, and their impact became visible so that even "programming" added into digital literacies as a component.

Fifth, assessment and evaluation constitute an important topic of research, with some 126 papers on the topic. We have proposed a summarizing classification of assessment tools, based on previous research, including tools in seven categories, e.g., diagnostic tools, and summative assessment tools (see Table 5). Policy makers, researchers and practitioners are looking for the most effective assessment methods for teaching computing as well as exploring and trying to find innovative approaches. The progressive learning trajectory of students is very important in computing education in schools, and the measurement and evaluation of students' performance in the class should be learner-centered. However, there is also a long way on the assessment and evaluation issues to be researched further due to the eclectic nature of the discipline.

Sixth, the top tracks of research within the K-12 area also include teacher education with some 257 papers, 210 of them are conference papers, 45 are journal articles and two are book chapters. Living in the age of knowledge and faced with rapid changes in technological development, it is expected that today's teachers must be prepared for lifelong learning, up-to-date knowledge and skills, as well as the ability to transform education using innovative tools and approaches.

Finally, a total of 156 papers cover the research on extracurricular approaches. In many countries, the absence of computing education or the low number of classes in schools is compensated by extracurricular ways to introduce computing to learners starting from early ages. In our analysis of many scientific articles, we noticed that formal computing education in K-12 schools is intertwined with outreach activities. One of the most popular activities around the world is CS Unplugged. Another attractive activity to complement the computing education in K-12 schools is the Bebras Challenge on CS and CT, where students of all ages have the opportunity to solve interesting short tasks and learn about a wide variety of computing topics and CS concepts.

Hence, CT and CS education constitute quite a broad discipline with many areas that researchers can address, e.g., the implementation of different pedagogical approaches in different age groups, the importance of using tangibles and programming toys for practicing ideas, the value of informal learning as well as the transformation of curriculum from informal to formal learning environments, and the valid and reliable way of assessing the learning outcomes.

In all, with this chapter we have provided an overview of the most relevant tracks of research within the K-12 computing education domain. There is still a lack of a more in-depth analysis of the historical evolution of CER in K-12, of more in-depth research, especially in the teaching of the most contemporary topics, and of deeper insights in the areas of assessment and evaluation, and teacher training, all of which we suggest for future research.

## References


1. Szabo, C., Falkner, N., Petersen, A., Bort, H., Cunningham, K., Donaldson, P., ... & Sheard, J. (2019). Review and use of learning theories within computer science education research: primer for researchers and practitioners. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 89–109).
2. Saqr, M., Ng, K., Oyelere, S. S., & Tedre, M. (2021). People, ideas, milestones: a scientometric study of computational thinking. *ACM Transactions on Computing Education* (TOCE), 21(3), 1–17.
3. Apiola, M., Saqr, M., López-Pernas, S., & Tedre, M. (2022). Computing Education Research Compiled: Keyword Trends, Building Blocks, Creators, and Dissemination. *IEEE Access*, 10, 27041-27068.
4. Saqr, M., López-Pernas S. & Apiola, M. (2023). Computing Education Research in Social Media, News, Blogs, Patents And Blogs: Capturing The Impact And The Chatter with Altmetrics. In: *Past, Present and Future of Computing Education Research* (in-press). Springer.
5. Lockwood, J., & Mooney, A. (2017). Computational Thinking in Education: Where does it fit? A systematic literary review. *arXiv preprint arXiv:1703.07659*.
6. Aho, A. V. (2012). Computation and computational thinking. *The computer journal*, 55(7), 832-835.
7. Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM*, 59(8), 26-27.
8. Tedre, M., & Denning, P. J. (2016, November). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling international conference on computing education research* (pp. 120-129).
9. Papert, S. A. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic books.
10. Denning, P. J., & Tedre, M. (2019). *Computational thinking*. Mit Press.
11. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
12. Dagienė, V., Futschek, G., & Stupurienė, G. (2019). Creativity in solving short tasks for learning computational thinking. *Constructivist Foundations*, 14(3), 382-396.
13. Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014, June). Computational thinking in K-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference* (pp. 1–29).
14. Pears, A., Tedre, M., Valtonen, T., & Vartiainen, H. (2021, October). What Makes Computational Thinking so Troublesome? In *2021 IEEE Frontiers in Education Conference* (FIE) (pp. 1–7).
15. Garneli, V., Giannakos, M. N., & Chorianopoulos, K. (2015, March). Computing education in K-12 schools: A review of the literature. In *2015 IEEE Global Engineering Education Conference* (EDUCON) (pp. 543–551).
16. Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, 42(1), 38-43.
17. Dagiene, V., & Stupuriene, G. (2016). Bebras - A Sustainable Community Building Model for the Concept Based Learning of Informatics and Computational Thinking. *Informatics in education*, *15*(1), 25-44.
18. Dagienė, V. (2018). Resurgence of informatics education in schools: A way to a deeper understanding of informatics concepts. In *Adventures between lower bounds and higher altitudes* (pp. 522-537). Springer, Cham.
19. Denning, P. J., & Tedre, M. (2021). Computational Thinking: A disciplinary perspective. *Informatics in Education*, 20(3), 361-390.
20. Vartiainen, H., Toivonen, T., Jormanainen, I., Kahila, J., Tedre, M., & Valtonen, T. (2021). Machine learning for middle schoolers: Learning through data-driven design. *International Journal of Child-Computer Interaction*, 29, 100281.

21. Bell, T., Witten, I. H., & Fellows, M. (1998). *Computer Science Unplugged: Off-line activities and games for all ages*. Computer Science Unplugged.
22. Bell, T., & Vahrenhold, J. (2018). CS unplugged—how is it used, and does it work? In *Adventures between lower bounds and higher altitudes* (pp. 497-521). Springer, Cham.
23. Apiola, M., & Sutinen, E. (2020, November). Mindset and Study Performance: New Scales and Research Directions. In *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research* (pp. 1–9).
24. Dolgopolovas, V., & Dagiene, V. (2021). Computational thinking: Enhancing STEAM and engineering education, from theory to practice. *Computer Applications in Engineering Education*, 29(1), 5-11.
25. Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127-147.
26. López-Pernas, Saqr, M., & Apiola, M. (2023). Scientometrics: A concise introduction and a detailed methodology for the mapping of the scientific field of computing education. In: *Past, Present and Future of Computing Education Research* (in-press). Springer.
27. Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., ... & Stupurienė, G. (2022). *Reviewing Computational Thinking in Compulsory Education* (No. JRC128347). Joint Research Centre.
28. Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017, November). Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of the 12th workshop on primary and secondary computing education* (pp. 65–72).
29. Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M. L., Minsky, M., ... & Silverman, B. (2020). History of logo. In *Proceedings of the ACM on Programming Languages*, 4(HOPL), 1–66.
30. Kerr, S. T. (1991). Educational reform and technological change: Computing literacy in the Soviet Union. *Comparative education review*, 35(2), 222-254.
31. Dagiene, V. (1999). Programming-based Solution of Problems in Informatics Curricula. In *Communications and Networking in Education: Learning in a Networked Society* / IFIP WG 3.1 and 3.5 (with 3.6). Aulanko, Hämeenlinna, Finland, June 13–18 (pp. 88–94).
32. Gal-Ezer, J., & Harel, D. (1999). Curriculum and course syllabi for a high-school CS program. *Computer Science Education*, 9(2), 114-147.
33. Province of Ontario Ministry of Education. 1999. *The Ontario Curriculum Grades 9 and 10: Technological Education*. Ontario Ministry of Education. Toronto.
34. Province of Ontario Ministry of Education. 2000. *The Ontario Curriculum Grades 11 and 12: Technological Education.* Ontario Ministry of Education. Toronto.
35. Tucker, A. (2003). *A model curriculum for K-12 computer science: Final report of the acm K-12 task force curriculum committee.* ACM.
36. Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). A model curriculum for K–12 computer science. *Report of the ACM K–12 Task Force Computer Science Curriculum Committee, 2nd Ed*. ACM, New York
37. Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., ... & Verno, A. (2011). *CSTA K–12 Computer Science Standards.* Revised 2011. ACM.
38. CSTA (2012). *Computer Science Teachers Association. Computer Science K-8: Building a Strong Foundation*. https://csteachers.org/documents/en-us/25b0b43f-0817-4c08-9210-f1a96f2b1c7d/1/
39. Britain, G. (2012). Shut down or restart?: *The way forward for computing in uk schools. Royal Society*. Retrieved 2/28/2020, from https://books.google.de/books.
40. Informatics Europe and ACM (2013). Europe cannot afford to miss the boat. Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education. http://www.informatics-europe.org/images/documents/informatics-education-europe-report.pdf
41. ACARA (2022). *Australian Curriculum, Assessment and Reporting Authority*. https://v9.australiancurriculum.edu.au/

42. Greaves, A. (2017, July). What training are primary teachers receiving to implement the new computing curriculum and are they prepared to succeed?. In *2017 Computing Conference* (pp. 1426–1430).
43. Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education* (TOCE), 14(2), 1–22.
44. Caspersen, M. E., Diethelm, I., Gal-Ezer, J., McGettrick, A., Nardelli, E., Passey, D., ... & Webb, M. (2022). *Informatics Reference Framework for School.* https://www.informaticsforall.org/wp-content/uploads/2022/03/Informatics-Reference-Framework-for-School-release-February-2022.pdf
45. Floyd, S. P. (2019, February). Historical high school computer science curriculum and current K-12 initiatives. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 1287–1287).
46. Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education,* 25(2), 199–237.
47. Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, *22*(2), 469-495.
48. Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, 22(2), 445-468.
49. Anwar, T., Jimenez, A., Bin Najeeb, A., Upadhyaya, B., & McGill, M. M. (2020, August). Exploring the Enacted Computing Curriculum in K-12 Schools in South Asia: Bangladesh, Nepal, Pakistan, and Sri Lanka. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (pp. 79–90).
50. Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is Involved and what is the role of the computer science education community? *ACM Inroads,* 2(1), 48–54.
51. Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in human behavior*, 72, 678-691.
52. Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends,* 60(6), 565-568.
53. Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 265–269).
54. Futschek, G., & Moschitz, J. (2011, October). Learning algorithmic thinking with tangible objects eases transition to computer programming. In *International conference on informatics in schools: Situation, evolution, and perspectives* (pp. 155-164). Springer, Berlin, Heidelberg.
55. Thomas, J. O. (2018, February). The computational algorithmic thinking (CAT) capability flow: An approach to articulating CAT capabilities over time in African-American middle-school girls. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (pp. 149-154).
56. Ragonis, N. (2012, July). Integrating the teaching of algorithmic patterns into computer science teacher preparation programs. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education* (pp. 339–344).
57. Clarke-Midura, J., Silvis, D., Shumway, J. F., Lee, V. R., & Kozlowski, J. S. (2021). Developing a kindergarten computational thinking assessment using evidence-centered design: the case of algorithmic thinking. *Computer Science Education,* 31(2), 117-140.
58. Manches, A., & O'Malley, C. (2012). Tangibles for learning: a representational analysis of physical manipulation. *Personal and Ubiquitous Computing*, 16(4), 405-419.
59. Buffum, P. S., Martinez-Arocho, A. G., Frankosky, M. H., Rodriguez, F. J., Wiebe, E. N., & Boyer, K. E. (2014, March). CS principles goes to middle school: learning how to teach "Big Data". In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 151–156).

60. Heinemann, B., Opel, S., Budde, L., Schulte, C., Frischemeier, D., Biehler, R., ... & Wassong, T. (2018, November). Drafting a data science curriculum for secondary schools. In *Proceedings of the 18th Koli calling international conference on computing education research* (pp. 1–5).
61. Mike, K., Hazan, T., & Hazzan, O. (2020, November). Equalizing Data Science Curriculum for Computer Science Pupils. In *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research* (pp. 1–5).
62. Bigham, J. P., Aller, M. B., Brudvik, J. T., Leung, J. O., Yazzolino, L. A., & Ladner, R. E. (2008, March). Inspiring blind high school students to pursue computer science with instant messaging chatbots. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education* (pp. 449–453).
63. Benotti, L., Martínez, M. C., & Schapachnik, F. (2014, June). Engaging high school students using chatbots. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 63–68).
64. Vachovsky, M. E., Wu, G., Chaturapruek, S., Russakovsky, O., Sommer, R., & Fei-Fei, L. (2016). Toward more gender diversity in CS through an artificial intelligence summer program for high school girls. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 303–308).
65. Zimmermann-Niefield, A., Polson, S., Moreno, C., & Shapiro, R. B. (2020, June). Youth making machine learning models for gesture-controlled interactive media. In *Proceedings of the Interaction Design and Children Conference* (pp. 63–74)
66. Long, D., & Magerko, B. (2020). What is AI Literacy? Competencies and Design Considerations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–16).
67. Kandlhofer, M., Steinbauer, G., Hirschmugl-Gaisch, S., & Huber, P. (2016, October). Artificial intelligence and computer science in education: From kindergarten to university. In *2016 IEEE Frontiers in Education Conference* (FIE) (pp. 1–9).
68. Sabuncuoglu, A. (2020, June). Designing one year curriculum to teach artificial intelligence for middle school. In *Proceedings of the* 2020 ACM ITiCSE *Conference* (pp. 96–102).
69. Touretzky, D., Martin, F., Seehorn, D., Breazeal, C., & Posner, T. (2019, February). Special session: AI for K-12 guidelines initiative. In *Proceedings of the 50th ACM technical symposium on computer science education* (pp. 492–493).
70. Barendsen, E., Grgurina, N., & Tolboom, J. (2016, October). A new informatics curriculum for secondary education in the Netherlands. In *International conference on informatics in schools: Situation, evolution, and perspectives* (ISSEP) (pp. 105-117). Springer, Cham.
71. Horn, M., & Bers, M. (2019). Tangible computing. *The Cambridge handbook of computing education research*, 1, 663-678.
72. Ching, Y. H., Hsu, Y. C., & Baldwin, S. (2018). Developing computational thinking with educational technologies for young learners. *TechTrends*, 62(6), 563-573.
73. Jayathirtha, G. (2018, January). Computational concepts, practices, and collaboration in high school students' debugging electronic textile projects. In *Conference Proceedings of International Conference on Computational Thinking Education* (pp. 1–7).
74. Litts, B. K., Kafai, Y. B., Lui, D., Walker, J., & Widman, S. (2017, March). Understanding high school students' reading, remixing, and writing codeable circuits for electronic textiles. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 381–386).
75. Lui, D., Kafai, Y., Litts, B., Walker, J., & Widman, S. (2020). Pair physical computing: high school students' practices and perceptions of collaborative coding and crafting with electronic textiles. *Computer Science Education*, 30(1), 72-101.
76. Merkouris, A., Chorianopoulos, K., & Kameas, A. (2017). Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *ACM Transactions on Computing Education* (TOCE), 17(2), 1–22.
77. Ludi, S., & Reichlmayr, T. (2011). The use of robotics to promote computing to pre-college students with visual impairments. *ACM Transactions on Computing Education* (TOCE), 11(3), 1–20.

78. Dummer, G., Savelsbergh, E., & Drijvers, P. (2020, October). Entering the car park-primary pupils' understanding of programmed control systems in real world situations. In *Proceedings of the 15th Workshop on Primary and Secondary Computing Education* (pp. 1–3).
79. Jayathirtha, G., & Kafai, Y. B. (2021, June). Program Comprehension with Physical Computing: A Structure, Function, and Behavior Analysis of Think-Alouds with High School Students. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education* (V. 1) (pp. 143–149).
80. Yoo, W. S., Pattaparla, S. R., & Shaik, S. A. (2016, March). Curriculum development for computing education academy to enhance high school students' interest in computing. In *2016 IEEE Integrated STEM Education Conference (ISEC)* (pp. 282–284).
81. Kafai, Y., & Vasudevan, V. (2015, June). Hi-Lo tech games: crafting, coding and collaboration of augmented board games by high school youth. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 130–139).
82. Sullivan, A., & Bers, M. U. (2019). Computer science education in early childhood: the case of ScratchJr. *Journal of Information Technology Education. Innovations in Practice*, 18, 113-138.
83. Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From scratch to "real" programming. *ACM Transactions on Computing Education* (TOCE), 14(4), 1–15.
84. Weintrop, D., Killen, H., Munzar, T., & Franke, B. (2019, February). Block-based comprehension: Exploring and explaining student outcomes from a read-only block-based exam. In *Proceedings of the 50th ACM technical symposium on Computer Science Education* (pp. 1218–1224).
85. Franklin, D., Weintrop, D., Palmer, J., Coenraad, M., Cobian, M., Beck, K., ... & Crenshaw, Z. (2020, February). Scratch Encore: The design and pilot of a culturally-relevant intermediate Scratch curriculum. In *Proceedings of the 51st ACM technical symposium on computer science education* (pp. 794–800).
86. Paparo, G., Hartmann, M., & Grillenberger, M. (2021, November). A Scratch Challenge: Middle School Students Working with Variables, Lists and Procedures. In *21st Koli Calling International Conference on Computing Education Research* (pp. 1–10).
87. Wagner, A., Gray, J., Corley, J., & Wolber, D. (2013, March). Using App Inventor in a K-12 summer camp. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 621–626).
88. Kim, S. W., & Lee, Y. (2017). The Effects of Programming Education using App inventor on Problem-solving Ability and Self-efficacy, Perception. *Journal of the Korea Society of Computer and Information*, 22(1), 123-134.
89. Basu, S. (2019, February). Using Rubrics Integrating Design and Coding to Assess Middle School Students' Open-ended Block-based Programming Projects. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 1211–1217).
90. Sentance, S., & Schwiderski-Grosche, S. (2012, November). Challenge and creativity: using. NET gadgeteer in schools. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (pp. 90–100).
91. Kelleher, C., Pausch, R., & Kiesler, S. (2007, April). Storytelling Alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 1455–1464).
92. Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education,* 215–220.
93. Lewis, C. M. (2010, March). How programming environment shapes perception, learning and goals: logo vs. scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 346–350).
94. Weill-Tessier, P., Costache, A. L., & Brown, N. C. (2021, March). Usage of the Java Language by Novices over Time: Implications for Tool and Language Design. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 328–334).

95. Swidan, A., & Hermans, F. (2019, May). The effect of Reading code aloud on comprehension: An empirical study with school students. In *Proceedings of the ACM Conference on Global Computing Education* (pp. 178–184).
96. Staubitz, T., Teusner, R., & Meinel, C. (2019, April). MOOCs in secondary education-experiments and observations from German classrooms. In *2019 IEEE global engineering education conference* (EDUCON) (pp. 173-182). IEEE.
97. Guniš, J., Šnajder, L., Tkáčová, Z., & Gunišová, V. (2020). Inquiry-Based Python Programming at Secondary Schools. In *2020 43rd International Convention on Information, Communication and Electronic Technology* (MIPRO) (pp. 750–754).
98. Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education*, 142, 103646.
99. Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (*TOCE), 18(1), 1–25.
100. Falloon, G. W. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jr. on the iPad. *Journal of Computer Assisted Learning,* 32, 576–593.
101. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60–67
102. Chen, C., Haduong, P., Brennan, K., Sonnert, G., & Sadler, P. (2019). The effects of first programming language on college students' computing attitude and achievement: a comparison of graphical and textual languages. *Computer Science Education*, 29(1), 23-48.
103. Weintrop, D., & Wilensky, U. (2015b, June). To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children* (pp. 199–208).
104. Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, 23(3), 239-264.
105. Grover, S., Jackiw, N., & Lundh, P. (2019). Concepts before coding: Non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education*, 29(2-3), 106-135.
106. Benton, L., Saunders, P., Kalas, I., Hoyles, C., & Noss, R. (2018). Designing for learning mathematics through programming: A case study of pupils engaging with place value. *International journal of child-computer interaction*, 16, 68-76.
107. Shamir, M., Kocherovsky, M., & Chung, C. (2019, March). A paradigm for teaching math and computer science concepts in K-12 learning environment by integrating coding, animation, dance, music and art. In *2019 IEEE Integrated STEM Education Conference* (ISEC) (pp. 62–68).
108. Aivaloglou, E., & Hermans, F. (2019). Early programming education and career orientation: the effects of gender, self-efficacy, motivation and stereotypes. In *Proceedings of the 50th ACM technical symposium on computer science education* (pp. 679–685).
109. Moritz, S. H., Wei, F., Parvez, S. M., & Blank, G. D. (2005). From objects-first to design-first with multimedia and intelligent tutoring. *ACM SIGCSE Bulletin*, 37(3), 99-103.
110. Sentance, S., & Waite, J. (2017, November). PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 113–114).
111. Blanchard, J., Gardner-McCune, C., & Anthony, L. (2020, February). Dual-modality instruction and learning: A case study in CS1. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 818–824).
112. Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., & Lui, D. (2014a). A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles. *ACM Transactions on Computing Education* (TOCE), 14(1), 1–20.
113. Kafai, Y., Searle, K., Martinez, C., & Brayboy, B. (2014b, March). Ethnocomputing with electronic textiles: Culturally responsive open design to broaden participation in computing

in American Indian youth and communities. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 241–246).
114. Searle, K. A., & Kafai, Y. B. (2015, July). Boys' Needlework: Understanding Gendered and Indigenous Perspectives on Computing and Crafting with Electronic Textiles. In ICER (pp. 31–39).
115. Sentance, S., Waite, J., Hodges, S., MacLeod, E., & Yeomans, L. (2017, March). "Creating Cool Stuff" Pupils' Experience of the BBC micro: bit. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (pp. 531–536).
116. Sapounidis, T., Demetriadis, S., & Stamelos, I. (2015). Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing*, 19(1), 225-237.
117. Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education* (TOCE), 18(1), 1–20.
118. Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & education*, 52(1), 1-12.
119. Grover, S., Bienkowski, M., Tamrakar, A., Siddiquie, B., Salter, D., & Divakaran, A. (2016, April). Multimodal analytics to study collaborative problem solving in pair programming. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge* (pp. 516–517).
120. Musaeus, L. H., & Musaeus, P. (2019, February). Computational thinking in the Danish High School: Learning coding, modeling, and content knowledge with NetLogo. In *Proceedings of the 50th ACM technical symposium on computer science education* (pp. 913–919).
121. Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective. *Computer Science Education*, 29(2-3), 136-176.
122. Lytle, N., Milliken, A., Cateté, V., & Barnes, T. (2020). Investigating different assignment designs to promote collaboration in block-based environments. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 832–838).
123. Tsan, J., Vandenberg, J., Zakaria, Z., Wiggins, J. B., Webber, A. R., Bradbury, A., ... & Boyer, K. E. (2020, February). A comparison of two pair programming configurations for upper elementary students. In *Proceedings of the 51st ACM technical symposium on computer science education* (pp. 346–352).
124. Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Communications of the ACM*, 43(5), 108-114.
125. Tsan, J., Lynch, C. F., & Boyer, K. E. (2018). "Alright, what do we need?": A study of young coders' collaborative dialogue. *International Journal of Child-Computer Interaction*, 17, 61-71.
126. Campe, S., Denner, J., Green, E., & Torres, D. (2020). Pair programming in middle school: variations in interactions and behaviors. *Computer Science Education*, 30(1), 22-46.
127. Tsan, J., Boyer, K. E., & Lynch, C. F. (2016, February). How early does the CS gender gap emerge? A study of collaborative problem solving in 5th grade computer science. In *Proceedings of the 47th ACM technical symposium on computing science education* (pp. 388–393).
128. Goode, J., Margolis, J., & Chapman, G. (2014, March). Curriculum is not enough: The educational theory and research foundation of the exploring computer science professional development model. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 493–498).
129. Barron, B., & Darling-Hammond, L. (2010). Prospects and challenges for inquiry-based approaches to learning. *The nature of learning: Using research to inspire practice*, 199-225.
130. Barron, B., & Darling-Hammond, L. (2008). Teaching for Meaningful Learning: A Review of Research on Inquiry-Based and Cooperative Learning. *Book Excerpt*. George Lucas Educational Foundation.

131. Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., ... & Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education* (TOCE), 15(2), 1–31.
132. Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM technical symposium on computer science education* (pp. 245–250). New York, NY: ACM.
133. Werner, L., Denner, J., & Campe, S. (2014). Children programming games: A strategy for measuring computational learning. *ACM Transactions on Computing Education (*TOCE), 14(4), 1–22.
134. Tsarava, K., Moeller, K., Pinkwart, N., Butz, M., Trautwein, U., & Ninaus, M. (2017, October). Training computational thinking: Game-based unplugged and plugged-in activities in primary school. In *European conference on games based learning* (pp. 687–695).
135. Helle, L., Tynjälä, P., & Olkinuora, E. (2006). Project-based learning in post-secondary education – theory, practice and rubber sling shots. *Higher Education,* 51, 287-314.
136. Drain, M. (2010). Justification of the dual-phase project-based pedagogical approach in a primary school technology unit. *Design and Technology Education, 15*, 7–14.
137. Romeike, R., & Göttel, T. (2012, November). Agile projects in high school computing education: emphasizing a learners' perspective. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (pp. 48–57).
138. Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist,* 26, 369-398.
139. Kwon, K., Cheon, J., & Moon, H. (2021). Levels of problem-solving competency identified through Bebras Computing Challenge. *Education and Information Technologies*, 26(5), 5477-5498.
140. Gülbahar, Y., Ilkhan, M., Kilis, S., & Arslan, O. (2013). Informatics education in Turkey: National ICT curriculum and teacher training at elementary level. In *Informatics in Schools: Local Proceedings of the 6th International Conference* (ISSEP) (pp. 77–87).
141. Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems,* 75, 661-670.
142. Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 29(1), 12-28.
143. Alves, N. D. C., von Wangenheim, C. G., Hauck, J. C. R., & Borgatto, A. F. (2020, February). A large-scale evaluation of a rubric for the automatic assessment of algorithms and programming concepts. In *Proceedings of the 51st ACM technical symposium on computer science education* (pp. 556–562).
144. Öqvist, M., & Nouri, J. (2018). Coding by hand or on the computer? Evaluating the effect of assessment mode on performance of students learning programming. *Journal of Computers in Education*, 5(2), 199-219.
145. Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, 162, 104083.
146. Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers and Education*, 148, 1–22.
147. Kong, S. C. (2016). A framework of curriculum design for computational thinking development in K-12 education. *Journal of Computers in Education,* 3(4), 377-394.
148. Grover, S., Cooper, S., & Pea, R. (2014). Assessing computational learning in K-12. In *Proceedings of the 2014 conference on Innovation & Technology in Computer Science Education* (pp. 57–62). New York, NY: ACM.

149. Román-González, M., Moreno-León, J., & Robles, G. (2019). Combining assessment tools for a comprehensive evaluation of computational thinking interventions. In *Computational thinking education* (pp. 79-98). Springer, Singapore.
150. Djambong, T., Freiman, V., Gauvin, S., Paquet, M., & Chiasson, M. (2018). Measurement of computational thinking in K-12 education: The need for innovative practices. In *Digital technologies: Sustainable innovations for improving teaching and learning* (pp. 193–222).
151. Mühling, A., Ruf, A., & Hubwieser, P. (2015). Design and first results of a psychometric test for measuring basic programming abilities. In *Proceedings of the workshop in primary and secondary computing education* (pp. 2–10). New York, NY: ACM.
152. Weintrop, D., & Wilensky, U. (2015a, August). Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs. In *Proceedings of the eleventh annual international conference on international computing education research* (pp. 101–110).
153. Gunn, C., & Peddie, R. (2008). A design-based research approach for eportfolio initiatives. In *Hello! Where are you in the landscape of educational technology? Proceedings Ascilite Melbourne* (pp. 363–367).
154. Moreno-León, J., & Robles, G. (2015). Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills. In *Scratch conference* (pp. 12–15).
155. Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010, September). Towards the automatic recognition of computational thinking for adaptive visual language learning. In *2010 IEEE symposium on visual languages and human-centric computing* (pp. 59–66).
156. Izu, C., Mirolo, C., Settle, A., Mannila, L., & Stupuriene, G. (2017). Exploring Bebras Tasks Content and Performance: A Multinational Study. *Informatics in Education*, 16(1), 39-59.
157. Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior,* 72, 558-569.
158. Gülbahar, Y. Kert, S. B. & Kalelioğlu F. (2019). Self-efficacy perception scale for computational thinking skills: Validity and reliability study. *Turkish Journal of Computer and Mathematics Education,* 10(1), 1-29.
159. Grover, S. (2017). Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In *Emerging research, practice, and policy on computational thinking* (pp. 269-288). Springer, Cham.
160. Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association,*(pp. 1–25). Vancouver, Canada.
161. Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education,* 109, 162-175.
162. Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education* (TOCE), 14(1), 1–16.
163. Mouza, C., Sheridan, S., Lavigne, N. C., & Pollock, L. (2021). Preparing undergraduate students to support K-12 computer science teaching through school-university partnerships: reflections from the field. *Computer Science Education*, 1-26.
164. Hubbard, A., & D'Silva, K. (2018, August). Professional learning in the midst of teaching computer science. In *Proceedings of the 2018 ACM Conference on International computing education research* (pp. 86–94).
165. Jocius, R., Joshi, D., Dong, Y., Robinson, R., Catete, V., Barnes, T., Albert, J., Andrews, A., & Lytl, N. (2020). Code, connect, create: The 3c professional development model to support computational thinking infusion. In *Proceedings of the 51st ACM technical symposium on computer science education* (pp. 971–977).
166. Blum, L., & Cortina, T. J. (2007). CS4HS: an outreach program for high school CS teachers. *ACM SIGCSE Bulletin*, 39(1), 19-23.

167. Franklin, D., Conrad, P., Boe, B., Nilsen, K., Hill, C., Len, M., ... & Waite, R. (2013, March). Assessment of computer science learning in a scratch-based outreach program. In *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 371–376).
168. Margolis, J., Ryoo, J. J., Sandoval, C. D., Lee, C., Goode, J., & Chapman, G. (2012). Beyond access: Broadening participation in high school computer science. *ACM Inroads*, *3*(4), 72-78.
169. Goode, J. & Chapman, G. (2008). *Exploring computer science*. Computer Science Equity Alliance, Los Angeles, CA.
170. Adams, J. C. (2007). Alice, middle schoolers & the imaginary worlds camps. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education* (pp. 307–311).
171. Bryant, C., Chen, Y., Chen, Z., Gilmour, J., Gumidyala, S., Herce-Hagiwara, B., ... & Rebelsky, S. A. (2019, February). A middle-school camp emphasizing data science and computing for social good. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 358–364).
172. Nite, S. B., Bicer, A., Currens, K. C., & Tejani, R. (2020, October). Increasing STEM interest through coding with microcontrollers. In *2020 IEEE Frontiers in Education Conference* (FIE) (pp. 1–7).
173. Stupurienė, G., Juškevičienė, A., Jevsikova, T., Dagienė, V., & Meškauskienė, A. (2021, November). Girls' Summer School for Physical Computing: Methodology and Acceptance Issues. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (ISSEP) (pp. 95-108). Springer, Cham.
174. Kenderov, P. S. (2017). Three Decades of International Informatics Competitions: How did IOI Start. *Olympiads in Informatics*, 11, 3-10.
175. Verhoeff, T. (2009). 20 years of IOI competition tasks. *Olympiads in Informatics*, *3*, 149-166.
176. Kiryukhin V, Okulov S (2007) *Methods of problem solving in informatics: international olympiads*. LBZ (BINOM. Knowledge Lab). Moscow. (in Russian)
177. Delal, H., & Oner, D. (2020). Developing middle school students' computational thinking skills using unplugged computing activities. *Informatics in Education*, 19(1), 1-13.
178. Oliveira, A. L. S., Andrade, W. L., Guerrero, D. D. S., & Melo, M. R. A. (2021, October). How do Bebras Tasks Explore Algorithmic Thinking Skill in a Computational Thinking Contest?. In *2021 IEEE Frontiers in Education Conference* (FIE) (pp. 1–7).
179. Vaníček, J., Šimandl, V., & Klofač, P. (2021). A Comparison of Abstraction and Algorithmic Tasks Used in Bebras Challenge. *Informatics in Education,* 20(4), 717-736.
180. Budinská, L., & Mayerová, K. (2019, November). From Bebras tasks to lesson plans–graph data structures. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (ISSEP) (pp. 256-267). Springer, Cham.
181. Kalelioğlu, F., Doğan, D., & Gülbahar, Y. (2021). Snapshot of Computational Thinking in Turkey: A Critique of 2019 Bebras Challenge. *Informatics in Education*. doi:https://doi.org/10.15388/infedu.2022.19
182. Lonati, V. (2020). Getting inspired by bebras tasks. How Italian teachers elaborate on computing topics. *Informatics in Education*, 19(4), 669–699.
183. Combéfis, S., & Stupurienė, G. (2020, September). Bebras based activities for computer science education: review and perspectives. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (ISSEP) (pp. 15-29). Springer, Cham.
184. Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., & Torelli, M. (2015, June). How challenging are Bebras tasks? An IRT analysis based on the performance of Italian students. In *Proceedings of the 2015 ACM conference on innovation and technology in computer science education* (pp. 27–32).
185. Černochová, M., & Selcuk, H. (2021, August). Primary Education Student Teachers' Perceptions of Computational Thinking Through Bebras Tasks. In *Open Conference on Computers in Education* (ISSEP) (pp. 15-27). Springer, Cham.
186. Dagienė, V., Sentance, S., & Stupurienė, G. (2017). Developing a two-dimensional categorization system for educational tasks in informatics. *Informatica*, 28(1), 23-44.

187. Datzko, C. (2019, November). The genesis of a bebras task. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (ISSEP) (pp. 240-255). Springer, Cham.
188. Van der Vegt, W., & Schrijvers, E. (2019). Analyzing task difficulty in a Bebras contest using cuttle. *Olympiads in Informatics*, 13, 145-156.
189. Araujo, A. L. S. O., Santos, J. S., Andrade, W. L., Guerrero, D. D. S., & Dagienė, V. (2017, October). Exploring computational thinking assessment in introductory programming courses. In *2017 IEEE Frontiers in Education Conference* (FIE) (pp. 1–9).
190. Kwon, S. M., Wardrip, P. S., & Gomez, L. M. (2014). Co-design of interdisciplinary projects as a mechanism for school capacity growth. *Improving Schools,* 17, 54–71.
191. Cortina, T. J. (2015). Reaching a broader population of students through“ unplugged” activities. *Communications of the ACM*, *58*(3), 25-27.
192. Unnikrishnan, R., Amrita, N., Muir, A., & Rao, B. (2016). Of Elephants and Nested Loops: How to Introduce Computing to Youth in Rural India. *ACM Press*, 137-146.
193. Del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of Primary Education. *Computers & Education*, 150, 103832.

# Conceptualizing Approaches to Critical Computing Education: Inquiry, Design, and Reimagination


**Luis Morales-Navarro and Yasmin B. Kafai**

## 1 Introduction

As CS education has gained an unprecedented momentum, becoming part of the K-12 curriculum over the past 10 years and with increasing enrollment in higher education [27], several critical issues in computing as a discipline have become more visible: (1) algorithmic bias is pervasive, reinforcing historical inequities and damaging minoritized communities, (2) discrimination in industry against minoritized peoples in hiring and management practices continues, and (3) computing continues to be taught and learned as a value neutral subject [25]. Today we recognize that code is not neutral, computing technologies reflect the values and biases of their creators, and computing alone is not the solution to all problems, it often deepens existing ones, perpetuates existing concerns, or even causes new problems [11]. But in most classrooms, computing is commonly introduced and experienced as a "value-neutral tool independent from society" ([42], p.31) without considering its societal and ethical implications [64, 75]. Yet learning computing with disciplinary authenticity requires attending to its critical and political dimensions [53].

More recently, efforts to foreground criticality in computing education have addressed this lack of concern for societal and ethical implications and limitations of the discipline through numerous proposals, among them: critical computational empowerment [63], justice-centered efforts [11, 45, 64], critical computational literacy [44], critical computing literacy [57], responsible computing [48], computational action [62], critical algorithmic literacy [12], abolitionist computing [37], computational empowerment [14, 34], liberatory computing [72] and counter-

L. Morales-Navarro (✉) · Y. B. Kafai
University of Pennsylvania, Philadelphia, PA, USA
e-mail: luismn@upenn.edu; kafai@upenn.edu



M. Apiola et al. (eds.), *Past, Present and Future of Computing Education Research*,
https://doi.org/10.1007/978-3-031-25336-2_21

hegemonic computing [16]. While this proliferation of proposals highlights the growing importance given to critical issues in computing education, it is unclear what each approach means by criticality in computing, who is participating in these efforts, and how students and teachers are engaged in critical computing activities.

In this conceptual chapter, we take a step back from individual efforts by identifying common historical roots in human computer interaction (HCI) and in language arts and literacy (LAL) studies and how these perspectives foreshadowed today's critical computing education (CCE) initiatives. We then identify and describe how three emergent approaches—(1) inquiry, (2) design, and (3) reimagination—address criticality in computing education. Finally, we discuss how these approaches highlight issues to be addressed and provide directions for future learning research and design.

## 2 Historicizing Criticality

Recent calls for considering the political dimensions of learning in design and research [5, 43] have been followed by efforts in critical science literacy [56], critical history and social studies [32, 35], critical data literacies [33, 61], and also in computer science education [38, 42]. Yet critical perspectives in computing and how computing is learned and taught have been present from the early days. For instance, Weizenbaum [73] argued that the technical innovations of computing did not necessarily promote social progress. Most importantly, he distinguished the differences in human and machine decision making, noting that computers lacked compassion in their calculations. From the education side, Papert [52] pointed towards the fallacy of seeing the computer (or software) as the agent of change in student learning, considering people and culture as the driving forces of learning. Further and deeper discussions about criticality in computing and education have historically developed in HCI around the Aarhus conferences community [3] and around the critical literacies movement [9]. These two perspectives combined can be helpful to situate and historicize current efforts in CCE and better understand the development of different pedagogical approaches.

In language arts and literacy (LAL) studies, critical literacies are seen as ways of being and doing through which learners participate in the world. Central here is Freire's work on critical consciousness and literacy. Freire [22] proposed an emancipatory model of literacy based on a dialectical relationship between humans and the world, where literacy is not seen as a collection of skills but rather a precondition for freedom, participation in the world and social empowerment within a wider project of social and political reconstruction. These ideas were further developed in the New London Group meetings where the multimodality of literacies and pedagogy were considered to propose that literacies must have a critical framing for learners to grow in their practice while consciously engaging with historical, social, and political contexts [10]. Today, critical literacies bring together ideas from marxist, queer, feminist, postcolonial, and critical race theories to inquire on the

dynamics of power present in the world and the learning process. In a recent review, Vasquez and colleagues [68] argue that such a framework does not always involve taking a negative stance, but rather looking at issues from different perspectives, analyzing, suggesting and creating possibilities for change and improvement. The vast literature in critical literacies can be helpful in bringing CCE to classrooms. In fact, thinking about computing by applying frameworks from literacies is not new. The relationship between reading and writing code as extensions of literacies has been extensively discussed [15, 69] by emphasizing how learning computing is necessary to fully participate in the world.

In HCI the term "critical computing" has a long, and often overlooked, history that dates back to 1975 when the first Aarhus conference convened to discuss the development of computing systems in context. These conversations at Aarhus emerged through dialogues on how computing could support workers with a particular interest in marxist approaches for understanding the design and use of technology in relation to class and power struggles [50]. Over time, these discussions have evolved, expanding critical computing to "critical action, not only as workplace actionism, but also by integrating a broader scope of critical analysis and critical practice" (p. 3)—grounded in political, economic, and aesthetic theories—in how computing systems are designed and used in the workplace, education, and at home [3]. The contributions of the Aarhus community and their perspectives on computing challenge many of the everyday practices of computing professionals and the ways we introduce learners to the field. For instance, within the Aarhus community, Burstall [6] reflected that computing education that prioritizes technical concepts at the expense of critical awareness of other people and the environment loses the ability to accomplish the goals of the field.

## *2.1 Empowerment*

Even though these critical perspectives in computing and education are products of different communities, contexts and conversations, taken together they provide useful underpinnings for CCE. Indeed the agency of learners, and the analysis as well as production of texts and code have been conceptualized as foundations for critical engagement in both traditions. To begin with, both HCI and LAL emphasize the need for empowerment or agency of learners. For instance, in Freire and Macedo's [24] emancipatory model of literacy, literacy is a precondition for empowerment, that is being able to fully participate in the world. Recently, critical literacies have been framed as a way of being and doing [51], extending literacies beyond an orientation for teaching and learning. This with the goal that learners can analyze and interrogate the micro features of texts and their macro—institutional, political, societal—conditions while focusing on how relations of power work, design and produce texts beyond classroom assignment [68]. The goal of literacy here is also one of empowerment, for learners to critically engage with the world to design social futures [9]. From the HCI perspective, Aarhus' "critical computing"

proposes that computing is a process of reality construction and transformation of the world [19]. This opens possibilities to reframe the development of computing applications from designing for requirements to designing for the opportunities of creating better, more just and equitable worlds [20]. In this sense, novice learners must be empowered to use their curiosity and creativity to critically interrogate the history, implications and limitations of computing and create for the possibility of more equitable futures. How do these perspectives envision promoting agency or empowerment of learners? Two approaches stand out: one being reading or analysis, the other being writing or production of "text" or "code".

### 2.1.1 Analysis

In both LAL and HCI analysis plays an important role in critical empowerment. For instance, in literacies, critical reading engages learners in reading beyond the words of the page. Indeed, for Freire "reading the world precedes reading the word" [24] as learners can use critical literacy to make sense of their everyday lived experiences. Critical reading involves deconstructing texts (e.g., media, discourse, technologies) to question how these are constructed [68]. This kind of reading generates opportunities for "unpacking myths and distortions and building new ways of knowing and acting upon the world" ([47], p. 22) by consciously investigating relationships of power, ideologies, and values in the process of reading [9]. As learners critique texts, they deconstruct and reconstruct them creating opportunities to disrupt, examine and sometimes dismantle problematic practices as well as to imagine alternatives, hypothesize how to change things and even take action [68]. Similarly, in the Aarhus community technological criticism also plays an important role for understanding the impact of computing and how computers are used. Here, reflection on the unconscious values embedded in computing applications and practices is central as well as investigating the ways in which technologies perpetuate oppression [58].

### 2.1.2 Production

To promote agency, writing or production are important. The New London Group [9], in reframing literacy studies, foregrounds the key role of designing new texts and redesigning existing texts for learners to participate in designing the future. This process of production requires that learners understand the positions from which they design so that they too read their own creations critically. This perspective resonates with Freire's [23] idea that "reading the word is not only preceded by reading the world, but also by a certain form of writing it or rewriting" (p. 18). In this context, critical literacies are not only critical reading but also critical writing where writing means designing and redesigning texts in ways that aspire towards justice and can be socially transformative.

Likewise, Christiane Floyd [21], who together with Kristen Nygaard played a key role in shaping the critical computing agenda within the Aarhus conferences, argues that in critical computing, understanding the impact of computing and how computers are used is not enough. She claims that critical computing is intertwined with the development of computing applications and the goal that these applications should have a positive impact in society. Furthermore, Floyd [21] argues that there is a difference between being critical *in* computing and being critical *on* computing. Being critical *in* computing is concerned with design and production of critical computing applications. This is distinct from being critical *on* computing, which centers technological criticism over the design of applications that can address critical issues. CCE has the potential to engage learners in being critical *on* computing through the analysis of computing applications discussed in the previous section. Yet, learners can also be critical *in* computing by deeply engaging with the social and political implications of computing through the production of applications [21]. From this perspective, social and technical factors are intertwined [1] and the idea of computer programming must be widened to take into account the social context of software use and its development [49]. Programming is framed as the means for non-technical ends or goals [1] which requires investigating how the decisions made when coding have social and environmental implications [50].

## 3 Approaches to Critical Computing Education

Reviewing the historical roots of criticality in both computing and education provides a backdrop to examine how current research promotes CCE. From this review of HCI and LAL it emerges that CCE must empower learners to critically analyze computing, examine power and values, and interrogate the implications and limitations of computing applications. Likewise, CCE should include creating computing applications that aspire towards justice and that reflectively consider the limitations and implications of computing for people and the environment. In this section, we propose that out of these two traditions, three different pedagogical operationalizations or framings CCE emerge: (1) critical inquiry, (2) critical design, and (3) critical reimagination. Each of these approaches promotes the empowerment of learners through analysis (reading), production (writing), or combinations thereof (see Fig. 1). Whereas our conceptualization of the inquiry approach stems out of the analysis (reading) tradition of critical literacy and the idea of being "critical on computing" from the Aarhus conferences, the design approach includes analysis as well as production or being "critical in computing." Reimagination involves both analysis and production with the goal of envisioning more equitable speculative futures for computing. While these approaches are distinct, they are not mutually exclusive and at times may even be complimentary; in practice many CCE efforts draw on more than one approach.

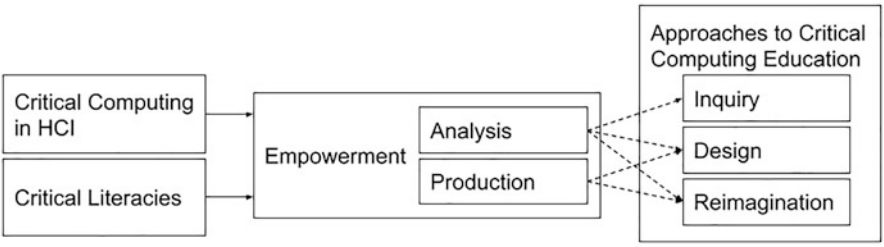


**Fig. 1** Relationships between historical underpinnings of criticality and approaches to critical computing education

## *3.1 Critical Inquiry*

One approach to CCE, which we call "critical inquiry" involves students inquiring on the implications of computing. For instance, in a social design experiment conducted by Vakil [33], teenagers examined surveillance technologies in their communities and created infographics to explain the implications and limitations of these technologies. Through this experience, students explored the ethical tensions of how the surveillance technologies they encounter and use in their everyday lives not just reproduce but may amplify injustices in new ways, disproportionately targeting and impacting minoritized communities. In a discussing another iteration of the design experiment, Vakil et al. [66] highlight how this intervention was designed to "foreground learning how to decode and unmake tech's relationship with power through artistic, moral and humanistic inquiry" (p. 2) by positioning youth as philosophers of technology. Here youth, through the production of a documentary film, examined the ways in which computing technologies are used to surveil immigrants and also how immigrant communities resist surveillance through the use of technology. In a similar fashion, Vogel [70] discusses how bilingual secondary school students critiqued the educational technologies used at school by inquiring into their embedded values and particularly their raciolinguistic ideologies.

Another example of critical inquiry is a course, co-constructed by Everson and colleagues with secondary students [17], in which learners explored inequities in access to computing education by creating data visualizations, investigated bias in machine learning, data privacy practices and their ethical implications. For instance, students researched how bias in machine learning models in everyday applications can have negative impacts in the lives of users and asked questions such as "What can we do?," demonstrating a desire for change. Similarly, Walker and colleagues [72] propose a series of activities that can engage Black secondary students with what they call liberatory computing. Among these, they suggest students investigate data practices by analyzing how computing artifacts treat data and how youth produce data themselves and conduct "evocative audits" to research the positive and negative ways in which computing systems affect the socio-political realities of their communities.

Other efforts, at an undergraduate level, center on incorporating critical inquiry in introductory courses in which learners first encounter foundational technical concepts [18]. Lin [45], for example, argues that affordance analysis can be used in higher education to examine how algorithms and abstractions such as data structures have political implications. For instance, students could discuss how the use of binary trees in hiring decisions, autocomplete for search engines, and shortest paths for navigation systems can have problematic unintended consequences that reproduce systemic injustice. Furthermore, Lin [46] proposes Critical Comparative Data Structures and Algorithms as a justice-centered approach to teaching and learning in undergraduate computer science. This approach centers on critique of the values embedded in data structures and algorithm design and solutionist approaches to computing by constantly comparing dominant approaches to design justice approaches [11]. The goal here is for undergraduate students to engage beyond learning programming to understanding computing as a socio-technical field where the decisions they make while programming have implications. Lin highlights the importance of doing this work in a learner-centered environment where students can have agency to propose topics and examples for discussion that are relevant to their interests and lived experiences. Kirdani-Ryan and Ko [39] provide another example of critical inquiry at an undergraduate level describing how they embedded discussions and assignments about ethics and justice in a low level software (computer systems) introductory course. In this course they framed computing as an object of critique and engaged students in investigating dominant and counter narratives in the discipline.

Whereas the examples discussed above incorporate critical inquiry throughout semester-long undergraduate courses, Horton and colleagues [31] show that short isolated modules can also be engaging and informative for students. In their study, the inclusion of two 50-min long modules during which students engaged in critical conversations around data privacy and ethics in introductory computing course (CS102) increased learner's interest in ethical issues and self-efficacy in dealing with ethical issues. Having students be critical on computing in the same courses where they are learning technical concepts and skills can situate conversations and applications of technical concepts in their social and historical contexts.

The "critical inquiry" approach centers on critique—that is looking at issues from multiple perspectives, analyzing dynamics of power, and suggesting possibilities for change [68], drawing on the tradition of LAL. This approach can be productive when integrated across introductory computing curricula in order to have learners consider how the decisions made while programming have social and environmental implications. That way, as novices learn technical concepts and programming skills these can be grounded in critical socio-technical conversations on computing [50]. When engaging in critique, learning activities can connect to the lived experiences of students, like in Vakil's experiment, by having them investigate the implications and limitations of the technologies they use or even further by critiquing their own creations. Critique can be empowering, giving students the rightful presence—that is legitimized membership and participation, where the community works towards justice by making injustice and social change visible [7]—to interrogate systems of

oppression and question what computing is and how computing is enacted in the world. At the same time, learners can have agency to decide what issues they want to investigate. Yet CCE has the potential to go beyond inquiry by engaging learners in the production of critical computing applications.

### *3.2 Critical Design*

A second approach to CCE, which we call "critical design", centers on designing and redesigning with computing in ways that aspire towards justice and change. In this approach, learners critically design applications that address the needs of communities. For instance, Lee and Soep [44] argue that when young people design applications to address their own community problems they create opportunities for themselves and their communities, and that "only through production of these digital tools will youth develop the agency required to make the changes they want to see" (p. 481). Through their experiences at Youth Radio they have documented how teenagers create mobile and web applications for their communities, including, for example, an app called "Know Your Queer Rights" that provided resources to support LGBTQ+ youth [60] or an app to track and raise awareness of gentrification [44]. These examples show how young people can take action to address inequities in their communities and design with computing.

Other efforts in critical design have students design for communities other than their own. Bar-El and Worsley [2] discuss a university course for undergraduate and graduate students centered around accessibility. The course was designed to encourage students to explore computing as a field that promotes democratization by having learners design tools and activities to broaden participation in CS. this intervention, students were embedded in the spaces for which they designed and were in dialogue with community members. The authors argue that by creating real-life projects for real communities, learners can benefit from interrogating the history and assumptions of computing and its applications [2]. Since the majority of participants in the course were engineering students, most already had a background in computing which was helpful for students to develop complex projects that included, for example, browser extensions to audit web accessibility or multi-modal interfaces for design tools to support people who are blind in computer aided design.

Tissenbaum and colleagues [63] propose that empowerment comes from creating interventions in the real and physical world in which learners analyze the problems of their communities and create real solutions. For them, learners must have opportunities to create applications that have an impact on their communities from the moment they begin to learn how to code [62]. This contrasts with common approaches to computing education that prioritize learning technical skills first and working on real-world applications later. In this line, Van Wart and colleagues [67] present a case study of two high school students participating in a 12-week after-school program where students investigated community problems, identified local needs and designed mobile applications to address these needs. They argue that in

this "practice-oriented computer science learning environment" students accessed disciplinary concepts and skills and demonstrated their competency *while* working on projects that addressed issues of justice relevant to their communities. Here learners had agency to learn the concepts that were most relevant to their goals in an environment that recognized them as local experts using computing as a tool to address real world issues.

Learners can also engage with critical design by making small classroom projects that address critical issues related to the technologies they use everyday. Vogel [70], for instance, presents a compelling case study in a secondary school classroom where bilingual youth worked on designing voice-based interfaces that could understand their translanguaging practices after their teacher introduced voice recognition functions in a programming environment.

While engaging in critical design, learners have the opportunity to participate in critiquing and producing computing applications. Critical design aligns well with the ideas from the Aarhus conference particularly seeing programming as a social activity where writing code is the means for non-technical ends or goals [1]. It also addresses coding, in a similar fashion to how critical literacies theorize writing, as the process in which learners design and redesign in ways that aspire towards justice while reading their own creations critically [68]. Critical design is also an opportunity to design computing applications beyond requirements, thinking about the opportunities [20] that computing gives creators and users to make more equitable worlds and to inquire into the sociopolitical values and purposes of making with computing technologies [71]. Furthermore, analyzing problems in their communities and designing computing applications to address these problems can support the development of learners' rightful presence, empowering students to be and do critical computing in authentic and meaningful ways. But empowering students also means giving them the creative power and freedom to pursue their own interests and projects. These design activities can connect to students' lives beyond deficit views of their experiences [37] recognizing how the cultural wealth of their communities can have a positive impact in the learning environment and the computing applications learners create. This requires thinking about cultural context beyond making superficial connections to computing concepts so that learners can participate in their cultures while computing, draw from their own cultures and even remake them by investigating authentic problems and concerns they can engage with in their designs [4, 30].

### *3.3 Critical Reimagination*

A third approach which we call "critical reimagination" involves rethinking the present and the past to critically reimagine computing to create more equitable and just futures. Holbert and colleagues [28], for instance, created the "Remixing Wakanda" project in which Black female teenagers used their personal stories to design speculative artifacts that address social and environmental injustice. Bringing

together Afrofuturist and constructionist ideas, participants imagined and created possible futures where their communities thrived and where technology helped create a more equitable and sustainable world. They reimagined how technology could be used proactively to improve their communities. They created speculative computing projects such as a cloak that expressed information about the health, wellness, mood and identity of its wearer or a trash can that converted waste into energy. Other efforts have focused on having students reimagine computing cultures that are equitable where female, Black and Brown youth can fully participate. Efforts employing restorying [59] build on Black feminist perspectives for youth to imagine and build the worlds and computing cultures they want to live and participate in. Shaw and colleagues [59] describe how youth engaged in interrogating dominant narratives about CS and crafted computational artifacts that reimagined CS, its values, who can participate and how it is done. In contrast to the "Remixing Wakanda" effort, which drew on the futuristic visions promoted in a popular movie, the restorying effort drew on the often forgotten historical connections between computing and textile work from Jacquard's loom as a predecessor to modern computing to the use of quilts by Black women to address social issues.

Speculating about the future of computing at a university level can often involve engaging in "Black Mirror" exercises, drawing on a popular science fiction TV series to inquire into the possible ethical dilemmas and social impacts of computing applications [40]. These are common in undergraduate computing courses on ethics and society. While here learners engage in being critical about computing and imagining futures, these are often dystopian. Yet, as Klassen and Fiesler [40] suggest these activities have the potential to also be suitable to imagine more ethical and just futures for computing.

Critical reimagination engages learners with the social and political reconstruction discussed by critical literacy scholars such as Giroux [26] that requires critique, production and developing an empowered voice to imagine and create alternatives for liberation and justice. As hooks [29] argues, imagination is emancipatory because thinking of possible futures involves critical thinking, analysis, and reflection of the present as well as a desire to build better worlds. In this approach when learners engage in production they design beyond requirements for the opportunity of creating better worlds. Learners address critical computing as emancipation [13] and through their speculative computational artifacts transform the world by constructing and proposing new realities [19]. They are empowered to reimagine what can be done with computing and what computing can be, having the rightful presence to question and rethink the discipline. Yet in critical reimagination activities it is important to provide space for student agency so that learners develop their own voices while accomplishing their desired computational goals. Reimagination connects the past, present and future, understanding that technology is not created in a void but builds on previous traditions. With critical reimagination the future is not pre-established but rather learners engage in what Freire [22] calls hopeful "revolutionary futurity," looking at the past to understand themselves and addressing how the present must change in order to imagine and build the future.

## 4 Considerations for the Learning Design and Research

By historicizing how criticality has been addressed in computing and education we situate CCE in both fields. The three approaches we discussed as framings for learning and instruction emerged from our analysis of efforts to design and research criticality in computing. From our analysis, we see that some of these current efforts in CCE align with the three proposed framings while others engage with more than one (see Table 1). For instance, some justice efforts [45] and liberatory computing [72] align with critical inquiry while critical computational empowerment [63], critical computational literacy [44] and computational action [62] prioritize critical design. Abolitionist computing [37] and counter-hegemonic computing [16], tend to align better with critical inquiry and reimagination. Other justice-centered efforts [64], responsible computing [48], critical computing literacy [57], and critical algorithmic literacy [12] engage both critical inquiry and design. Computational empowerment [14, 34] and justice-centered efforts such as Costanza-Chock's design justice [11] address critical inquiry, design and reimagination.

In identifying the different directions criticality assumes in each of the framings, we also noticed several issues that require special attention in designing and researching learning tools, activities, and environments: (1) who is involved in addressing critical issues in computing, (2) how to avoid the pitfalls of techno-solutionism, (3) how learners engage creatively with criticality, (4) how do we connect learning disciplinary skills and concepts with criticality, and (5) how to support teachers in bringing CCE to their classrooms?

First, criticality in computing should be addressed by all. In reviewing the various efforts in critical inquiry, design and reimagination we noticed that these involved mostly students from historically excluded communities. While this was probably done with the best intentions, to broaden participation, we should also be aware that such efforts could put a double "burden" on learners of minoritized communities [37], first as those who predominantly experience marginalization and discrimination when using technologies and participating in computing, and then as those tasked to identify challenges, develop new solutions, fix problems and reimagine alternate futures. This is important work but it is essential that all learners are engaged in CCE.

Second, while having students create computing applications that address real world problems is beneficial, it is equally important to consider what already exists in the communities that learners engage with, to encourage them to be critical towards their own work and avoid the pitfalls of techno-chauvinism and techno-solutionism—the beliefs that technology is always the solution [11]. Indeed, the critical design and reimagination approaches must acknowledge the limitations of computational solutions, questioning the possible implications of using computing to "fix the world."

Furthermore, we must consider how learners engage creatively with criticality. In inquiry, design and reimagination learners can connect to their personal interests and their lived experiences, to think, create and share their ideas with their peers.

**Table 1** Different CCE efforts and their engagement with criticality

| CCE effort | Engagement with criticality |
|---|---|
| **Abolitionist Computing** [37]<br>Integrating an abolitionist framework to CS to open up world-building possibilities that affirm Black Life. | *Inquiry:* Examining anti-Blackness in CS and CS education.<br>*Reimagination*: Reimagining CS through Black Life-affirming world-building projects. |
| **Computational Action** [62]<br>Engaging youth to take action with computing by making applications that have an impact in their communities. | *Design:* Students research problems in their communities and learn computing by creating applications to address these problems. |
| **Computational Empowerment** [14, 34]<br>Engaging students in understanding computing technologies and their effects on their lives and society by critically constructing and deconstructing computing artifacts. | *Inquiry:* Students critique and assess everyday technologies by considering their impact and implications.<br>*Design:* Students create computing applications that address problem situations.<br>*Reimagination:* Students co-create the future of computing by critically decoding and coding artifacts. |
| **Counter-hegemonic Computing** [16]<br>Engaging with Black students' counter-hegemonic practices in computing. | *Inquiry:* Students examine negative and positive frames of reference in computing by considering power and identity at societal and individual levels.<br>*Reimagination:* Using computing for emancipating counter-hegemonic practices. |
| **Critical Algorithmic Literacies** [12]<br>Enabling youth to critique and understand the algorithmic systems they encounter everyday. | *Inquiry:* Children analyze data sets related to the world in which they live.<br>*Design:* Children create and experiment with algorithms within "sandboxes for dangerous ideas." |
| **Critical Computational Empowerment** [63]<br>Engaging young learners in creating personally meaningful applications that have impact in the real world | *Design:* Students research problems in their communities and learn computing by creating applications to address these problems. |
| **Critical Computational Literacy** [44]<br>Engaging young people in creating projects that address inequities and injustice while learning to code. | *Design:* Students research problems in their communities and learn computing by creating applications to address these problems. |
| **Critical Computing Literacy** [57]<br>Centering doing and being with computing to broaden participation of girls in the discipline. | *Design:* Youth create applications to address needs in their communities. |
| **Design Justice** [11]<br>Remaining a community-centered alternative to computing where marginalized communities inquire on the implications of technology to explicitly challenge structural inequities through design. | *Inquiry:* Communities research the implications of computing applications to propose alternatives that foster justice.<br>*Design:* Communities design computing applications and tools to create a more equitable and sustainable world.<br>*Reimagination:* Transform computing through a community-centered design approach that aspires towards collective liberation and ecological sustainability. |

(continued)

**Table 1** (continued)

| CCE effort | Engagement with criticality |
|---|---|
| **Justice-centered Approaches to CS** [46] Constantly comparing dominant CS approaches to design justice approaches particularly with regards to the values embedded in data structures and algorithms. | *Inquiry:* Engage learners in investigating the values and implications of technical decisions made while programming. |
| **Justice-centered Computing** [64] Interrogating the sociopolitical context of CS education through critical inquiry into the curriculum, design of learning environments and purpose of CS education. | *Inquiry:* Students research ethics, power and politics of computing technologies they use everyday. *Design:* Students learn computing with a purpose, investigating community problems and designing applications to address them. |
| **Liberatory Computing** [72] Computing curriculum to motivate and prepare Black students to address racism embedded in society. | *Inquiry:* Students research issues of racism and inequity in computing with the goal of becoming data activists. |
| **Responsible Computing** [48] Approaching computing as a socio-technical field while students design artifacts. | *Inquiry:* Students discuss issues of justice, implications and consequences of computing. *Design:* Students design computing artifacts that take into account societal implications. |

To do this, learners' agency must be authentic, giving space for students to decide what issues to investigate and what kind of artifacts they want to create. This with the goal that they can engage in what Freire [22] calls the "creative transformation" of understanding, coding and participating in the world. Ultimately, while we want learners to "have the opportunity to experience the full conceptual and expressive powers of coding" ([54], p. 121), we also see a need to connect criticality back to creativity and curiosity rather than to consider creativity as a distinct engagement with code. Several of the examples we presented in different framings of CCE provide compelling illustrations how creative engagement can be coupled with critical inquiry or design. Here, involving learners in participatory design activities [14] is key to create learning experiences that promote creative engagement with criticality and build on learners' passions and lived experiences.

We should also consider how we join criticality with efforts for developing technical coding skills rather than seeing them as separate enterprises. In K-12, we take note that efforts in CCE predominantly take place in out-of-school settings. This is partly because current curricular frameworks and standards are very much focused on teaching and learning of computational skills and concepts, often relegating criticality to the sidelines. Yet alternatives are possible. In Denmark, for instance, the national K-12 computing curriculum framework [8] inherited many of the ideas of the Aarhus conferences [14, 34] giving computational empowerment the same importance as computational thinking. Here empowerment is addressed as the concern for learners to understand how computing affects their lives and society to creatively and critically participate in the construction of computing applications. On the other hand, recent efforts in higher education [18, 39, 46] push for the

integration of critical inquiry in introductory courses. There is plenty of potential to further critical design and reimagination in higher education, particularly in non-introductory courses, where students' existing technical knowledge can lead to novel applications as illustrated in the work of Bar-El and Worsley [2].

Finally, supporting teachers to integrate CCE in their classrooms is crucial. Most of the examples discussed in this chapter are from researcher-led small interventions. Yet, scaling CCE education may be challenging as teachers are often trained to Teach computing from a technical perspective only and as a value neutral subject [42]. While recent efforts, such as the publication of a text-book for secondary teachers [41], a graphic novel that address issues of critical computing [55], or a site with crowd-sourced critical coding activities [74] may support teachers in learning about and engaging their students with CCE, it is also crucial to partner with educators to co-design professional development experiences. Researchers and teachers should also partner in co-designing and redesigning classroom activities and curriculum to engage with criticality. For instance, Jayathirtha and colleagues [36] illustrate how existing coding activities can be re-configured so that students consider implications and limitations in human-computer interaction.

## 5 Conclusions

Our goal in this conceptual chapter is not to pit approaches against one another but to recognize that inquiry, design and reimagination each offer valuable ways to create learning activities that can engage with criticality in computing education research. Indeed, as Vakil and Higgs [65] write, the goal is that learners "can understand, analyze, critique, and reimagine the technologies that shape everyday lives" (p.31). The approaches, in fact, offer partial but complimentary ways for addressing criticality. Inquiry focuses on understanding the underpinning of technologies and power structures. Design promotes understanding criticality while making computing applications, creating a relevant context for learning technical skills and concepts, and reimagination centers on rethinking the present and the past to critically reimagine computing for the future.

In conceptualizing the proposed operationalizations of CCE, we are reminded that being critical involves both understanding the role of computing in society and of society in computing [38]. This requires not only discussing critical issues but deeply inquiring, designing and reimagining how computing can be transformed to be more just and equitable and to positively impact the lives of communities. Ultimately, as we saw in our historization of CCE, the goal is to empower learners to fully participate in computing through critique and production. We hope that our conceptualization of three emerging approaches contributes to advance our understanding of criticality in computing education.

**Acknowledgments** An earlier version of this chapter titled "Conceptualizing three approaches for integrating criticality in K-12 computing education" was presented and published in the proceedings of the International Conference of the Learning Sciences (ICLS) 2022.

## References

1. Andelfinger, U.: On the intertwining of social and technical factors in software development projects. In: Social Thinking-Software Practice, pp. 185–203 (2002)
2. Bar-El, D., Worsley, M.: Making the maker movement more inclusive: Lessons learned from a course on accessibility in making. International Journal of Child-Computer Interaction **29**, 100285 (2021)
3. Bertelsen, O., Bouvin, N., Krogh, P., Kyng, M.: Critical computing: Between sense and sensibility. In: Proceedings of the fourth decennial Aarhus Conference, pp. 20–24 (2005)
4. Blikstein, P.: Travels in troy with freire: Technology as an agent of emancipation. In: Social Justice Education for Teachers, pp. 205–235. Brill Sense (2008)
5. Booker, A.N., Vossoughi, S., Hooper, P.K.: Tensions and possibilities for political work in the learning sciences. In: Proceedings of the International Conference of the Learning Sciences. ISLS (2014)
6. Burstall, R.M.: Computing: yet another reality construction. In: Software development and reality construction, pp. 45–51. Springer (1992)
7. Calabrese Barton, A., Tan, E.: Designing for rightful presence in stem: The role of making present practices. Journal of the Learning Sciences **28**(4–5), 616–658 (2019)
8. Caspersen, M.E.: Informatics as a fundamental discipline in general education: The danish perspective. Perspectives on Digital Humanism p. 191 (2022)
9. Cazden, C., Cope, B., Fairclough, N., Gee, J., Kalantzis, M., Kress, G., Luke, A., Luke, C., Michaels, S., Nakata, M.: A pedagogy of multiliteracies: Designing social futures. Harvard educational review **66**(1), 60–92 (1996)
10. Cope, B., Kalantzis, M.: Introduction: Multiliteracies: The beginnings of an idea. In: Multiliteracies: Literacy learning and the design of social futures, pp. 3–8. Taylor and Francis (1999)
11. Costanza-Chock, S.: Design justice: Community-led practices to build the worlds we need. The MIT Press (2020)
12. Dasgupta, S., Hill, B.M.: Designing for critical algorithmic literacies. In: Algorithmic Rights and Protections for Children. (2021). https://doi.org/10.1162/ba67f642.646d0673
13. Dearden, A., Walker, S., Watts, L.: Choosing friends carefully: allies for critical computing. In: Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility, pp. 133–136 (2005)
14. Dindler, C., Smith, R., Iversen, O.S.: Computational empowerment: participatory design in education. CoDesign **16**(1), 66–80 (2020)
15. DiSessa, A.A.: Changing minds: Computers, learning, and literacy. Mit Press (2001)
16. Eglash, R., Bennett, A., Cooke, L., Babbitt, W., Lachney, M.: Counter-hegemonic computing: Toward computer science education for value generation and emancipation. ACM Transactions on Computing Education (TOCE) **21**(4), 1–30 (2021)
17. Everson, J., Kivuva, F.M., Ko, A.J.: "a key to reducing inequities in like, ai, is by reducing inequities everywhere first" emerging critical consciousness in a co-constructed secondary cs classroom. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, pp. 209–215 (2022)
18. Fisler, K., Friedler, S., Lin, K., Venkatasubramanian, S.: Approaches for weaving responsible computing into data structures and algorithms courses. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2, pp. 1049–1050 (2022)

19. Floyd, C.: Software development as reality construction. In: Software development and reality construction, pp. 86–100. Springer (1992)
20. Floyd, C.: Developing and embedding autooperational. Social Thinking–software Practice p. 5 (2002)
21. Floyd, C.: Being critical in, on or around computing? In: Proceedings of the 4th Decennial Conference on Critical Computing: Between Sense and Sensibility, CC '05, p. 207–211. Association for Computing Machinery, New York, NY, USA (2005). DOI https://doi.org/10.1145/1094562.1094601
22. Freire, P.: Pedagogy of the oppressed. Bloomsbury (1970)
23. Freire, P.: Reading the world and reading the word: An interview with paulo freire. Language arts **62**(1), 15–21 (1985)
24. Freire, P., Macedo, D.: Literacy: Reading the word and the world. Routledge (1987)
25. Gebru, T.: Race and gender. The Oxford handbook of ethics of AI pp. 251–269 (2020)
26. Giroux, H.A.: Literacy and the pedagogy of voice and political empowerment. Educational theory **38**(1), 61–75 (1988)
27. Halvorson, M.J.: Code Nation: Personal computing and the learn to program movement in America. ACM (2020)
28. Holbert, N., Dando, M., Correa, I.: Afrofuturism as critical constructionist design: building futures from the past and present. Learning, Media and Technology **45**(4), 328–344 (2020)
29. hooks, b.: Theory as liberatory practice. Yale JL & Feminism **4**, 1 (1991)
30. Hooper, P.K.: Looking BK and moving FD: Toward a sociocultural lens on learning with programmable media. MacArthur Foundation Digital Media and Learning Initiative (2008)
31. Horton, D., McIlraith, S.A., Wang, N., Majedi, M., McClure, E., Wald, B.: Embedding ethics in computer science courses: Does it work? In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, pp. 481–487 (2022)
32. Hostetler, A., Sengupta, P., Hollett, T.: Unsilencing critical conversations in social-studies teacher education using agent-based modeling. Cognition and Instruction **36**(2), 139–170 (2018)
33. Irgens, G.A., Simon, K., Wise, A., Philip, T., Olivares, M.C., Van Wart, S., Vakil, S., Marshall, J., Parikh, T.S., Lopez, M.L., et al.: Data literacies and social justice: Exploring critical data literacies through sociocultural perspectives. In: Proceedings of the International Conference of the Learning Sciences. ISLS (2020)
34. Iversen, O.S., Smith, R.C., Dindler, C.: From computational thinking to computational empowerment: a 21st century pd agenda. In: Proceedings of the 15th Participatory Design Conference: Full Papers-Volume 1, pp. 1–11 (2018)
35. Jackson, A.: The impact of critical history practices on history learning. In: Proceedings of the International Conference of the Learning Sciences. ISLS (2020)
36. Jayathirtha, G., Chipps, J., Morales-Navarro, L.: Redesigning an electronic textiles computer science activity to promote critical engagement. In: 2021 IEEE Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT), pp. 1–2. IEEE (2021)
37. Jones, S.T., et al.: We tell these stories to survive: Towards abolition in computer science education. Canadian Journal of Science, Mathematics and Technology Education **21**(2), 290–308 (2021)
38. Kafai, Y.B., Proctor, C.: A revaluation of computational thinking in k–12 education: Moving toward computational literacies. Educational Researcher p. 0013189X211057904 (2021)
39. Kirdani-Ryan, M., Ko, A.J.: The house of computing: Integrating counternarratives into computer systems education. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, pp. 279–285 (2022)
40. Klassen, S., Fiesler, C.: "run wild a little with your imagination" ethical speculation in computing education with black mirror. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, pp. 836–842 (2022)
41. Ko, A.J., Beitlers, A., Wortzman, B., Davidson, M., Oleson, A., Kirdani-Ryan, M., Druga, S.: Critically Conscious Computing: Methods for Secondary Education (2022)

42. Ko, A.J., Oleson, A., Ryan, N., Register, Y., Xie, B., Tari, M., Davidson, M., Druga, S., Loksa, D.: It is time for more critical cs education. Communications of the ACM **63**(11), 31–33 (2020)
43. of Learning Writing Collective, P.: The learning sciences in a new era of us nationalism. Cognition and Instruction **35**(2), 91–102 (2017)
44. Lee, C.H., Soep, E.: None but ourselves can free our minds: Critical computational literacy as a pedagogy of resistance. Equity & Excellence in Education **49**(4), 480–492 (2016)
45. Lin, K.: Do abstractions have politics? toward a more critical algorithm analysis. In: 2021 IEEE Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT), pp. 1–5. IEEE (2021)
46. Lin, K.: Cs education for the socially-just worlds we need: The case for justice-centered approaches to cs in higher education. In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1, pp. 265–271 (2022)
47. Luke, A.: Defining critical literacy. Moving critical literacies forward: A new look at praxis across contexts pp. 19–31 (2014)
48. Mozilla, F.: Teaching responsible computing playbook (2021). URL https://foundation.mozilla.org/en/what-we-fund/awards/teaching-responsible-computing-playbook/
49. Nygaard, K.: Program development as a social activity. In: IFIP Congress, pp. 189–198 (1986)
50. Nygaard, K.: How many choices do we make? how many are difficult? In: Software development and reality construction, pp. 52–59. Springer (1992)
51. Pandya, J., Ávila, J.: Moving critical literacies forward. New York: Routledge (2014)
52. Papert, S.: Information technology and education: Computer criticism vs. technocentric thinking. Educational researcher **16**(1), 22–30 (1987)
53. Philip, T.M., Sengupta, P.: Theories of learning as theories of society: A contrapuntal approach to expanding disciplinary authenticity in computing. Journal of the Learning Sciences **30**(2), 330–349 (2021)
54. Resnick, M., Rusk, N.: Coding at a crossroads. Communications of the ACM **63**(11), 120–127 (2020)
55. Ryoo, J.J., Margolis, J.: Power On! MIT press (2022)
56. Ryu, M., Daniel, S., Tuvilla, M., Wright, C.: Refugee youth, critical science literacy, and transformative possibilities. In: Proceedings of the International Conference of the Learning Sciences (2020)
57. Scharber, C., Peterson, L., Chang, Y.H., Barksdale, S., Sivaraj, R.: Critical computing literacy: Possibilities in k-12 computer science education. Pedagogies: An International Journal **16**(2), 136–151 (2021)
58. Sengers, P., Boehner, K., David, S., Kaye, J.: Reflective design. In: Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility, pp. 49–58 (2005)
59. Shaw, M.S., Ji, G., Zhang, Y., Kafai, Y.B.: Promoting socio-political identification with computer science: How high school youth restory their identities through electronic textile quilts. In: 2021 IEEE Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT), pp. 1–8. IEEE (2021)
60. Soep, E., Lee, C., Van Wart, S., Parikh, T.: 7 code for what. In: Popular Culture and the Civic Imagination, pp. 89–99. New York University Press (2021)
61. Stornaiuolo, A.: Authoring data stories in a media makerspace: Adolescents developing critical data literacies. Journal of the learning sciences **29**(1), 81–103 (2020)
62. Tissenbaum, M., Sheldon, J., Abelson, H.: From computational thinking to computational action. Communications of the ACM **62**(3), 34–36 (2019)
63. Tissenbaum, M., Sheldon, J., Seop, L., Lee, C.H., Lao, N.: Critical computational empowerment: Engaging youth as shapers of the digital future. In: 2017 IEEE Global Engineering Education Conference (EDUCON), pp. 1705–1708. IEEE (2017)
64. Vakil, S.: Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. Harvard Educational Review **88**(1), 26–52 (2018)
65. Vakil, S., Higgs, J.: It's about power. Communications of the ACM **62**(3), 31–33 (2019)
66. Vakil, S., McKinney de Royston, M.: Youth as philosophers of technology. Mind, Culture, and Activity pp. 1–20 (2022)

67. Van Wart, S.J., Vakil, S., Parikh, T.S.: Apps for social justice: Motivating computer science learning with design and real-world problem solving. In: Proceedings of the 2014 conference on Innovation & technology in computer science education, pp. 123–128 (2014)
68. Vasquez, V.M., Janks, H., Comber, B.: Critical literacy as a way of being and doing. Language Arts **96**(5), 300–311 (2019)
69. Vee, A.: Coding literacy: How computer programming is changing writing. Mit Press (2017)
70. Vogel, S.: "los programadores debieron pensarse como dos veces": Exploring the intersections of language, power, and technology with bi/multilingual students. ACM Transactions on Computing Education (TOCE) **21**(4), 1–25 (2021)
71. Vossoughi, S., Hooper, P.K., Escudé, M.: Making through the lens of culture and power: Toward transformative visions for educational equity. Harvard Educational Review **86**(2), 206–232 (2016)
72. Walker, R., Sherif, E., Breazeal, C.: Liberatory computing education for african american students. In: 2022 IEEE Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT), pp. 85–89. IEEE (2022)
73. Weizenbaum, J.: Computer power and human reason: From judgment to calculation. WH Freeman & Co (1976)
74. Xin, X., Moriwaki, K.: Critical coding cookbook: Intersectional feminist approaches to teaching and learning (2022). URL https://web.archive.org/web/20220713222240/https://criticalcode.recipes/
75. Yadav, A., Heath, M., Hu, A.D.: Toward justice in computer science through community, criticality, and citizenship. Communications of the ACM **65**(5), 42–44 (2022)