Yasemin Gülbahar
Erinç Karataş (Eds.)

LNCS 8730

Informatics in Schools

Teaching and Learning Perspectives

7th International Conference on Informatics in Schools:
Situation, Evolution, and Perspectives, ISSEP 2014
Istanbul, Turkey, September 22–25, 2014, Proceedings

Springer

Yasemin Gülbahar Erinç Karataş (Eds.)

Informatics in Schools

Teaching and Learning Perspectives

7th International Conference on Informatics in Schools:
Situation, Evolution, and Perspectives, ISSEP 2014
Istanbul, Turkey, September 22-25, 2014
Proceedings

Springer

Volume Editors

Yasemin Gülbahar
Erinç Karataş
Ankara University
Department of Informatics
Fatih Cad. No: 33
06110 Ankara, Turkey
E-mail:{gulbahar, ekaratas}@ankara.edu.tr

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-09957-6 e-ISBN 978-3-319-09958-3
DOI 10.1007/978-3-319-09958-3
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014947965

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

“This proceeding is dedicated to Roland Mittermeir, our dear colleague and mentor who passed away too early”

Preface

The International Conference on Informatics in Schools: Situation, Evolution, and Perspective (ISSEP), co-hosted by Ankara University and Istanbul University in Istanbul, Turkey, is the 7th in the series of ISSEP conferences. Although Informatics and Information and Communication technologies (ICT) can be used interchangeably in different cultures, the main focus of this conference intends to be informatics and computer science education. Hence, the aim of this conference is to set up collaboration between researchers and practitioners in the area of informatics education and computer science education in schools with a different focus every year.

Informatics and computer science education shifts depending on the advances in computing in terms of teaching, learning, thinking, and problem solving, as well as many other innovative dimensions. The world talks about digital people and curriculums intend to supply competencies for learners to teach digital skills and informatics to create an information society. As also reported by the joint Informatics Europe & ACM Europe Working Group on Informatics Education (2013), informatics education has gained importance as the scientific and engineering basis for the information society, and the global political discourse about the importance of innovation, high technology and information technologies, however, not all the countries started to implement and see the real benefit of informatics curricula due to various reasons. Although some European countries introduced successful informatics fundamentals into their curricula by the 1970s, these efforts have been dropped due to lack of awareness of the importance of informatics and the frequent misunderstanding that digital awareness is all that needs to be taught [1].

Similarly, in a recent report named "Rebooting the Pathway to Success: Preparing Students for Computing Workforce Needs in the United States" written by Kaczmarczyk and Dopplick (2014), it is also underlined that rigorous computer science education should play a core role in secondary education. The findings of this report are rich, and applicable to every culture. According to the report, education in the essentials of computer science is key, both for the individual in terms of promising careers, and for a country developing and sustaining a competitive 21st century workforce and succeeding in innovation [3]. On the other hand, the critical, analytical and computational thinking concepts [2], together with problem solving skills, has gained more attention in recent years, and many tools are being invented to teach these kinds of skills to learners through programing logic. Many countries are looking to embed the appropriate tools and procedures into curricula in order to impart the desired competencies to learners.

Almost all countries are looking for better curriculums to teach computer science and informatics concepts and they are trying to find the best ways for doing this. This conference series contributes to conceptual discussions about didactics and implementation ideas for decision-makers and all stakeholders from a research point of view. Thus, papers about computer science education, competence measurement for informatics, emerging technologies and tools for informatics, teacher education in informatics and curriculum Issues are included in this volume.

The volume consists of two keynote papers and 13 contributed papers. The latter were selected out of 33 submissions; i.e. 39% of papers were accepted for these proceedings. Another 10 papers, 7 short papers and 4 workshop descriptions are published in the local proceedings [4]. Altogether, ISSEP 2014 encompassed presentations covering research and theoretical papers, best practice and country reports, and discussion papers from 18 countries.

The proceedings were opened with two papers by keynote speakers. Walter Gander's keynote aims to provide evidence for revealing that informatics has to become a basic subject in the context of general education in our schools. Hence, the paper starts with some historical background in Switzerland and continues with the current situation, which is followed by the scope of general education and importance of education as a basic subject. Walter Gander clearly discriminates between the concepts, reminds us of the formula suggested by respected authorities "Computer Science in Schools = Digital Literacy + Informatics" and underlines that informatics is the science behind information technology. Thus, he concludes with a clear judgement stating that digital literacy skills are short-lived knowledge, changing with technology; whereas informatics is long living knowledge, lasting forever and does not change with technology.

The keynote by G. Barbara Demo and Lawrence Williams describes diverse Scratch activities proposed in different learning situations. The activities address different competencies from simple to complex ones for different age groups. Especially in the beginner stages, Scratch seems to be a powerful tool, not only for learners to create a complete product to present to other learners that enhances motivation, but also for teachers to introduce a number of fundamental elements of computer science, such as algorithmic complexity. Hence, the researchers underline the advantages of using Scratch for introducing programing experience, since it is easier to switch to other programing languages or environments. Moreover, they stated that Scratch assists and guides learners in the process of achieving competencies about the logic of programing.

The next section, **"Computer Science Education"**, was the most popular heading for this conference, and addresses issues of what to teach and how to evaluate. Jiří Vaníček discussed the issues of motivation, interactivity, multiple-choice answers, and content topics for Bebras Informatics Contest on Informatics and Computer Fluency, since these might be useful for authors of Bebras tasks, as well as for creators of informatics curricula. Andreas Grillenberger and Ralf Romeike pointed out that database concepts and examples commonly used in computer science education need to be updated, based on facts such as the rapidly growing impact of data processing, social and ethical implications of big data, and privacy issues. Being a first implementation of Darmstadt Model for a Latin American country, Nubia Alejandra Fecht and Ira Diethelm provided an analysis of computer science education in Venezuela using this model, which hopefully provides insight for those who will develop, organize and evaluate CSE and ICT lessons. In their paper, Simone Opel and Torsten Brinda explored the different aspects of Learning Field-orientated “Computer Science Education”, in vocational computer science education and “Computer Science in Context” in secondary education, by comparing the similarities and differences of these concepts to derive requirements for a general model of these situated and activity-orientated teaching concepts in computer science education. The paper by Valentina Dagiene, Linda Mannila, Timo Poranen, Lennart Rolandsson and Gabriele Stupuriene, shared the findings from a multi-national study of students' results in the international IT contest “Bebras”, and concluded with the importance of finding more efficient and trustworthy ways of evaluating difficulty levels upfront and choosing a suitable task set. As the last paper of the “Computer Science Education” section, Nataša Kristan, Dean Gostiša, Gašper Fele-Žorž and Andrej Brodnik presented a new system that supports both non interactive and standardised interactive tasks, which they have successfully evaluated in multiple competitions for a potential use in Bebras and related competitions.

In the continuing section named **“Competence Measurement for Informatics”**, there are two papers on competences. Based on the compiled competence models that emerged from the project “Educational standards in vocational schools”, Markus Brunner and Monika Di Angelo provide a didactical concept, that meets the requirements of the educational standards in general, and competence orientation in particular, and a proposal for the implementation of these educational standards in the competence area of industrial information technology. Jonas Neugebauer, Peter Hubwieser, Johannes Magenheim, Laura Ohrndorf, Niclas Schaper, and Sigrid Schubert intended to develop a theoretically and empirically sound competence model for the domains of system comprehension and system modeling. This was developed alongside an evaluated measurement instrument to assess competences of students in upper secondary computer science education in German schools. They used the results to evaluate the competence model, to revise the measurement instrument, and to define proficiency levels in a competence level model by using the method of scale anchoring.

Innovative use of technologies and tools are ways to change the methods and approaches of how we teach. Hence, these issues are addressed under the section named **“Emerging Technologies and Tools for Informatics”**. In their paper, Janka Majherova, Hedviga Palasthy and Emilia Janigova analyzed the use of emerging Internet technologies, known as Web 2.0, within secondary education in Slovakia and compared the opportunities for sharing educational materials, as well as the possibilities of accessing the Internet and its use by pupils. Maciej M. Sysło and Anna Beata Kwiatkowska discussed the advice and results from their observations and didactical experience gathered when teaching about recursion in different contexts and on various education levels (K-12 and tertiary).

Teacher education has always been an important issue to be addressed. Often enough, teachers lack competencies or the insight which is expected of them. Researchers always point out the need to change curricula, however many graduates jump into work without knowing these up-to-date changes and issues. Specific issues about **“Teacher Education in Informatics”** are discussed by several researchers. The paper by Claudio Mirolo outlined the basis of a core module as part of a renewed program offered by the University of Udine, for the education of prospective teachers of informatics, which aims at deepening the teachers’ awareness about the variety of coexisting views and their pedagogical implications. In their paper, Ana-Maria Stoffers and Ira Diethelm investigated teacher profiles for planning informatics lessons, since teachers differ in their perception of self and students’ roles, and in the explanations given for choosing teaching methods, contents, and learning objectives.

The proceedings close with the section on **“Curriculum Issues”** with a paper by Carlo Bellettini, Violetta Lonati, Dario Malchiodi, Mattia Monga, Anna Morpurgo, Mauro Torelli, and Luisa Zecca. They reported on the design of previously conducted workshops which intended to give pupils the opportunity to explore a computer science topic: investigate it first hand, make hypotheses that can then be tested in a guided context during the activity, and construct viable mental models.

With all of these valuable efforts and works, I believe informatics education will become more and more recognized by the authorities and all stakeholders each year. It is my sincere pleasure to thank all those who contributed to ISSEP 2014 with their academic insights. In addition to those already mentioned, there are many people who also contributed to this conference in the preparation, communication, reviewing, organizing, and publishing process. I would like to extend my thanks to the members of Organizing and Programme Committee for their time and conscientious work. Thanks also to Erinç Karataş and Müge Adnan for helping with the proceedings, and finally, special thanks to Ira Diethelm who provided guidance throughout the whole conference process.

References

1. Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education. Informatics education: Europe cannot afford to miss the boat (2013), http://www.informatics-europe.org/images/documents/informatics-education-europe-report.pdf
2. Barr, V., Stephenson, C.: 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 (2011), https://www.iste.org/docs/nets-refresh-toolkit/bringing-ct-to-k-12.pdf?sfvrsn=2
3. Kaczmarczyk, L., Dopplick, R.: Rebooting the Pathway to Success: Preparing Students for Computing Workforce Needs in the United States. Association for Computing Machinery (ACM), New York (2014), http://pathways.acm.org/ACM_pathways_report.pdf

4. Gülbahar, Y., Karataş, E., Adnan, M. (eds.): Informatics in Schools - Local Proceedings of the 7th International Conference ISSEP 2014 - Selected Papers. Ankara University Press, Ankara (2014)

June 2014 Yasemin Gülbahar

Organization

Program Committee

Müge Adnan	Mugla University, Turkey
Ayfer Alper	Ankara University, Turkey
Bahar Baran	Dokuz Eylul University, Turkey
Torsten Brinda	Universität Duisburg-Essen, Germany
Valentina Dagiene	Vilnius University, Lithuania
Ira Diethelm	Oldenburg University, Germany
Çiğdem Erol	Istanbul University, Turkey
Nuray Gedik	Akdeniz University, Turkey
David Ginat	Israel Institute of Technology, Israel
Yasemin Gülbahar	Ankara University, Turkey
Jan Guncaga	Catholic University in Ruzomberok, Slovakia
Sevinç Gülseçen	İstanbul University, Turkey
Juraj Hromkovič	ETH Zürich, Switzerland
Ivan Kalaš	Comenius University in Bratislava, Slovakia
Erinç Karataş	Ankara University, Turkey
Janka Majherova	Catholic University in Ruzomberok, Slovakia
Roland Mittermeir	Alpen-Adria-Universität Klagenfurt, Austria
Ferhan Odabasi	Anadolu University, Turkey
Malgorzata Pankowska	University of Economics in Katowice, Poland
Zerrin Ayvaz Reis	Istanbul University, Turkey
Ralf Romeike	Friedrich-Alexander University Erlangen-Nürnberg, Germany
Carsten Schulte	Freie Universität Berlin, Germany
Sue Sentence	Computing at School Cambridge, UK
Simon Simon	University of Newcastle, UK
Maciej Syslo	UMK Torun, U. Wroclaw, Poland
Erkan Tekinarslan	Abant Izzet Baysal University, Turkey
Sacip Toker	Mehmet Akif Ersoy University, Turkey
Pelin Yüksel	Inonu University, Turkey
Erman Yükseltürk	Kirikkale University, Turkey
Soner Yildirim	Middle East Technical University, Turkey
Recep Çakir	Amasya University, Turkey

Additional Reviewer

Winczer, Michal

Organizing Committee

Yasemin Gülbahar Co-chair	Ankara University, Turkey
Sevinç Gülseçen Co-chair	Istanbul University, Turkey
Soner Yildirim	Middle East Technical University, Turkey
Erinç Karataş	Ankara University, Turkey
Müge Adnan	Mugla University, Turkey
Orçun Madran	Hacettepe University, Turkey

Table of Contents

Informatics and General Education

Walter Gander

Department of Computer Science, ETH Zurich, 8092 Zurich, Switzerland
and Computer Science Department, Baptist University, Hong Kong
gander@inf.ethz.ch
http://www.inf.ethz.ch/personal/gander/

Abstract. Computer Science, Informatics, Information and Communication Technology (ICT), Computational Thinking, Digital Literacy and New Media are just some examples of buzz words which are used and discussed in schools by educators and by political authorities responsible for updating curricula. In many schools informatics is still missing as a basic subject. STEM education still concentrates on teaching science, technology, engineering, and mathematics. In this paper we show that informatics has to become a basic subject in the context of general education in our schools.

Keywords: Computer Science Education, Informatics, ICT.

1 Some Historical Notes

In the following we summarize the development in Switzerland, similar developments took place also in other countries. In the first decade after World War II many universities started to built their own computers. At ETH we were lucky to rent the Zuse Z4 machine from 1950 to 1955 [7]. In 1950 this was the only working computer in mainland Europe in the institute of Eduard Stiefel, the *Seminar für angewandte Mathematik*. In these pioneering years Eduard Stiefel built with his collaborators their own machine, the ERMETH [4]. One of his collaborators, Heinz Rutishauser, developed together with colleagues from GAMM[1] and ACM[2] the programming language ALGOL [5] which I consider the *Latin* of the programming languages. After Rutishauser passed away too early in 1970, ETH continued to pioneering the field of programming languages with Niklaus Wirth who developed PASCAL, and later MODULA and OBERON.

In spite of these great achievements, computer science was not recognized by academia as a new basic science. It took years till the pressure from industry was large enough to convince the ETH management and the other departments to finally introduce 1981 a curriculum for computer science studies at ETH.

In schools the pocket computer was adopted and replaced slowly the slide rule in the seventies, but it was only with the the advent of the personal computer around 1984 that some responsibles for education showed interest to introduce it in schools. Committees were formed to develop teaching material and to

[1] Gesellschaft für Angewandte Mathematik und Mechanik https://www.gamm-ev.de/
[2] Association for Computing Machinery http://www.acm.org/

study how to integrate computer science as a new subject in the STEM-oriented tracks of the Swiss Gymnasia. The minister of education finally decided in 1986 to change the name of the subject "Descriptive Geometry" to "Applied Mathematics" and opened this way the door for young teachers to teach computer science while giving the old teachers the opportunity to continue with descriptive geometry till their retirement.

In 1986 the emphasis of the curriculum in computer science was to learn to program and several schools developed successfully good textbooks for programming in PASCAL. Since the personal computer at that time had almost no software, creativity and spirit of discovery inspired the students, though at the same time many teachers were frustrated by frequent breakdowns and system changes. Many of those enthusiastic former high-school students – the first generation educated in computer science – are now successful computer scientists. Michael Gove, the Secretary of State for Education in the UK, refers to that spirit in his famous speech of January 2012[3]

> With minimal memory and no disk drives, the Raspberry Pi computer can operate basic programming languages, handle tasks like spread sheets, word-processing and games, and connect to wifi via a dongle – all for between £16 and £22. This is a great example of the cutting edge of education technology happening right here in the UK. It could bring the same excitement as the BBC Micro did in the 1980s.

In the following years many applications were developed, computers became ubiquitous, cheaper and easier to handle (e.g. the Macintosh) and the need to develop your own applications vanished. Finally the Internet was born and connected the world. Therefore a strong movement emerged 1995 in Switzerland that teaching programming in schools was no longer necessary. Instead one should concentrate on teaching skills to make good use of the computers. As the applications became more sophisticated and more complex, teachers had to be trained first. The computer industry, in particular Microsoft and Intel, made agreements with whole countries to train the schools on their products[4].

2 Situation Today

Computers determine our life, we live in the *digital age*. We communicate electronically in social networks, we use e-mail and sms. Post-offices are shut down because the paper correspondence has dramatically decreased. We use office software for text processing, presentations and spreadsheet calculations. Books are still being printed but at the same time also made available electronically[5]

[3] `https://www.gov.uk/government/speeches/michael-gove-speech-at-the-bett-show-2012`
[4] `http://partner-fuer-schule.nrw.de/stiftung/projekte/abgeschlossene-projekte/intel-r-lehren-fuer-die-zukunft-i.html`
[5] `http://www.digitalbookindex.com/about.htm`

and devices have been developed for reading such digital books. Radio and television have become digital and music can be downloaded as digital files from servers. We are experiencing the dramatic change in photography: chemically processed films have been replaced by electronic bit-strings. Search machines have revolutionized the way we get our information – Wikipedia has become our encyclopedia, libraries and archives are digitized and become available on-line.

In Switzerland 80% of the young people (age 12–19) possess a smart-phone. The dependency of our youth on the so called "new media" is frightening. According to the Zurich drug prevention agency, doing without smart-phone for three days, can cause already withdrawal symptoms in some adolescents[6]. It is therefore not surprising that educators stipulate media teaching in the curriculum reform which is currently discussed in Switzerland[7].

The confusion is high because many persons equate computer science with using new media. Fruitless discussions and disagreements are unavoidable when people use the same words for different meanings, contents and in different contexts.

To much use of new media leads to "digital dementia" as Julian Ryall writes in The Telegraph on 24 Jun 2013 [8]

> Doctors in South Korea are reporting a surge in "digital dementia" among young people who have become so reliant on electronic devices that they can no longer remember everyday details like their phone numbers.

Informatics Europe[9] and ACM Europe[10] formed 2012 a task force to discuss the issue of computer science education. This Working Group defined in their report[11]:

Computer Science in Schools = Digital Literacy + Informatics

The report states:

> Any citizen of a modern country needs the skills to use IT and its devices intelligently. These skills, the modern complement to traditional language literacy in language (reading and writing) and basic mathematics, are called *digital literacy.*

[6] `http://www.tagesanzeiger.ch/zuerich/region/Wer-eine-Tendenz-zur-Abhaengigkeit-hat-haelt-das-nicht-aus/story/26621914`

[7] `http://www.lehrplan.ch/`

[8] `http://www.telegraph.co.uk/news/worldnews/asia/southkorea/10138403/Surge-in-digital-dementia.html`

[9] `http://www.informatics-europe.org/`

[10] `http://europe.acm.org/`

[11] *Informatics education: Europe cannot afford to miss the boat.* Report of the joint *Informatics Europe & ACM Europe Working Group on Informatics Education, April 2013* `http://www.informatics-europe.org/images/documents/informatics-education-europe-report.pdf`

> Many modern primary and secondary school curricula have started to include digital literacy elements, teaching students to be comfortable with the basic tools of the digital world.
>
> Digital literacy should indeed be a required part of the education of all Europeans. Its teaching should start in first grade and students should be familiar with the basic skills by age 12.

Informatics on the other hand is the *science behind information technology.* The report states.

> For a nation or a group of nations to compete in the race for technology innovation, the general population must in addition to digital literacy understand the basics of the underlying discipline, *informatics.* On the road to an information society, informatics plays the same enabling role as mathematics and physics in previous industrial revolutions

3 General Education

The German word "Allgemeinbildung" refers to a so called *general education* which one needs to develop oneself as a human being. It is the fundamental knowledge which humans need to acquire to orient themselves in the world.

At the time of Leonardo da Vinci (1452-1519), a gifted person like him could achieve general education by studying all available books. Today due to the knowledge explosion even specialists have problems to keep up with the literature in their field. It is said that half of all authors who have ever written something are still alive ...

The goals of the general education in Swiss Gymnasia are defined in the regulations issued by the Conference of Directors of Education of the Cantons[12].

Here is an excerpt from these regulations:

> The aim of Secondary Education is to provide to the students *basic knowledge* with respect to lifelong learning as well as to promote their open-mindedness and the ability to independent judgments. The schools are striving for a broad, balanced and coherent education and not to provide specific skills or vocational training.

One of these goals is also:

> High school graduates know and are familiar in their natural, *technical*, social and cultural environment, and this in relation to the present and the past, at national and international level.

[12] Verordnung des Bundesrates/Reglement der EDK über die Anerkennung von gymnasialen Maturittsausweisen (MAR) vom 16. Januar/15. Februar 1995. `http://www.edk.ch/dyn/13723.php`

Thus the purpose of general education is to provide *fundamentals*, long lasting *basic knowledge* and not *ephemeral knowledge.* Furthermore this basic knowledge is focused on natural, social and cultural but especially also on *technical* environment.

Traditional fundamental technical subjects which are unquestioned are mathematics, chemistry, physics, biology. There is no high-tech without mathematics, no engineering without physics and chemistry, no medicine without biology.

However, in our modern world *nothing works without computer science!* Computer science has become a *leading science*, it is responsible for progress and breakthroughs in all sciences. *It is high time that informatics finally be established as a basic subject in all schools.*

Franklin D. Roosevelt said September 20, 1940 at the University of Pennsylvania:

> We cannot always build the future for our youth, but we can build our youth for the future.

And Simon Peyton Jones wrote 2012:

> European nations are harming their high school students, both educationally and economically, by failing to offer them an education in the fundamentals of computer science.

4 Informatics as Basic Subject

Informatics should be introduced as one of the fundamental basic subjects in our schools. However, the schools did not cope with the change of our society to an information society in a digital world. As ACM and CSTA write in their report *Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age*[13], the schools are lagging behind this development. The report finds that

> Major gaps exist in the adoption of computer science standards at the secondary (high school) level. Only 14 states have adopted secondary state education standards for computer science instruction to a significant degree (defined as more than 50% of ACM and CSTAs national model computer science standards), leaving more than two-thirds of the entire country with few computer science standards at the secondary school level. Further, 14 states (and the District of Columbia) do not have even one upper-level standard for computer science instruction as part of their secondary education standards

Most states treat high school computer science courses as simply an elective and not part of a students core. This is an amazing step backward since already in 1969, Sandra Forsythe, the wife of George Forsythe (the founder of the

[13] `http://www.acm.org/runningonempty/`

computer science department in Stanford and one of the fathers of Silicon Valley) had written a textbook for schools *Computer Science, A First Course* [1]. This book is remarkable – many fundamental topics can still be taught today without changes: programming, algorithms, data structures (e.g. trees), stepwise decomposition, procedure and functions.

What is the goal of informatics education? George Forsythe wrote already 1968 [2]:

> The most valuable acquisitions in a scientific or technical education are the *general-purpose mental tools which remain serviceable for a lifetime.* I rate natural language and mathematics as the most important of these tools, and computer science as a third ...
> The learning of mathematics and computer science together has pedagogical advantages, for the basic concepts of each reinforce the learning of the other.

George Forsythe was indeed farsighted! It is amazing how precise he predicted already 1963 the importance of computer science [3]:

> Machine-held strings of binary digits can simulate a great many kinds of things, of which numbers are just one kind. For example, *they can simulate automobiles on a freeway, chess pieces, electrons in a box, musical notes, Russian words, patterns on a paper, human cells, colors, electrical circuits, and so on.* To think of a computer as made up essentially of numbers is simply a carryover from the successful use of mathematical analysis in studying models ... *Enough is known already of the diverse applications of computing for us to recognize the birth of a coherent body of technique, which I call computer science.*

Today informatics is the science of systematic, automated processing of information (representation, storage, management and communication). The goals of informatics as a basic subject is (a) to teach the students to understand the principles and functioning of today's digital world and (b) to train the students in constructive problem solving.

Jan Cuny, Larry Snyder, and Jeannette M. Wing coined the term "Computational Thinking"[14]. What they mean is:

> Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.

Jeannette M. Wing wrote in [6]:

> It [Computational Thinking] represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

[14] `http://www.cs.cmu.edu/~CompThink/`

Computational thinking is a methodology for anyone to be used to solve problems. Especially it applies to problems solving with computers. It involves the following steps

- *Analyze a task or problem*, model and formalize it.
- Search for a way to solve it, find or design an *algorithm*.
- *Program.*
- *Run the program*: let the computer work, maybe correct, modify the program, *interpret the results.*

Programming is an essential step in this process. It is an important activity for all the students in general education. It is *creative* (there are often several ways to solve a problem. It is *constructive* (the solution has to be constructed and reached in finite steps). Programming finally teaches *precise working* (any small error prevents the solution) and trains *computational thinking.*

5 Conclusions

We have argued that computer science in the school must consist of two parts:

1. Learn to make good use of IT and its devices. We call this *Digital Literacy.* These skills are short living knowledge, they change with technology.
2. Learn the fundamentals of computer science which are essential to understand our digital world. We call this *Informatics.* It is long living knowledge which lasts forever and does not change with technology.

References

1. Forsythe, A., Kennan, T.A., Organick, E.I., Stenberg, E.: Computer Science, A First Course. John Wiley & Sons (1969); 2nd edn. (1975)
2. Forsythe, G.: What to do till the computer scientist comes. Amer. Math. Monthly 75, 454–462 (1968)
3. Forsythe, G.: Educational implications of the computer revolution. In: Freiberger, W.F., Prager, W. (eds.) Applications of Digital Computers, pp. 166–178. Ginn, Boston (1963)
4. Neukom, H.: ERMETH: The First Swiss Computer. IEEE Annals of the History of Computing 27(4), 5–22 (2005)
5. Rutishauser, H.: Description of ALGOL 60, Handbook of automatic computation, vol. 1, Part b. Springer, Berlin (1967)
6. Wing, J.M.: Computational Thinking. Communications of the ACM 49(3) (2006)
7. Zuse, K.: The Computer – My Life. Springer (1993)

The Many Facets of Scratch

G. Barbara Demo[1] and Lawrence Williams[2]

[1] Informatics Department, University of Torino
C.so Svizzera 185, 10149 Torino, Italy
barbara@di.unito.it
[2] School of Sport and Education, Brunel University
Uxbridge, Middlesex, UB8 3PH, UK
lawrence.willams@brunel.ac.uk

Abstract. This paper describes different Scratch activities proposed in different learning situations: for primary school-children aged about 9-10 years old; for in-service teachers active in various levels and types of schools, but with very low or without computing competences; for teachers and pupils in middle schools; for students in their beginning two years of technical secondary school (not necessarily specialized in informatics); and for future informatics teachers having a good knowledge of computer science. In all of these situations, Scratch has been introduced as a system for producing original stories, rather than as a programming system. For beginners, this choice proves to be successful in inspiring and motivating their intent to carry on with new activities. This success is mostly due to the possibility of producing, in a relatively short time, artifacts that teachers can immediately use in school; and the students feel they have created a complete product to show to other students (or to the family and to friends). Informatics experts, in particular informatics teachers, discover a pedagogical methodology they are not used to for introducing to programming. All the while, by using Scratch, we can introduce a number of fundamental elements of computer science such as algorithmic complexity. Also, students go through an introductory programming experience, making a faster and smoother change to other programming languages and other environments. In general, the use of Scratch supports and facilitates a process toward achieving competences that many consider necessary to our future young people.

Keywords: fostering creativity, training teachers, basic computing competences.

1 Introduction

Among the fundamental informatics competences that everybody should acquire, programming is the most concrete, and probably the most particular, aspect of computing for people having little or no acquaintance with computer science. Besides, it can be a pleasant way of acquiring the basics of informatics, because one may produce artifacts which are useful and enjoyable to interact with, provided that suitable software is used. Scratch is one of the systems best suited to this purpose.

Scratch has been developed at the MIT Media Lab by the Lifelong Kindergarten group directed by Mitchel Resnick. The aim focused on building a system for beginners where they could express themselves and their creativity while being introduced to informatics [11]. It is not only a programming language: it also provides an environment where a user finds several integrated tools (for drawing and recording, for example). More generally, Scratch is a system for producing stories that can have one or more characters (Sprites) acting on a Stage, with one or more Backgrounds, and sounds of different types (for example voices, music, noises). Characters behave as specified by means of code sequences, called Scripts. Here the word script is to be intended mostly as referring to roles in the theatre, or cinema. Of course, characters and their behaviours may be of quite diverse types: they can describe a journey; tell a joke; outline a narrative episode which happened to the storyteller; can be the instructions for using or doing something; or a story can be an original creative story [13]. Besides this, in Scratch, we can specify solutions to different problems, which is what programmers actually do while working in more sophisticated programming environments [1]. Unfortunately, often teachers who are expert in informatics propose traditional algorithms developed in a traditional way, as well, during the first activities with Scratch. Therefore, the students lose the opportunities for engagement in original activities, nearer to the spirit of the system.

This paper describes a pedagogical methodology emphasizing the several facets of Scratch. More specifically, section 2 concerns the use of Scratch for introducing to computing students and teachers from all types and levels of education, having no, or very little, informatics knowledge. In section 3, the focus is on how beginners are asked to develop some personal story which helps them understand that some programming components are present, also, in our everyday life. That means they are already in our way of thinking. Section 4 concerns the use of Scratch in middle school where interdisciplinary activities are particularly important to teachers and pupils.

All the activities described have been conceived, designed and developed:

- together with teachers, i.e. those who work in schools. They have also developed these activities with their students. Indeed, we consider it vital to stimulate and collect suggestions from teachers, and take into account their activities in schools
- in compliance with the European guidelines, to avoid juxtaposing initiatives which often cause lack of depth and confusion in schools
- with a practical mind, considering the several difficulties normally present in schools, such as disciplinary problems, poverty of means, and poverty of school time, because often we can count on just one hour of continuity for an activity. As an example: one of the scheduled projects could not be carried out because: "the informatics laboratory does not exist any more: computers have been moved one per classroom to maintain the digital class-register", as one of the teachers said.

In all these situations, Scratch has proved to be a program development environment that has been successful in inspiring and motivating pupils, with the intent to carry on with new activities. These are suitable for producing, in a relatively short time, material to be immediately used in school, or felt as a complete product to be shown to other students (or even to the family and to friends). In the environments where we carried out our experiences, Scratch has been shown to stimulate the pupils' dedication to these activities, and the will to continue. We could not have the same working atmosphere during introductory activities when we used different environments for different programming languages.

In the last section, among conclusive comments, we suggest that other programming environments using different languages are suitable to continue the experiences for enhancing the informatics competences of students and teachers.

2 Introducing Computing

The first reason why story-telling fits introductory activities well is that it starts with a simple coding exercise yet producing a stimulating result. Indeed a story is a sequence of actions, performed by the different characters, one after the other (or in parallel, but we do not consider this possibility with beginners). Despite this simplicity, the results can be very stimulating for both children and adults: an example is the Solar System Discovery developed by Lawrence Williams and his students.

See: http://www.literacyfromscratch.org.uk/index.htm

On Williams' web pages one can find many other stories and interesting materials.

The above two reasons suggest we should centre on a short story as the first activity, especially for people knowing almost nothing about informatics. To be as simple as possible, our beginning stories have characters' actions synchronized simply on a time base, i.e. "after n seconds from the start" a given script of character X begins.

The entire project methodology is based on an active learning paradigm defined by Lawrence Williams, and already experimented with different groups of attendees in Italy and in UK [9]. Notice how, our working group being English-Italian, we began with stories in these two languages (more languages were developed after attendees from different countries came to our workshops).

The steps of the first activity are the following:

a. a story is shown full screen (dialogues are in English for activities in Italy, vice versa for activities in UK),
b. the components of the Scratch system are swiftly introduced
c. attendees are then asked to translate the first few sentences of the story in the language they like better. This allows them to look into (and through) the code to choose the sentences to be translated. During this step, usually attendees "discover" different aspects of the system presented in the previous step, for example that one script belongs to one character, that one character may be associated more than one script,
d. the translation step can result in different, though scaffolded, changes of the original story regarding images, sounds and dialogue
e. results are uploaded in a shared environment, and discussed together.

During these steps, attendees work and produce something they immediately feel is their own, something they can show and, for teachers, something they can propose to their students. This approach is warmly appreciated by the participants.

If new dialogues are introduced, they can cause problems in synchronizing the characters' actions. Comments on this aspect often lead to consideration of different possibilities for the characters entering the scene, or performing some actions, thus broadening the knowledge of the system. Also, discussing all together the story already implemented, and the changes made, gives a chance to share new aspects of the system among participants.

Story-telling is particularly important in primary and middle schools, but not exclusively. As already pointed out, an engaging story can be implemented with very few commands used in a repeated pattern to produce the whole story. This aspect is important for anyone new to programming. Obviously it is more important if we think to a story-telling activity in a primary and middle school, where the focus of the learning, for the pupil, is not on the actual coding. Rather, the pupil is developing a narrative, with characters and dialogue, and the coding is merely the tool for presenting the story in an entertaining way. This means that when difficulties arise, the pupil wants to overcome them in order to complete the story, rather than simply getting some abstract coding correct. Besides, story-telling can be a common use of Scratch in all disciplines of middle school.

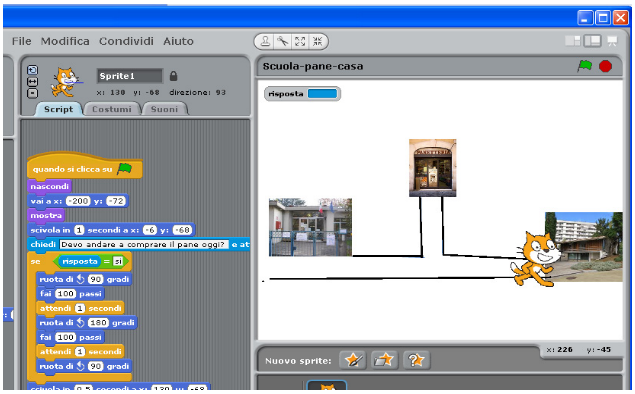
3 Personal Stories and Concrete Programming

There are many projects proposing computing science activities without a computer beginning, with the well known CS-unplugged project by Tim Bell, Ian H. Witten and Mike Fellows [3], several contests for primary and middle schools such as the Bebras contest that was started in Lithuania by Valentina Dagiene in 2004 [2, 6, 12], Olympics of problem-solving started in Italy by Giorgio Casadei in 2006 with problems to solve on paper [5].

Algorithms are not exclusive to the digital world: they have to do with our lives. In 2006, for the master course of the Italian MIUR "Enhancing teaching of scientific disciplines in primary school", attendees were presented with informatics activities introducing them to programming mostly "unplugged", based on discussing algorithms present in our everyday normal life. Often educational robotics is seen as a proper introduction to programming, because it gives the opportunity of "concrete programming" i.e. making physical artifacts like small robots move around, avoiding obstacles, as pupils would do if they were asked to [8].

The procedures for performing arithmetic operations that everybody learns in primary school are examples of algorithms which we normally use, without knowing the name. Bringing everyday experiences into programming points out this relationship, and makes first programming experiences easier. As an example, we have asked students to tell their personal stories about going back home from school. One story was: "On Tuesday, I go to my grandparents' home, while the other days I go to my place." Another one was: "Going back home from school if we need bread I go to the bakery, otherwise I go directly home for lunch." This latter story wass specified in the Scratch activity shown in figure 1.

In Scratch, we can explicitly transmit the relationship between some students' behaviour and its specification in a programming language, beginning with pupils' personal experiences, or personal stories such as holiday stories, for example. This makes programming the concrete experience that is called concrete programming dealing with educational robotics [8].



```
quando si clicca su
nascondi
vai a x: -200 y: -72
mostra
scivola in 1 secondi a x: -6 y: -68
chiedi Devo andare a comprare il pane oggi? e at
se risposta = si
    ruota di 90 gradi
    fai 100 passi
    attendi 1 secondi
    ruota di 180 gradi
    fai 100 passi
    attendi 1 secondi
    ruota di 90 gradi
```

Fig. 1. "Going home from school: do we need bread?"

Scratch allows personal activities to become the story of the scene, because pupils make a character (a sprite) behave like one of them would behave in several environments. In figure 1 the background shows a path going from school to a house with a street on the left reaching a bakery ("panetteria"). When the sprite arrives at the corner he asks, "Do we need bread today?" and goes to the bakery if the answer is positive, otherwise he goes directly home.

In this way, a pupil describes the background what would be his/her environment, draws his/her avatar/sprite, and specifies in Scratch the scripts for the sprite to have his/her behaviour. The answer yes to the question, "Do we need bread?", asked to the user running the program, makes the sprite go to the bakery, then turn to go back to the crossroad, and finally go home, as the schoolchildren would have done in case the bread was not necessary.

These activities have been proposed to students from primary to first year of secondary school during lectures introducing programming, after story-telling activities.

Everyday life has plenty of stories like the School-bakery-home in figure 1. These stories are problems already solved, because adults and young people live them every day: they are our personal stories. Teaching characters to behave in the same way only or basically requires learning how to express, in a digital environment, the actions we perform in our normal life. For someone attending lectures introductory to computing, this is a going from story-telling to telling-my-story.

Other activities can be introduced similarly. In [8] the holidays that some schoolchildren spent in their summertime are "translated" into a journey for an NXT robot. Arriving in the town where the grandmother of one student is living, and not finding her at home, results in going to somewhere else, and then back until she arrives home.

This is another example of a personal-story expressed using the repeat-until-grandmother-arrives pattern that doesn't require much explanation, and becomes the repeat-until-condition-satisfied pattern in the student's mind.

An evolution of the stage just described concerns developing "stories-with-crossroads" or interactive stories i.e. more sophisticated stories having not only sequential actions yet with a development that depends on the interactions of the sprites with the users who are looking at the story, or on the interactions with other characters.

4 Interdisciplinary and Advanced Activities

From middle school, different disciplines are taught by different teachers. In Italy, there is no discipline where teaching informatics is mandatory, nevertheless the National Indications of the Education Ministry wish for having some programming activity in the Technology discipline. The problem is that this discipline has only two hours per week, and also when several other technologies are taught. Moreover, technology professors normally are architects having very little, if any, knowledge of informatics. As a consequence, informatics risks having little place in primary schools, and no place at all in the middle ones. To ride over this problem, it is important to propose tools making easy-to-share activities among different disciplines, and to spread introductory computing competences to as many as possible teachers and students.

Among early interdisciplinary activities in a middle school, we had practical experiences with simple linear equations in one variable involving Mathematics teachers. It is easy to find examples of simple linear equations and lectures on how these can be solved. It is uncommon to find how to put them into practice. While working on educational robotics, we discussed with teachers why modeling a problem by an equation was not a frequent exercise. An example for measuring a path, with circuits covered by a small robot, is proposed in [7]. In our Scratch activities, we discussed with the mathematics professors the "guess a number" game that students sometimes play at school, already described in [1]. We had the students develop a program in which a cat asks the user sitting in front of the screen to do actions such as: "think of a number between 1 and 9" (call it x), "now add 1", "multiply the result by 2", "subtract the number you thought of at the beginning", "subtract 4". At this point the cat says "Tell me the number you finished with". Once the user says his/her answer, call it y, the cat computes x=y+1 and says the original number x. In the referred case the result comes from the equation 2(x+1)-x-4=y. The script with the commands that the cat gives during the game is simply a sequence of instructions, followed by the number of seconds before the next command is given. Input and output communication commands are used. This, and similar experiences, normally lead to discuss and work with Mathematics teachers, who analyze the game as an exercise on one variable equation.

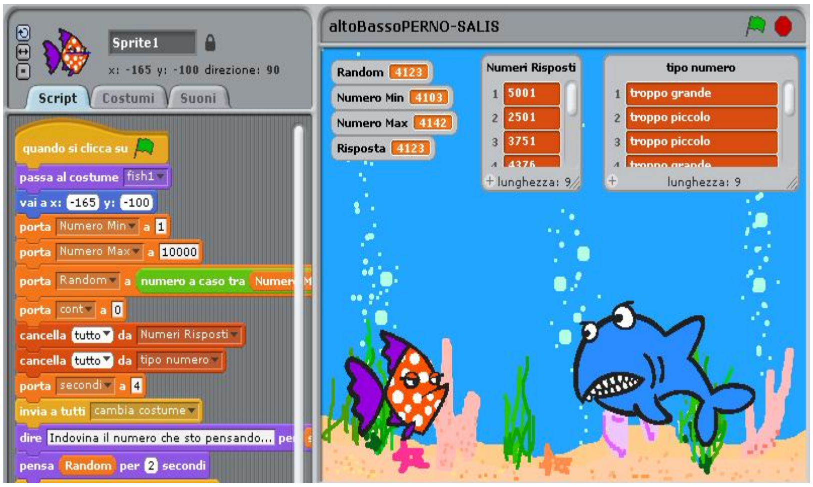
Another example is the photosynthesis activity: after a lesson about the photosynthesis a story was developed by assembling together ideas and drawings from different Scratch activities that pupils had implemented.

Other pupils have developed stories describing monuments and curiosities of their country or where they tell jokes or outline narrative episodes that happened to them.

In a secondary school, students about 16 years old developed a Scratch video-game on the "Divina Commedia" by Dante Alighieri. In it, the scenes include many distinctive traits of the poem, and the player increases her/his score answering to questions verifying his/her comprehension. In a technical secondary school a group of students implemented the simulation of a token ring net: by using Scratch this activity was developed in a short time and students were allowed to focus on the peculiar aspects of the simulated net type.

It is worth noticing that, in a secondary school, when the informatics professor assigned typical algorithmic exercises, the students implemented their solutions with sprites asking and answering in a story-telling fashion, with coloured environments and jokes.

All the while, by using Scratch, we can introduce a number of fundamental elements of computer science such as algorithm complexity. An example is the activity HighLow (altoBasso) shown in Figure 2 developed by the students Perno and Salis in their second year of a technical secondary school in Torino. In HighLow, Smallfish chooses an integer between 1 and 10000. Bigfish tries to guess the number with no help other than "your number is too high"("troppo grande") or "…too low" ("troppo piccolo") told by Smallfish. The goal of the activity is to guess the number with the lowest number of tries. The activity begins playing in class the game among students trying different strategies. After a while, the students begin to specify their own strategies and to express them in Scratch. Usually some groups of students come out with a sort of binary search strategy as the one implemented in Figure 2.



```
quando si clicca su
passa al costume fish1
vai a x: -165 y: -100
porta Numero Min a 1
porta Numero Max a 10000
porta Random a numero a caso tra Numer
porta cont a 0
cancella tutto da Numeri Risposti
cancella tutto da tipo numero
porta secondi a 4
invia a tutti cambia costume
dire Indovina il numero che sto pensando... pe
pensa Random per 2 secondi
```

Numeri Risposti	
1	5001
2	2501
3	3751
4	4376
lunghezza: 9	

tipo numero	
1	troppo grande
2	troppo piccolo
3	troppo piccolo
4	troppo grande
lunghezza: 9	

Fig. 2. "HighLow" activity

5 Concluding Remarks

The experiences here described have been developed over several years, ever-increasing our appreciation of the Scratch environment for introducing computing science both to students and teachers. After all, Scratch was not initiated as a programming environment for a general use, and it is unreasonable to think of, or introduce it as such. Scratch authors aimed at building an environment that was easy to use for creating digital stories. Let's remember that procedures were not available for a long time, and have been introduced only through some extension project, for example the Berkeley ByOB project [4], or the Scratch 2.0 version available from summer 2013. For students choosing to continue, and wanting to develop a deeper programming competence, we suggest, for example, Python as the next language. It is a textual language, interesting because it maintains some simplicity, but also it is offered in programming environments that are nearer to those of languages more used in computing, such as C or Java.

We propose the activities here described as part of new curricula introducing to computing science teachers and students in k-12 education. Since years computer science researchers ask to change the curricula usually present in our schools mostly offering digital literacy abilities to our students. It is worth noticing that also several education researchers express critical evaluations toward the role played by the digital tools as they are currently present in our schools [10]. We think that these two critical attitudes, coming from two different directions, expose an undeniably problematic state of the informatics presence in k-12 education. It could be positive for future curricula if computer science and education researchers should converge, because of their common negative judgements, and together conceive new informatics curricula.

As we said above, the activities here described were offered in training courses for both in-service and future teachers, or directly to students and their teachers in schools. Teachers with very low or no informatics knowledge have been guided to practice the many different Scratch facets here described. The aim was that after our workshops they could, by themselves, adjust in collaboration with their students, the activities developed together.

The informatics teachers have learnt that informatics can be developed by telling stories, and that, for example, the binary search can be developed as a riddle. Now, using Scratch in the first two years of secondary schools, they have students developing coding as a story, even finding the maximum and minimum of a set of numbers. Mathematics teachers have built, together with their students, activities where they put in practice other parts of the curriculum. We mentioned here the idea of a game, where linear equations in one variable are used; or simply they developed a memory riddle for training in making calculations. Similar ideas have been developed in other disciplines. In all cases, it turns out that Scratch allows us to see the computer, and programming, also as powerful tools to express everyone's creativity.

Acknowledgments. We are quite grateful to all the teachers and the students we worked with particularly to those who are still creating and proposing with us to their students and colleagues new educational activities. Discussions with Alberto Barbero, Alessandro Rabbone and Fabrizio Ferrari are always inspiring.

References

1. Barbero, A., Demo, G.B.: The Art of Programming in a Technical Institute after the Italian Secondary School Reform. In: Proc. ISSEP 2011, Bratislava (2011)
2. Bebras Contest, International Contest on Informatics and Computer Fluency, http://bebras.org/
3. Bell, T., Witten, I.H., Fellows, M.: CS Unplugged (1998), http://csunplugged.org
4. ByOB, Build Your Own Blocks 4.0, http://snap.berkeley.edu/ (last visited March 1, 2014)
5. Casadei, G., Teolis, A.: k-12 Problem Solving Olympic Games. In: Proc. Didamatica 2009, Trento (2009) (in Italian)
6. Dagienė, V.: Informatics Education for New Millennium Learners. In: Kalaš, I., Mittermeir, R.T. (eds.) ISSEP 2011. LNCS, vol. 7013, pp. 9–20. Springer, Heidelberg (2011)
7. Demo, G.B.: From Mini Rover Programs to Algebraic Expressions. In: Proceedings 10th IEEE International Conference on Advanced Learning Technologies, ICALT, pp. 336–340 (2010)
8. Demo, G.B., Marcianò, G., Siega, S.: Concrete Programming using Small Robots in Primary Schools. In: Proc. 7th IEEE International Conference on Advanced Learning Technologies, ICALT (2007)
9. Demo, G.B., Williams, L.: Scratch Story-Telling for Introducing Computing to In-service Teachers, Internal Report, Informatics Department of the University of Torino, Italy (September 2013)
10. Ranieri, M.: Le insidie dell'ovvio. Tecnologie educative e critica della retorica tecnocentrica (The Pitfalls of the obvious. Educational technologies and critique of the technology-centred rhetoric). ETS Publisher, Pisa (2011) ISBN: 9788846727916 (in Italian)
11. Resnick, M., et al.: Scratch: Programming for All. ACM Communications 52(11), 60–67 (2009)
12. Sysło, M.M.: Outreach to Prospective Informatics Students. In: Kalaš, I., Mittermeir, R.T. (eds.) ISSEP 2011. LNCS, vol. 7013, pp. 56–70. Springer, Heidelberg (2011)
13. Williams, L., Cernochova, M.: Literacy from Scratch. In: Proceedings of the 10th IFIP World Conference on Computers in Education, WCCE 2013, Torun, Poland, July 2-5, pp. 17–27. Copernicus University Publ., Torun (2013)

Bebras Informatics Contest: Criteria for Good Tasks Revised

Jiří Vaníček

University of South Bohemia in České Budějovice, Czech Republic
vanicek@pf.jcu.cz

Abstract. The Bebras International Contest on Informatics and Computer Fluency has significantly grown in the number of participating countries and participants in recent years. Six years ago Dagienė and Futschek determined the criteria for good contest tasks, which are frequently used by the International Bebras Committee for selecting and improving tasks for national contest organizers. New experience and findings from several surveys allow us to reconsider these criteria from a new viewpoint and to assess which of these criteria are still actual, which need revisions and whether some new criteria are needed if the process of creation of informatics tasks for the contest is to be improved. The paper discusses the issues of motivation, interactivity, multiple-choice answers and content topics. The reviewed criteria and categories might be useful for authors of Bebras tasks as well as for creators of informatics curricula.

Keywords: teaching informatics, computer science education, informatics task, informatics contest.

1 Bebras Contest and School Curricula

The Bebras project [1] can undoubtedly be regarded as one of the most important milestones in the area of reintroduction of informatics content into schools across Europe over the past few years. In contrast to "official" ways such as definition of national curricula, taxonomy of learning objectives, outcomes and contents, the form of an informatics contest designated for average eager pupils interested in technology approaches the issue from the other pole: it looks for suitable topics and problems, suitable situations, didactical transformations of the original informatics topic.

Contest tasks focus on problem solving [2]. The process of their creation begins with authors from a number of participating countries, which is followed by an international workshop where tasks are selected and classified, improved and their wording is refined, after which contests are prepared by individual nations, tasks translated, finalized and used in the contest. This process is therefore a very good way to creation of a rich set of tasks that can also be used in school curricula in many countries.

The tasks represent small isolated problems and situations that can be integrated into the subject that is still very novel in schools in many of the participating countries. Thus the Bebras contest may play an important role in creation of school curricula from the "bottom", from basic elements, from individual questions, on which broader informatics concepts may be illuminated. School informatics is built on different types of tasks than tasks used in the Bebras contest; it uses practical and more extensive problems, projects, inquiries. The Bebras contest presents those types of tasks that are not based on practical activities on the computer to demonstrate the acquired skills. In this respect they are close to school mathematics, which also poses artificial problems that define the didactical situation. These problems can address concrete misconceptions of pupils, can focus on showing extreme situations in which understanding of a given concept is refined. The characteristics of informatics tasks in the contest are similar in some of these parameters.

School informatics, especially in the area of teaching algorithmization, has been using a whole range of software environments, microworlds and didactical programming languages. These learning environments provide the background for concrete practical tasks and more extensive pupils' programming projects. Mathematics educators also use ways of creating learning environments based on one task. Wollring mentions the relation "Task $\subset$ Task format $\subset$ Learning environment" and introduces 6 basic principles for designing learning environments: subject matter and meaning, articulation and social organization, differentiation, logistics, evaluation and links to other learning environments [3]. Learning environments that are close to children's everyday experience, e.g. Snake, Bus, Family, Stairs are used by Hejný in his scheme-oriented approach to mathematics education [4]. Bebras tasks can make a convenient starting point for creation of such learning environments that can be used in contest tasks and that will conveniently supplement environments already used when teaching algothmization. Creation of learning environments and research in problem posing may be one of the possible future trends in didactics of informatics.

2 Criteria for a Good Task

The history of the Bebras international contest on informatics and computer fluency (*http://bebras.org*) dates back to 2004 when Valentina Dagienė organized this contest for the first time in Lithuania [1]. The main goal of the contest was to motivate pupils to study informatics. The idea of "learning informatics through contest" has become popular in a number of countries. Last year the list of participating countries featured 29 countries from 4 continents. The Czech Republic joined the contest in 2008. Czech organizers' intention was to demonstrate to pupils as well as to their teachers how wide the field of informatics was [5]. This was in response to situation in informatics education in the Czech Republic, where compulsory ICT education is limited to information literacy, user-approaches to technology, to teaching how to consume technology.

In 2008, Dagienė and Futschek (in cooperation with Hein, Pohl, Cock, Sysło) published the paper *Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks*, in which they classify tasks into two content categories and formulate several key and recommended criteria for creation of informatics tasks. [6] These principles proved to be very useful for us as authors and organizers. They served as a good guide in coordination and selection of questions and their finalization in international workshops. The listed criteria considerably contributed to improvement of task quality and the process of their preparation.

Six years have passed since publication of the paper. This is a sufficient period of time for getting experience. It is now time to consider whether the listed categories and criteria are still topical or whether other useful criteria should be introduced. Our experience from work on the international team, our assessment of more than 300 tasks that have been used in the Czech national rounds of the contest within the past six years and several surveys among contestants and teachers - school coordinators serve as the starting point for discussion of some of the criteria.

3 What Should the Contest Be Testing?

Dagienė and Futschek mention that "users need also some thinking skills while applying technology" [6]. This could be referred to as computational thinking, whose definition for K-12 is currently much discussed on the grounds of CSTA and ISTE [7]. Computational thinking approach to informatics education at upper secondary schools is also discussed by Syslo [8], the same approach can be come across in the conception of Slovak national curriculum framework for primary schools [9] and in Computing program of study in the national curriculum framework in England 2014 [10].

Dagienė and Futschek claim the best way to develop thinking skills is to solve problems [6]. And it is true that this basic demand on a good informatics contest task has been met over the years. Gradually, tasks asking about important personalities and events of informatics history, tasks testing knowledge of facts or mastered knowledge of e.g. some algorithms have disappeared from task proposals. Authors of tasks also try to meet another demand that the contestants need no pre-knowledge when solving a task [2, 6]. They try to include all the needed information in the assignment. However, this leads sometimes to very lengthy assignments. Authors face the dilemma whether the additional information does not make the task less comprehensible and clear.

We tried to verify this experimentally in the Czech national round in 2013. The task used in this experiment was an interactive task about passage through labyrinth. We wanted to test whether reading a long text of the assignment may not be more difficult for the contestants than the possibility to discover the behavior of the system experimentally. Two versions of the same task were prepared. 14-15 year old contestants were assigned this task in the form of full verbal description of behavior of the system. 10-11 year old contestants were assigned the same task shortened to one quarter of the original length, only with basic instructions and a short task. When preparing the contest some teachers - pre-testers objected that the assignment for younger pupils was too brief. The same objections could be heard from older contestants who has taken the variant for younger contestants. However, younger pupils had no problems when solving the task in the competition. More than 90 % of the 6031 answers were correct, which clearly shows that the contestants were able to grasp the brief, incomplete assignment. This suggests that the possibility to discover rules on one's own may be more advantageous in certain questions than having to read them formulated.

It is most difficult to provide all the relevant information in a contest task as required by the criterion if the task is related to everyday work with computer, digital literacy or broader social context.

4 What Areas Should the Tasks Stem Out From?

Dagienė and Futschek introduce the following proposals of topics for the contest. They are six:

INF	Information comprehension - representation (symbolic, numerical, visual), coding, encryption
ALG	Algorithmic thinking including programming aspects
USE	Using computer systems (e.g. search engines, email, spread sheets ets. - general principles, but no specific systems.
STRUC	Structures, patterns and arrangements - combinatorics, discrete structures (graphs, etc.)
PUZ	Puzzles - logical puzzles, games (mastermind, minesweeper, etc.)
SOC	ICT and society, social, ethical, cultural, international, legal issues [6]

This is not the only existing proposal for categorization of contest tasks. E.g. Kalaš and Tomcsányiová propose categorization of informatics tasks into four categories: algorithmization, information comprehension, problem solving, digital literacy [11].

It is a question how well these topics cover the field of informatics and whether tasks can be distributed equally into the different topics, whether the proportion of tasks from each of the topics is about the same. To answer the question we analysed all proposals that were sent by authors for review in 2012 and 2013. The analysis of 424 proposals shows that:

- More than one half (216) of the tasks were classified by their authors as ALG type.
- 23 % of the tasks could not be classified into one topic category, their authors placed it into two (or more) topics simultaneously
- Some authors were not happy with the offered topics and used their own classification of the task type, e.g. *languages, combinatorics, graph, logic, sequence, constraints*.
- only 7 % of the tasks could be classified as USE (using ICT, digital literacy) and only 2 % as SOC (social and legal issues in use of ICT)
- other topics were represented as follows: INF 26 %, STRUC 19 %, PUZ 11 %.

The aim of defining topics is among others to guarantee that contests offer a variety in content and cover the whole spectrum of informatics tasks. Current distribution of tasks suggests that the topics are not well defined.

It seems purposeless to have a category that does not offer sufficient number of tasks. That is the case of the topic SOC, in which not a single task was accepted in

2013. It looks like this topic is too narrow. It would sound logical to incorporate this topic into the topic USE with which it shares its interconnectedness with everyday life unlike other topics which are much more theory-based.

The topic ALG on the other hand seems to be too wide and should be divided into subtopics. There are more possibilities for this subdivision:

- classification according to skills needed for task solution (e.g. algorithm design, error debugging and correction, search for output state, search for initial input state before algorithm implementation, exploration of algorithm universality, feasibility, selection of the most effective algorithm)
- classification according to extent of formalization (procedures in everyday life situations, algorithms using program structures, specialized algorithms for specific classes of situations, algorithms by design paradigms, algorithms known from theoretical informatics etc.)
- classification according to type of algorithm (sorting, searching algorithms …)

The fact that many authors used their own categories related to mathematical or logical fundaments of informatics suggest that an additional useful topic LOG (MAT) – constraints for making decisions, simple predicate logic, combinatorics, binary systems) should be introduced. The topic PUZ could then focus on problem solving, games and labyrinths, comprehension of rules, game strategies.

5 Digital Literacy in Tasks

The topic USE, which covers everyday use of ICT, digital literacy, user-centered approach, use of applications, is an integral part of compulsory curriculum in many countries. Blaho states that the field of ICT is often understood as mastering technology, as initial stage to informatics [9]. Schubert and Schwill regard ICT education as a framework of basic education in the area of informatics, communication and information technology [12]. If tasks connected to use of computer applications are included into contests like Bebras, the contest draws nearer to school curriculum and is more easily acceptable by schools and teachers as it meets the general public understanding of informatics as something connected to use of computers.

How do pupils perceive an informatics contest? A survey was carried out in December 2012 in the Czech national round of the Bebras contest. The questionnaire was answered by 13 %, i.e. 3500 contestants. Among other questions the respondents were given the opportunity to express their opinions on the contest. Apart from the expected classes of responses evaluating the contest (from “amazing” to “horrible”), comments on the respondent’s performance (“why didn’t I manage?”) and technical questions (“when will the lists of the best contestants be published?”), we could often come across opinions in which the contestants (especially from upper secondary schools) claimed the contest was not too much about informatics. Some of the participants’ comments follow:

- “More questions from informatics next time, please.”
- “This had nothing to do with Informatics!!!”

- “Why is this called a contest in informatics if only 2 (out of 15) tasks are about computers?”
- “The questions should focus on IT much more. These questions seemed to be from mathematics.”
- “I wonder what the contest questions have to do with informatics. Maybe nothing at all?”
- “The fact that one answers on a computer doesn’t make this an informatics contest.” [13].

Some of the comments imply that pupils believe the contest is called an informatics contest as it is sat at a computer. These pupils see a distinction between informatics and computers. However, the comments also imply that either the informatics tasks are too artificial or distant from everyday life, or that pupils’ conception they acquire at schools of what informatics is, of its basic concepts and types of problems it solves is wrong. It is also possible that pupils are not used to be solving difficult tasks on computers for whose solution they would have to apply reasoning, logic, make decision, and formalize their answers. These skills are never needed when merely running applications. Then they conclude that problems requiring thinking and reasoning are problems from the realm of mathematics or logic [14].

Tasks from the topic USE are problematic as they must be anchored in everyday life. Real-life situations connected to digital literacy often ask for experience, pre-knowledge or general knowledge and can rarely be solved merely by reasoning or computational thinking. This results in a clash of these tasks with one of the above mentioned criteria of Dagienė and Futschek [6], namely that no task can rely on pre-knowledge of details of specific IT systems. If this much needed criterion is used mechanically, many interesting questions may be rejected. In consequence authors might prefer not to propose tasks in this category for uncertainty and fear of their rejection. In this point of view it is easier and safer to prepare a theory-based task and thus avoid the risk of contact with everyday life situation. As the number of proposals in this topis is very low, the problem of application of this criterion is real.

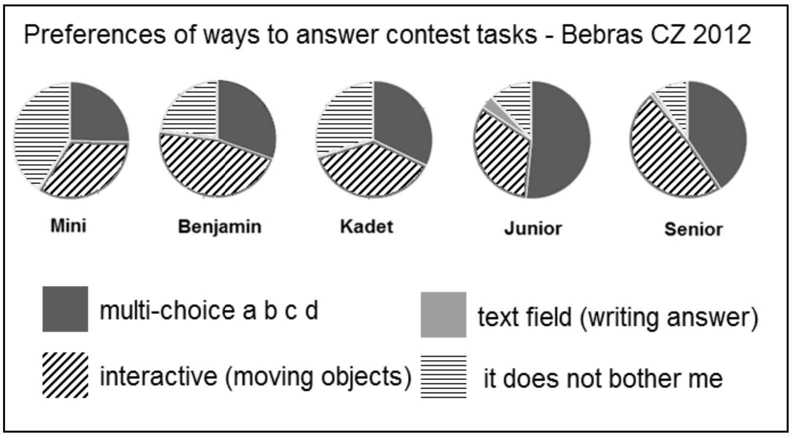


Fig. 1. Preference in types of answers for different age categories

Analogically tasks from the topic SOC related to legislation often face the situation when the expected (moral) practice of a person does not correspond lo legislation. Moreover, legislation may vary from one country to another and also within one country in some period of time. The reason for rejecting SOC tasks is often precariousness as the author's proposed correct answer may not be correct in other countries or may not be the only possible answer in different interpretations.

Especially because of tasks from the topic USE, which are perceived as crucially important in some of the participating countries and which not included in tests in sufficient numbers, the pre-knowledge criterion should be reformulated in such a way that tasks still do not require pre-knowledge of specific software applications but allow use of situations from work with software commonly used.

6 Interactivity of Tasks

Dagienė and Futschek also demand that tasks "*should* have interactive elements (simulations, solving activities, etc.)" [6]. We investigated how interactivity of a task contributes to its attractiveness.

In the questionnaire from 2012, the contestants also indicated their preferences in tasks (Fig. 1). Our analysis of their answers shows that fewer than one half of the contestants decidedly prefer interactive tasks. Interactive tasks are not even the most popular task type in all categories [13].

In the same 2012 questionnaire, repeated in 2013, we also asked about popularity and difficulty of the used tasks. The contestants were asked to choose from a list of tasks the one they found most interesting and the one they found easiest. With two exceptions, the most interesting task was one of the interactive tasks in all 10 categories. This shows that interactivity, if associated with a specific task, is evaluated differently than interactivity in general.

Then we compared the ratio of interactive and other tasks in the answers. This showed that, with the exception of the category Mini for primary schools, interactive tasks were seen as much more attractive and therefore also easier. However, this was not true universally; interactive tasks requiring work with keyboard (e.g. tasks simulating the tool Find/Replace) were much less interesting than graphic drag and drop tasks or tasks using mouse clicking. Thus we conclude that interactivity is attractive because of its graphic component and manipulation with the mouse.

The reason why interactive tasks were not so favourably accepted in the youngest category Mini might be that more than one half of all tasks in this category were interactive. This inflation of interactivity might have caused loss of their attractiveness.

One must stress that interactivity may substantially change character of the task. An experiment was carried in the national round in 2012 in which we prepared the same task both as an interactive and multiple choice task. While the multiple choice task was evaluated as average (the seventh most often selected as the most interesting task out of 15), its interactive variant was selected as the most interesting.

The possibility to manipulate or simulate in the computer environment may affect how pupils solve the task. They may be diverted from thinking to experimenting, to error and trial strategy in which they prefer cognitively less demanding methods. This might be one of the causes of popularity of interactive tasks. One must always consider carefully whether the demand that a task be interactive does not make the task much easier, of lower quality.

Attractiveness of interactive tasks may also result in a situation in which pupils pay more attention and spend more time solving interactive tasks than multiple choice tasks. This could then mean that interactive tasks are solved correctly by more contestants due to this extra attention. We wanted to verify this hypothesis through the database of 2013 national round. There were 34 454 contestants. We analysed 13 interactive and 57 non-interactive tasks. The outcome of this analysis does not verify the hypothesis. The contestants spent on average 2.8 times shorter time solving one interactive than non-interactive task.

7 Assignment and Motivation in Tasks

Dagiene and Futschek also mention that the story of the task plays a crucial role in contestnats' motivation [6]. Posing questions and tasks for the Beaver contest is often approached as "dressing a CS task into an ICT attire". In the pool of proposals from previous years, one can discern several approaches to posing tasks:

– finding some classical informatics task and giving it an attire of a simple story, most often using the fairytale character of a Beaver-moderator, who introduces the problem
– taking a task which is primarily about computer science and searching for an everyday life situation or a situation which somebody could imagine as real-world that corresponds to the original task problem
– starting from application of informatics in another discipline (mathematics, physics, biology) and constructing a story with a real-world situation which illustrates use of some informatics concept or principle used for its solution
– observing the world around you (or your teaching) and letting it inspire you to formulation of a task (usually from the topic USE).
– starting from a practical task drilling the skill to use some application and posing a multiple choice question. Changing the task from the instructional or practical (do this, create this) to situational (a situation is described and a question is added).

The character of Beaver (beaver goes to school, beaver drives a car) is very frequent in the pool of proposals. This personification of beaver brings paradoxical situations. In some tasks e.g. the beaver eats meat or bridges are constructed for this water animal to get from one river bank to another. Tasks introduced through beaver "stories" sometimes do not sound very real, or their solution does not sound practical. Contestants' especially upper secondary school contestants') comments on tasks such as "beaver did, beaver went, …" were full of irony. Older contestants found tasks with people working on computer in place of beavers much more acceptable. However, some contestants in categories for pupils younger than 13 claimed they liked the beaver and demanded that "more tasks with the beaver" be included. Therefore the authors of proposals should always bear in mind that the effect of motivating contestants by this fairytale character changes with age.

When making proposals, authors also have to face the risks that their tasks may favour pupils who have had the opportunity to memorize the solving procedure. There are some tasks that make use of a typical procedure characteristic for the particular type of task and a mere application of the procedure leads to the correct solution. This discriminates contestants who have not come across this type of task at school and have not had the chance to memorize its solution. For example any square grid path problem can be solved very easily if the contestant is familiar with the basic principle that the number of possible paths to a given junction equals to the sum of possible paths for all neighbouring previous junctions. If the contestant does not know this principle it is very hard to discover it in the 3 minute time limit. Any task based on a typical informatics problem described in literature bears the risk that some of the contestants will be familiar with its solving method.

Somewhat problematic is in our opinion the demand on political correctness of tasks, i.e. the criterion that "Good tasks contain no gender, racial or religion stereotypes." [6]. Not underestimating the import of this proclamation, our experience from totalitarian communist times makes us very cautious when putting this criterion for contest tasks into practice. If e.g. a task in which a boy from Germany gives a piece of ham to a boy from the Czech Republic is rejected because it could offend somebody's religious beliefs, it may be seen as limiting authors' freedom. We could then also ask whether the danger of the impact gender stereotype could have on success rate of girls in solving a task would not result in rejection of a task in which girls string beads.

Responsibility for creation of a set of contest tasks for national rounds is in the hands of national authorities and it should be their role to review the pool of proposals sensitively and make decisions about the tasks' political correctness in the context of their own national culture, and only after if not acceptable in their cultural context reject them or modify them. We think that such reviewing process on international level is dangerous.

8 Quality of Wrong Choices

Quality of the task assignment is one of the criteria of good tasks to which much attention is paid by Dagienė and Futschek: this involves wording of the question, correct answer and its justification. Rules for good assignment include e.g. the rule not to use negation in questions as the contestant often overlook it or have problems with its logic. Our experience of the process of task development from an international workshop and from analysis of a pool of proposals suggests that it is equally as important to pay attention to formulation of the incorrect answers in multiple-choice tasks as their quality may considerably influence task difficulty. The contestant chooses from four choices and if some of these can be eliminated without actually understanding the task, the chance that they will guess the right answer grows.

This can happen if

1. the set of choice answers is badly constructed
2. it pays off to go through all the choice answers and test them in the assignment rather than solve the problem
3. a common mistake missing among offered answers warns the contestant
4. there are weak choices just to make up the needed number of choices

Ad 1. Some sets of choice answers are constructed by deriving wrong variants from the correct answer by minor modifications. The contestant, if experienced in taking tests, sees that one of the offered answers shares characteristics with each of the other variants, realizes it is the source variant and marks it as the correct one without actually understanding the issue.

Let us illustrate the point:

Question: What will be the output of this programme?
(we do not need to show the programme here)

a) Result: c=20
b) Result: c=36
c) Result: c=40
d) Result: c=20+c

Without having to read the task assignment we can infer that the right answer is a). The numerical result of the calculation is the same as in answer d) and we can expect answers b), c) to be consequences of a mistake in the calculation. Answer d) involves a mistake of a different type. Although it is not worthless if the pupil finds the right answer by such logical reasoning, it has nothing to do with his/her knowledge of informatics, only with his/her knowledge of sitting tests.

Ad 2. If the author of a task wants the contestant to solve the task by thinking, the choice answers should be in such a form that it does not pay off to test the choice answers in the assignment. In ideal case reading of the assignment guides the contestant to knowledge that directly points at the right answer whereas testing all variants is very time demanding with a considerable risk of making a mistake.

The following set of choice answers Dice (the assignment is not important) is an example of a set whose author avoided mistakes from 1 and 2:

a) draw_2A, draw_2, turn_90, draw_2
b) draw_2, turn_90, draw_2, draw_2A
c) draw_2A, turn_90, draw_2, draw_1
d) draw_2, draw_2A, turn_90, draw_2

This set of choices does not allow us to guess the correct answer without understanding the assignment and having the required knowledge. And it would be too time demanding to test all the choices in the task assignment.

Ad 3. Especially in case that a task easily leads to various erroneous results (e.g. calculation using an algorithm), it is crucial to ensure that the set of choice answers includes the most likely mistakes. If the contestant makes a mistake and gets a result that is not among the choice answers, he/she is alerted to the fact that his/her solution is wrong. Thus the contest is less regular as some contestants being wrong get a hint in this form while others do not. Apart from selecting choice answers carefully, authors should also modify the task in such a way that their solvers do not make too many different mistakes leading to too many different results.

Ad 4. If a contestant does not know the correct answer, he/she is likely to be trying to guess it. Then existence of one or more obviously nonsensical answers that can be eliminated at once without any knowledge of the topic considerably increases the contestant's chance to guess the correct answer. E.g. in case of questions requiring a yes/no answer authors should ask about two phenomena simultaneously so that the number of possible answers is extended (e.g. yes,yes/yes,no/no,yes/no,no).

The following is our proposal of a criterion for good multiple-choice task:

- the problem in the task should offer a reasonable number of possible answers (neither too few, neither too many)
- the incorrect answers should represent all the typical mistakes the contestant may make while solving the task
- the correct answer should not stand out (by its length, choice of words etc.)
- every task should have a set of choice answers comparable in quality

9 Conclusion

Criteria for a good task for international informatics competition proved to be very useful as they guide authors of proposals to posing more usable tasks.

- Based on our findings we recommend that task topics be reorganized in such a way that they become more useful in defining the content of national contests and that they are representative of the field of informatics. We recommend that the topic SOC be integrated in the topic USE and the topic ALG be split into additional criteria. We recommend that the topic LOG (MAT) be added.
- The contest becomes more comprehensible if contestants come across tasks that they perceive as work with computer. That is why also tasks from everyday work with computers should be included and the criterion demanding elimination of preknowledge should be reconsidered as it may sometimes be counterproductive when applied in the topic USE. This might be subject to discussion.
- We propose that the criterion of suitable answers in multiple choice tasks be added; attention must be paid to selection of the wrong variants as they affect quality of the whole task.
- Interactivity of tasks makes the contest more attractive if interactivity means manipulation with mouse. On the one hand interactivity makes the contest more appealing, on the other hand it may affect the character and difficulty of the task, which must always be taken into account.
- If tasks stem out from real life situations, it will be appreaciated by older contestants. Younger will enjoy motivation through the fairytale character of Beaver.

In conclusion we would like to stress the most important aspect of the contest: the created solid and growing community of authors and researchers can significantly contribute to future coordinated process of inclusion of informatics content into school curricula in many countries.

References

1. Dagienė, V.: The Bebras Contest on Informatics and Computer Literacy – Students Drive to Science Education. In: Joint Open and Working IFIP Conference, ICT and Learning for the Net Generation, Kuala Lumpur, pp. 214–223 (2008)
2. Dagienė, V., Futschek, G.: Knowledge construction in the Bebras problem solving contest. In: Kynigos, C., Clayson, J., Yiannoutsou, N. (eds.) Constructionism: Theory, Practice and Impact, pp. 678–680, The Educational Technology Lab, Univ. of Athens, Athens (2012)
3. Wollring, B.: Kennzeichnung von Lernumgebungen für den Mathematikunterricht in der Grundschule. In: Forschergruppe, K. (ed.) Lernumgebungen auf dem Prüfstand. Bericht 2 der Kasseler Forschergruppe Empirische Bildungsforschung Lehren – Lernen – Literacy, pp. 9–26. Kassel University Press GmbH, Kassel (2008)
4. Hejný, M.: Vyučování matematice na 1. stupni ZŠ orientované na budování schémat: Aritmetika. PedF UK, Praha (2013)
5. Vaníček, J.: Potenciální a skutečný dopad informatické soutěže do změn kurikula ICT v České republice. In: Kalaš, I. (ed.) DidInfo´2012, Univerzita Mateja Béla, Banská Bystrica, pp. 15–24 (2012) ISBN 978-80-557-0342-8
6. Dagienė, V., Futschek, G.: Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks. In: Mittermeir, R.T., Sysło, M.M. (eds.) ISSEP 2008. LNCS, vol. 5090, pp. 19–30. Springer, Heidelberg (2008)
7. Barr, V., Stephenson, C.: Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community? ACM Inroads 2(1) (2011)
8. Sysło, M.M., Kwiatkowska, A.B.: Informatics for all high school students: a computational thinking approach. In: Diethelm, I., Mittermeir, R.T. (eds.) ISSEP 2013. LNCS, vol. 7780, pp. 43–56. Springer, Heidelberg (2013)
9. Blaho, A., Salanci, L.: Informatics in Primary School: Principles and Experience. In: Kalaš, I., Mittermeir, R.T. (eds.) ISSEP 2011. LNCS, vol. 7013, pp. 129–142. Springer, Heidelberg (2011)
10. Computing. Programmes of study for Key Stages 1 - 4. National curriculum in England. Computing at School Working Group (2013), `http://media.education.gov.uk/assets/files/pdf/c/computing%2004-02-13_001.pdf`
11. Kalaš, I., Tomcsányiová, M.: Students' Attitude to Programming in Modern Informatics. Informática na Educação: Teoria & Prática 12(1), 127–135 (2009)
12. Schubert, S., Schwill, A.: Didaktik der Informatik, 2nd edn. Spektrum Akademischer Verlag, Heidelberg (2011)
13. Vaníček, J.: Searching for CS Tasks in ICT Curricula at Lower Secondary School Level. In: Reynolds, N., Webb, M., Syslo, M., Dagiene, V. (eds.) Learning While We are Connected, Proceedings of 10th IFIP WCCE, vol. 3, pp. 119–120, Toruń: Uniwersytet Mikolaja Kopernika (2013)
14. Vaníček, J.: Computer Science Tasks and Topics as a Part of ICT Curricula in the Eyes of Pupils and Teachers. Journal of Technology and Information Education 5(1), 67–74 (2013)

Big Data – Challenges for Computer Science Education

Andreas Grillenberger and Ralf Romeike

Friedrich–Alexander–Universität Erlangen–Nürnberg (FAU)
Department of Computer Science, Computing Education Research Group
Martensstr. 3, 91058 Erlangen, Germany
{andreas.grillenberger,ralf.romeike}@fau.de

Abstract. Data processing is a central topic of computer science and hence also in secondary computer science education, which includes strategies for storing, managing and retrieving data. In the context of Big Data, this field changes tremendously: established ideas, such as avoiding redundancies and storing data in a persistent and consistent way, are dropped in order to speed up the access to distributed stored data as well as its availability. Furthermore, with the rapidly growing impact of data processing on everyone's daily life, computer science education needs to address these aspects as well as their social and ethical implications, such as privacy issues.

This paper points out the major challenges that arise from the outlined developments by evaluating whether database concepts and examples commonly used in CS education need to be updated.

Keywords: Big Data, NoSQL, Data Management, Databases, Data Analysis, Data Privacy, Challenges.

1 Introduction

The concept of data processing is central to computer science and hence to computer science education. Not only manipulating data through programming, but also the efficient storage, management and retrieval of data are seen as central aspects. These topics are strongly affected by Big Data, a phenomenon that arose in recent years and which introduced several new innovations. For example, established concepts of data management, such as avoiding redundancies and inconsistencies by saving data in normalized relational database management systems (RDBMS), are dropped. Instead, newer database management systems (DBMS), like the NoSQL databases, are optimized for performance and distributed storage [11].

Other innovations come from the way Big Data affects everybody's life, as Big Data is fundamental to the functionality of various popular applications: search engines, translation tools, social media (e.g. for friend finders), online shops (e.g. for product recommendations) and so on.

In this paper we will point out major challenges computer science education will have to deal with, when considering the new aspects arising from the outlined developments into teaching. First, Big Data is discussed in terms of its innovations to data management. Then, we will provide the educational context by analyzing the state of research on databases and Big Data in computer science education. On this basis, we will derive and discuss the relevance of changes arising from these developments and we will describe the challenges within computer science education.

2 Managing Big Data

The term Big Data describes the management and analysis of large amounts of data with high complexity and varying structure. When dealing with Big Data, typical database systems are reaching their limit: even if they can handle the large ***volume*** of data, there will remain at least two even more complex problems. The high changing rates of these data (***velocity***) require highly responsible databases, ideally without any blocking operations. Additionally, the ***varying structure*** of data prevents database administrators of defining a data schema. Such a schema is needed for ensuring data consistency when using common relational databases (RDBMS). As Laney [18] summarizes, Big Data can be characterized by the three Vs "volume", "velocity" and "variety" (cf. fig. 1).

These challenges when using the more or less standard RDBMS led to the development of non-relational database management systems, commonly known as NoSQL databases. This term is meanwhile used as a generic name for all non-relational databases and is interpreted as "not only SQL"[1] [11]. The main characteristics of NoSQL databases are [11]:

- a non-relational data model,
- distributed and horizontally scalable,
- schema-free or only with weak restrictions on schema.

Also, in contrary to traditional databases, the NoSQL databases do not guarantee the ACID properties (**A**tomicity, **C**onsistency, **I**solation, **D**urability) [12]. Instead, they are described as **B**asically **A**vailable, **S**oft-state, **E**ventually consistent (BASE) [11]. Such databases offer fast access to and high availability of data, but cannot guarantee failure safety, as data are only written to permanent storage periodically. Also, consistency of data cannot be ensured, because typically NoSQL databases do not enforce a defined data schema in order to reach high performance and partition tolerance. According to the CAP theorem [4], it is not possible to guarantee the three properties availability, consistency and partition tolerance at the same time. Only two of these three characteristics can be ensured concurrently, cf. fig. 2. So, traditional RDBMS mainly focus on consistency and availability, and therefore need to drop partition tolerance—they are

[1] In the original work on the NoSQL database [23], the term NoSQL was used in the way "no usage of SQL", because this RDBMS did not support SQL.

described as consistent-available (CA) according to the CAP theorem. In contrast, the NoSQL databases are typically optimized for distributed databases with high performance and therefore must drop consistency (so they are described as available and partition-tolerant (AP)). The third type—consistent and partition-tolerant (CP)—is less widely spread and not associated to whether RDBMS or NoSQL, as both types can be found in this category. In contrast to these developments, the main topic in today's database education is teaching the concepts of (relational) database systems and the knowledge for dealing with such systems. In order to fill up this gap between CS and database education, the influence of these developments on CS education has to be discussed.

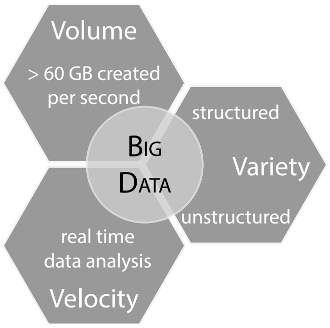


Fig. 1. Three Vs of Big Data

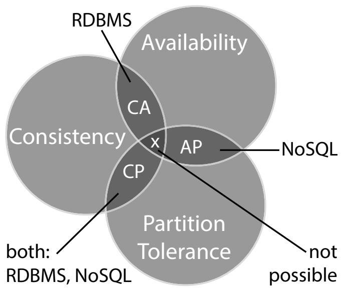


Fig. 2. CAP theorem

While the topic databases plays an important role in secondary school education, with the growing impact of data and its analysis this topic needs to be expanded. *Data management* describes the processes of storing and retrieving data from databases as well as planning, organizing and utilizing these methods [3]. This includes "practices that control, protect, deliver and enhance the value of data and information assets" [8] which can be summarized as data safety, data privacy and data analysis. Aspects of data management are often already considered in computer science curricula, e.g. under the term "information management" in the ACM K-12 curriculum [24]. So, data management offers the chance to bring these aspects together in one topic and to include several additional aspects.

3 Databases and Data Management in Computer Science Education

The topic databases, which comprises the efficient storage, management and retrieval of data, is central to secondary computer science education in several countries (e.g. Germany [16], Austria [6,7]). The ACM curriculum for K-12 computer science [24] considers data management in the context of information management: database systems, data modeling and the relational model, query languages, data mining, hypertext and hypermedia, digital libraries. The UNESCO/IFIP informatics curriculum for secondary schools [25] considers databases in two ways: Working with a database and designing a database. Numerous publications especially in the late 1980s / early 1990s, discuss the relevance of the topic for computer science at school (e.g. [26]). However, even though the importance and influence of databases and data management is growing rapidly, these topics were hardly discussed in computer science education within the last years: Antonitsch [1] proposes databases as a topic in the context of information retrieval by questioning the common approaches for database instruction. Other publications on these topics mostly propose and/or discuss tools for enhancing database education and in particular for supporting the teaching of SQL (for example [14], [20]). The shortage of research in this field may be explained by the few changes in the concepts of (relational) databases since they were established by Edgar F. Codd. The main changes affected platforms; these are relevant for the DBMS and for tools for using them, but not for the concepts of databases themselves or for educational purposes. In contrary to the discussion in the data management community, new models for database, like the NoSQL databases, have not yet been discussed in the context of database education at school. The same applies to higher education, where recently claims are made to consider the emergence of NoSQL databases in the curricula [19].

In addition to these aspects from computer science, CS education is also affected by social and ethical implications: while current teaching only considers data privacy and safety as marginal topics, mainly integrated with other topics like databases, nowadays the relevance of these aspects is changing tremendously. As new possibilities are opened up for dealing with large amounts of data and to gain new information by analyzing them, also new threats are emerging. Especially, there is a strong impact on data privacy and data security. Furthermore, everyone needs to deal with data in daily life, as the impact of data and data-driven applications is continuously increasing. Hence, there are clear differences between the requirements coming from daily life and current teaching practice.

4 Challenges for Computer Science Education

As pointed out in the previous section, a gap between curricula / teaching practice and the state-of-the-art of computer science emerges from the recent developments. Moreover, there are also recognizable differences between the demands of daily life and current CS education. In acknowledge to this issues, organizations such as the German Informatics Society start claiming that "Big Data is a topic for education" [13]. However, by considering Big Data for teaching, computer science education is faced with several challenges, which include revising the curricula for the topics databases and data management in a critical way. Key questions that need to be considered are:

– Which key concepts and principles are sustainable?
– Which topics may become outdated or need to be discussed in a different light in the future?

– Which aspects does Big Data add?
– What does everyone need to know about and for dealing with Big Data?

Therefore, we will hereafter discuss the major challenges, with which computer science education will have to deal when considering the new aspects coming from the described topics in teaching of data management. This analysis does not focus on a specific curriculum but applies generally to CS education, because while typical curricula strongly differ in many aspects, there is a clear agreement on important aspects in database education.

4.1 Discussing the Relevance of Database Concepts

There is agreement that general school education needs to emphasize persistent concepts over skills that are only usable for a short time span. For computer science, various sets of criteria for selecting topics with general educative value have been proposed. The most influential ones are the "Great Principles of Computing" by Denning [10] and the "Fundamental Ideas of Computer Science" by Schwill [21]. One typical criterion is, that topics for general education need to be relevant in various contexts. This criterion is named "horizontal criterion" by Schwill, "broadly influential" by Denning. It prevents special aspects, which are only relevant in few contexts, from being considered as relevant for general education. Another criterion points out that only those concepts shall be taught that are and will be historically relevant. Due to the recent developments in data management, e.g. Big Data, the concepts and principles currently taught in school are now challenged. For example, RDBMS use an explicit data schema [12], while NoSQL databases use an implicit one [11]. By forcing data into matching the data schema defined by the user/administrator, RDBMS ensure consistency. This approach leads to long outage times if the schema has to be modified, because then data has to be adopted to the new schema immediately. Since today data often vary in structure over time, this would happen too frequently. So, NoSQL databases prevent such outages by not-enforcing a schema (so they have only an implicit one), but therefore they must deal with inconsistent data. Hence, it has to be examined whether always enforcing a data schema holds for a key concept of databases.

Another challenge is the altered significance of redundancy. While in RDBMS data are typically stored in a in a normalized way in order to avoid redundancies [12], the concept of redundant storage is used intentionally in NoSQL databases [11] in order to store data in one place instead of spreading it over multiple tables. Hereby, reading data from NoSQL databases is accelerated, because in comparison with RDBMS no join operations are needed. Therefore, the challenge for computer science is to rethink the idea of redundancy from something that generally should be avoided to a method that is applied in order to achieve certain specific goals. While redundancy in the context of RDBMS has been only discussed in the context of database normalization, it should be discussed, if the concept of redundancy holds for a fundamental idea of data management.

Even the small selection of differing concepts between RDBMS and NoSQL databases described above, leads to important considerations, as differing aspects cast a new light on meeting the criteria for general educational topics: until Big Data became popular, it seemed obvious that the main concepts of relational databases are as well main concepts of databases in general, because there were mainly RDBMS. Nowadays, it is necessary to differentiate between relational and non-relational databases, so it is easily recognizable, that for example the concept of strictly normalizing data schemata is only fundamental to relational databases. Therefore, the concepts concerning databases in general, and not only concerning either RDBMS or NoSQL databases, have to be detected. When looking at the described fundamental ideas and great principles, it is obvious that only the concepts concerning the topic databases in general should be selected for teaching in order to ensure teaching of general educative topics.

In conclusion, current considerations on which concepts of databases should be part of computer science education need to be reconsidered in the light of Big Data and data management by posing the question *"Which concepts are fundamental to databases?"*.

4.2 Involving Big Data Examples into Teaching

Even though the intention to use examples of students' everyday life by discussing offline databases held true until recent years, in future web databases and large data collections will become more common to students and affect their daily life more than offline databases. Furthermore, such databases offer the chance to use public and up-to-date (open) data sources and hence open up the possibility for students to gain insight into data by using analysis methods themselves. At the moment, database education especially discusses offline databases, for example a member database of a sports club, of products the students (or fictive persons) bought, or of music groups [15]. Some main aspects of databases can hardly be demonstrated by such small examples, especially because some of them, like the product database, could be equally addressed in a spreadsheet application. Since small databases are often used as local copy on each computer, characteristics such as the multi-user capabilities of databases cannot be realized. Taking into account the impact of Big Data, this problem will be even intensified, because using small examples will not be possible for illustrating the main characteristics of Big Data and data analysis, because they are particularly based on the high number of available data. Especially, by using small data sets it is not possible to obtain satisfying analysis results of statistical relevance. According to the mathematical law of large numbers, scattering is minimized when analyzing large amounts of measures (data), while results are strongly distorted when only analyzing small amounts of data.

Also, the changes in concepts of databases strongly affect selection of examples for teaching. While typical examples in database education are often chosen in order to discuss databases considering the relational data model, normalization, consistency and so on, these criteria for selection will clearly differ when considering Big Data as topic for education. In the future the examples discussed in class need to be able to clarify the possibilities of data analysis in order to enable students to recognize main concepts and methods of data analysis.

Summarizing, small data sets will not be sufficient in future data management education anymore; instead students need to be enabled to work with large amounts of data themselves, so that they can recognize the advantages of databases and data management. This also implies storing the data on a central server, because of the quantity of data. Therefore, two main functionalities of databases are emphasized implicitly: storing data in a central place and accessing them concurrently with many clients [12]. This results in the challenge to determine and open up the possibilities to use the available data, mainly coming from open data projects, in class. Hereby, teachers are enabled to provide examples that better fit the usage of databases in economy, but at the same time, students are enabled to analyze data themselves and so recognize which kind of data may be obtained by combining different data sets.

4.3 Teaching Data Analysis for Understanding Data Mining

Nowadays, data are a valuable resource. They often contain more information than visible at a first glance. Hence, data mining is one of the most rapidly growing business factors as well as a threat to data privacy. In order to provide students with an understanding of the value of data as well as the possibilities and threats of collecting, processing and evaluating personal data, this aspect of data management should be added to the curriculum. According to the World Economic Forum, "personal data will be the new 'oil' - a valuable resource of the 21st century" [27]. Just like any other resource, raw data must be made usable before being able to benefit from them. For data, this is done by analysis: unnecessary data are stripped, remaining data are aggregated, combined with other data, and so on until a final result is usable. Yet in typical computer science education examples, there is an emphasis on structuring data, storing them in relational databases and retrieving them efficiently. However, only limited new information is extracted from these data. A reason for this may be that data are mainly retrieved from the database in the same way they were stored before. Often these data are supplemented with results of aggregate functions, but they only add few new information. In contrast, data analysis methods offer the chance to discover more information and coherences by using the main methods for data analysis [17], like:

- *Clustering* sorts new data sets into related groups (clusters) according to determine similarities. For example, in social networks clusters of users sharing the same interests may be determined in order to propose interesting groups to a user.
- *Classification* as well as clustering aims for finding patterns in the data. However, by classification methods the categories are defined beforehand. Emphasis is put on determining characteristics and attributes of these categories.

- *Association* is used for determining interdependent occurrences of certain events. This allows for finding inferences for interdependencies between data sets. Such inferences are formulated as "if-then-relations" and can be used in order to predict e.g. future behavior after occurrences of certain events.

Summarizing, a challenge for computer science education is to enable students to understand data mining processes as well as using data analysis methods in order to acquire new information. Therefore, concepts and topics of data analysis need to be analyzed with respect to the criteria for selecting topics for general education described above [10], [21].

4.4 Changes in the Relevance of Data Modeling

Key aspects of data analysis are the methods for structuring data, for interpreting them and for recognizing their important aspects. Especially for structuring, another fundamental idea of computer science is involved: (data) modeling. In current teaching, static data modeling is typically applied in order to define data structures before inserting the data into a database. In the future, the objective of modeling will change: When working with data in order to analyze them, typically prestructured data will be used, for example coming from Open Data projects. Even if they are not structured explicitly, at least an implicit structure is defined by the way they are stored. Therefore, structuring such data by hand is not necessary anymore. But when using data analysis methods, static data modeling will still be relevant for providing an overview over data sets. Also, especially when combining different data sets, it will be necessary to visualize data structures as in most cases these data sets are not aligned for being combined, because at the moment of structuring, future uses are often unknown.

Additionally, when working with NoSQL databases, the relevance of static data modeling will decrease, because these databases do not enforce a data schema. Instead, data is stored less spread, but rather coherent in ideally only one database table/collection often structured exactly the way they are received from the data sources. This is the case, as Big Data does not fit into a defined schema, because of its varying structures. In particular, by saving data in a coherent way retrieving them from the database is accelerated.

Not only static data modeling, but also dynamic modeling techniques are involved when discussing data analysis. In particular, as data analysis consists of complex processes and sequences of operations, the analysis process can clearly be visualized and clarified by using sequence diagrams. By discussing data analysis in this way, the recurrence of the typical methods and operations of data analysis is clearly visible. Therefore, these techniques would be emphasized in comparison to only using a data analysis tool without discussing the methods and operations used. In addition, sequence modeling also has a strong relevance in daily life, as visualizing and discussing sequences of operations is necessary in various fields, not only in computer science. Therefore, emphasizing sequence modeling in data management education also raises its general educational value.

Summarizing, this leads to another challenge: At the moment, data modeling is the main modeling technique when dealing with data in CS education. When considering Big Data and data analysis, on the one hand the usage of data modeling changes from mainly structuring data into also visualizing prestructured data. On the other hand, also process / sequence modeling will increase in importance for planning data analysis processes.

4.5 Sharpening the View on Data Privacy

Since Big Data strongly affects the topic "data privacy", it is not surprising that several publications discuss aspects of Big Data and data analysis in this context. These topics are commonly taught in the context of database education. For example, the "simulation game data privacy"[2] is a learning environment in which students may discover threats for data privacy on their own. Such material typically cover fundamental data privacy problems. But problems caused by Big Data, which strongly differ, generally are not yet taken into consideration. Therefore, data implicitly given to the vendors, like when clicking on a web page link, increases in value. In this context, actual material on data privacy needs to be reviewed and revised in order to cover new challenges for this topic.

Also, new possibilities raise new threats in this field. Today, data is captured continuously in daily life, for example obviously by smartphones and tablets, but also in a more hidden way by cars, smart electricity meters, and such. With the ability to analyze such large amounts of data, large parts of daily life may be reconstructed and users profiles may be generated. By combining information from different sources, gathering private data like friendship relations, hobbies, habits and so on, is possible. This applies not only for obviously private data, like friendship relations, or for personal data like name and birth date, but also for data that seem harmless at the first glance: particularly meta-data, like date and time a text message was sent on or the position a photo was taken at.

When discussing the threats for data privacy, methods for preventing such analysis of own data should be pointed out. As with modern devices, (web) applications and services, data are often collected in a hidden way, a first challenge is to recognize that a specific application/service/device might collect data. Even when suspecting such a collection of data, it is hardly possible to prevent it other than not using the application. Although, in certain cases it is possible to distort data analysis, for example by using pseudonyms or by entering incorrect data. Thereby, no correct conclusions can be drawn from the data analysis, when intentionally wrong data are spread in order to prevent differing between genuine and false data, or when consistently using pseudonyms. But even with pseudonymization, it cannot be ensured that no private data may be recognized by data analysis: for example, when AOL released a strongly pseudonymized set of search results in 2006, it was possible to discover contact data of persons contained in this data set by using simple analysis methods [2].

[2] `http://www.informatik-im-kontext.de/index.php/entwuerfe/planspiel-datenschutz-2-0`, in German (last retrieved: 20th April 2014).

As discussed in several publications, CS education also should raise students awareness on data protection and privacy, so these topics are part of various curricula and educational standards (like the CSTA K-12 Computer Science Standards [22] or the German Educational Standards for Computer Science [5]). In current curricula, these aspects are often integrated with other topics, especially databases, and thus mentioned only marginally. This cannot satisfy the relevance of these topics, neither at the moment nor in the future when privacy problems will become even more important because of Big Data. This leads to the challenge that the relevance of the topic data privacy as well as material for teaching this topic at school need to be revised in order to fit new requirements.

5 Discussion

It is a grand challenge for computer science education on the one hand to aim for teaching of long lasting concepts and principles of computer science, and on the other hand to be consistent with current scientific developments. In the field of data management, traditional concepts and teaching examples are challenged by recent developments, such as Big Data, NoSQL and Open Data. By considering these in data management education, various new and motivating aspects can be found for teaching—with strong references to daily life of the students.

Using these new possibilities in data management education will clearly change the face of this topic, which is currently limited to concepts of databases and their application. In contrast, new concepts call for a stronger emphasis on broader aspects of data management, e.g. still data storage (as in databases), but in combination with data usage and data analysis. This will provide even more relations to students' daily life, especially by being educated in dealing with their own (personal) data. In order to take the chance of involving such new possibilities, several changes in CS education curricula will be required. Therefore, it is necessary to take a more in-depth look on these aspects for being able to evaluate which changes are promising and which are not.

Also, discussing Big Data in class allows a view on modern or newly emerging professions, like the "data scientist" [9], which involves aspects from computer science but also from mathematics and statistics. By discussing such fields of applied computer science, also the view on the field of CS may be sharpened: a typical bias on computer science is, that it mainly consists of programming. But by having a look on such interdisciplinary fields like data analysis, it is obvious that computer science involves more aspects than writing programs.

Nowadays everybody is affected by this field of computer science, because Big Data is being collected and analyzed in many different systems of daily use. Therefore, considering Big Data as topic in computer science education also enables students to form better founded opinions on actual topics in society, like on (early) data retention or on programs of intelligence services like the NSA's PRISM program. While it is commonly known, that in such projects huge amounts of data are collected, often the consequences remain vague. At this point, computer science education will help understanding possibilities of analyzing such (big) data, as well as the related consequences.

References

1. Antonitsch, P.K.: Databases as a Tool of General Education. In: Mittermeir, R.T. (ed.) ISSEP 2006. LNCS, vol. 4226, pp. 59–70. Springer, Heidelberg (2006)
2. Barbaro, M., Zeller, Jr., T.: A Face Is Exposed for AOL Searcher No. 4417749. (August 2006), `http://www.nytimes.com/2006/08/09/technology/09aol.html`
3. Bodendorf, F.: Daten- und Wissensmanagement [Data and Knowledge Management]. Springer-Lehrbuch, Springer (2005)
4. Brewer, E.: CAP twelve years later: How the "rules" have changed. Computer 45(2), 23–29 (2012)
5. Brinda, T., Puhlmann, H., Schulte, C.: Bridging ICT and CS: Educational Standards for Computer Science in Lower Secondary Education. In: Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2009. ITiCSE '09, pp. 288–292. ACM, New York (2009)
6. Bundesministerium für Unterricht, Kultus und Kultur: Lehrplan der AHS-Oberstufe, Fach Informatik, Pflichtgegenstände [Curriculum for the Austrian Secondary School, Subject "Informatics", compulsory topics], `http://www.bmukk.gv.at/medienpool/11866/lp_neu_ahs_21.pdf`
7. Bundesministerium für Unterricht, Kultus und Kultur: Lehrplan der AHS-Oberstufe, Fach Informatik, Wahlpflichtgegenstände [Curriculum for the Austrian Secondary School, Subject "Informatics", compulsory topics], `http://www.bmukk.gv.at/medienpool/11876/lp_neu_ahs_21.pdf`
8. DAMA International: The DAMA Guide to the Data Management Body of Knowledge - DAMA-DMBOK. Technics Publications, LLC, USA (2009)
9. Davenport, T.H., Patil, D.J.: Data scientist: the sexiest job of the 21st century. Harvard Business Review 90(10), 70–77 (2012)
10. Denning, P.J.: Great Principles of Computing. Commun. ACM 46(11), 15–20 (2003)
11. Edlich, S., Friedland, A., Hampe, J., Brauer, B., Brückner, M.: NoSQL [in German]. Hanser, Carl Gmbh + Co. (2011)
12. Elmasri, R., Navathe, S.: Fundamentals of Database Systems. Addison Wesley Publishing Company Incorporated (2011)
13. GI (Gesellschaft für Informatik e.V.): Handlungsempfehlungen an die politischen Akteure [Recommendations for action at the Political Actors], Big Data Days (2013), `http://www.gi.de/fileadmin/redaktion/Hauptstadtbuero/Handlungsempfehlungen.pdf`
14. Grillenberger, A., Brinda, T.: eledSQL: A New Web-based Learning Environment for Teaching Databases and SQL at Secondary School Level. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education, WiPSCE 2012, pp. 101–104. ACM, New York (2012)
15. ISB (Staatsinstitut für Schulqualität und Bildungsforschung): Informatik am Naturwissenschaftlich-technologischen Gymnasium, Jahrgangsstufe 9 (Handreichung) [Informatics at the Scientifiy-Technological Secondary School in Bavaria (Recommendations)] (2007)
16. ISB (Staatsinstitut für Schulqualität und Bildungsforschung): Lehrplan für das Gymnasium in Bayern, Fach Natur und Technik [Curriculum for the Bavarian Secondary School, Subject "Informatics", Scientific & Technical Branch] (2009)
17. Kemper, A., Eickler, A.: Datenbanksysteme: eine Einführung [Database Systems: An Introduction]. Oldenbourg (2006)

18. Laney, D.: 3D Data Management: Controlling Data Volume, Velocity, and Variety. Tech. rep., META Group (February 2001)
19. Luukkainen, M., Vihavainen, A., Vikberg, T.: Three Years of Design-based Research to Reform a Software Engineering Curriculum. In: Proceedings of the 13th Annual Conference on Information Technology Education, SIGITE 2012, pp. 209–214. ACM, New York (2012)
20. Sadiq, S., Orlowska, M., Sadiq, W., Lin, J.: SQLator: An Online SQL Learning Workbench. In: Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2004, pp. 223–227. ACM, New York (2004)
21. Schwill, A.: Fundamental ideas of computer science. Bull. European Assoc. for Theoretical Computer Science 53 (1994)
22. Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., Boucher Owens, B., Stephenson, C., Verno, A.: K–12 Computer Science Standards. Computer Science Teachers Association, Association for Computing Machinery (2011)
23. Strozzi, C.: NoSQL: a non-SQL RDBMS (1998), `http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%page`
24. Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., Verno, A.: A Model Curriculum for K–12 Computer Science. Final Report of the ACM K-12 Task Force Curriculum Committee, CSTA (2003), `https://www.acm.org/education/education/curric_vols/k12final1022.pdf`
25. UNESCO/IFIP: Information and Communication Technology in Secondary Education (2000), `http://wwwedu.ge.ch/cptic/prospective/projets/unesco/en/`
26. Witten, H.: Datenbanken - (k)ein Thema im Informatikunterricht? [Databases - No Topic in Computer Science Education?]. LOG IN 2 (1994)
27. World Economic Forum: Personal Data: The Emergence of a New Asset Class (2011), `http://www3.weforum.org/docs/WEF_ITTC_PersonalDataNewAsset_Report_2011.pdf`

All electronic sources were at last retrieved on 17th April 2014.

Analysis of Computer Science Education in Venezuela Using the Darmstadt Model

Nubia Alejandra Fecht and Ira Diethelm

Carl von Ossietzky University, Computer Science Education,
26111 Oldenburg, Germany
{nubia.alejandra.fecht,ira.diethelm}@uni-oldenburg.de

Abstract. From time to time societies have to change their educational system in order to adapt to new social or economic conditions. This is especially necessary for teaching the subjects CSE and ICT because their scientific background alters very quickly and their relevance grows steadily. Venezuela had to meet this challenge in a way that cannot really be compared with European circumstances because of the political, ideological and regional differences. For this study we went to Venezuela where we conducted several interviews with representatives of the Venezuelan educational system who are involved in Computer Science Education (CSE). The interview data was analyzed using the Darmstadt Model, a model created especially for analyzing the situation of CSE in different countries. It was the first time this model was used to examine the situation of a Latin American country. Our results provide further information to people who have to develop, organize and evaluate CSE and ICT lessons.

Keywords: educational system, sociocultural-related factors, curriculum issues, policies, ideology, computer science education (CSE), Darmstadt Model (DM), Berlin Model (BM).

1 Introduction

To establish a new educational field is something other countries have done before, but it has possibly not been attempted before with the amount of effort and consistency that Venezuela put into establishing CSE and ICT. This process can therefore be called unique. It was not an organic process but a tremendous effort undertaken by the state of Venezuela and especially by its president Hugo Chavez. Chavez' personal decision to present millions of Venezuelan students with laptops is one example of his substantial role in this process. But whereas great importance was attached to providing the technical equipment, training the teachers and developing didactic materials for computer science were unfortunately neglected. The most fundamental result of this development is the predominance of the subject ICT in comparison with the subject CSE in Venezuela. This has especially one serious unintended consequence: Despite a considerable improvement in computer related education in Venezuela, most of the students remain mere users and consumers of technology. Not enough of them learn to use the computer creatively or even understand how it works.

To analyze the situation of CSE scientifically we set up a qualitative study based on the Darmstadt Model, which we briefly describe in section 2. We also analyzed accessible papers on CSE in Venezuela and conducted and evaluated interviews which we describe in section 3. We interviewed difference teachers who work for public and private schools from different parts of the country as well as representatives with different degrees of responsibility from the ministry of education. The results of our analysis are presented in sections 4 and 5, followed by the conclusion.

2 Theoretical Framework: The Darmstadt Model

The Darmstadt Model was created in 2011 to compare the situation of CSE in different countries and to provide a basis for research and a transfer of research results from one school or country to another. It is derived from the Berlin Model and is the result of a qualitative analysis of reports from different countries. It consists of three dimensions [7], see figure 1.

The first dimension is the "Berlin Model Top Dimension". This level contains all parts of the Berlin Model. The second dimension is the "Level of Responsibility". On this level all significant decision makers are listed who, are responsible for education in any way, whether at different levels of education authorities or in schools. And the third dimension is called "Educationally Relevant Areas". This level contains 13 categories, each with sub-categories.

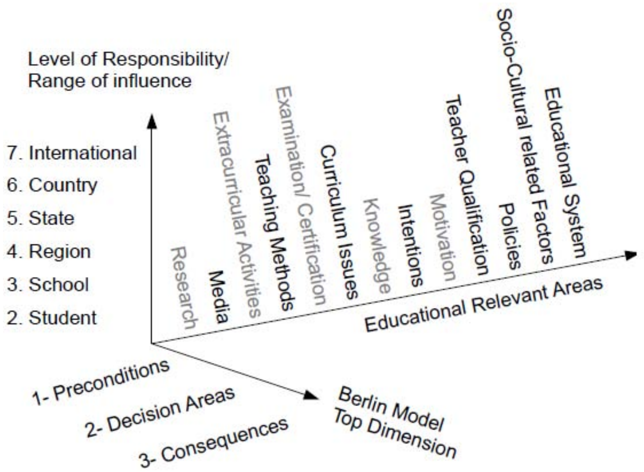


Fig. 1. The Darmstadt Model, from [7]

If one wants to use the Darmstadt Model to portray the status of CSE as an international, constantly evolving science, it is necessary to define all terms exactly and capture everything concerning CSE that is thought, written and done in different countries, [7, p. 5]. When it comes to collecting information and arranging it temporarily it is probably better to use only a few broad categories rather than many well defined areas and concepts. This applies especially when a relatively large group of people from different countries is involved.

The strength of the Darmstadt Model lies in its versatility. It can be used for both collecting information and arranging it. The remaining weaknesses of the Model will decrease the more it is used and it will be adapted to the reality of life.

3 Qualitative Methodology

According to Landsheere [9] and others and due to the general lack of information a qualitative study appeared to be the best way to meet the requirements of the three dimensions of the Darmstadt Model. But it was never the aim to map the situation of CSE in total but rather to produce a first picture by scientific methods. We describe the planning and the theoretical basis of the interviews and give some information about the interview partners and their regional positions.

Planning the Interviews: In preparation for our journey to Venezuela we did a survey of the literature about CSE in Venezuela to gain a first insight into the official point of view on CSE, [12,11,4,1,10,8]. The information from these documents and the items of the Darmstadt Model served as a basis for our questions. Because of the economic and political importance of the north we expected to produce a meaningful and representative cross section of Venezuela's efforts and results in CSE by concentrating on the capital of Venezuela, Caracas, and several states in the north including Tachira, Merida, Barinas and Trujillo, see figure 2. We also got into contact with the Venezuelan Ministry of Education, which helped us to arrange appointments, [12]. Based on the dimension "Level of Responsibility/Range of Influence" of the Darmstadt Model we decided to talk to people of different levels of administration and in different functions regarding CSE.

Interview Partners: We interviewed teachers working in the public system as well as teachers working in the private system. At the Ministry of Education we interviewed several people from the Canaima Project including teachers, sub coordinators, developers and the main coordinator of that project. We also conducted interviews with the education coordinator in the Anden region: (Merida, Tachira, Trujillo and Barinas), the general director of curriculum in Merida. (Area of Education Merida, Venezuela), and also with private primary-school teachers, a coordinator at a private school and a secondary-school teacher of computer science to 7th and 10th grades at a public high school. We also interviewed an eighth grade student and a tenth grade student from a private secondary school and the coordinator of the master of education program, a

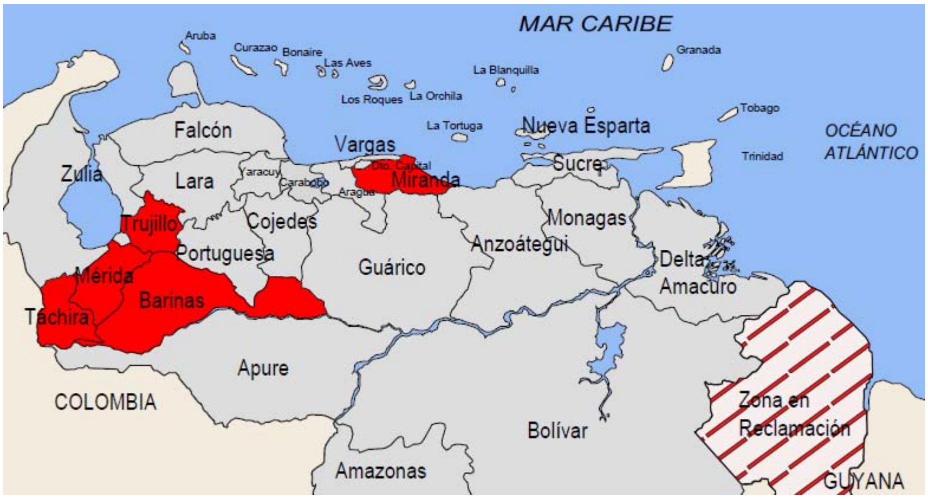


Fig. 2. The Venezuelan states visited (colored red), from [14]

professor lecturing on algorithms, programming and statistics at a university. So we had interview partners from almost all levels of responsibility and ranges of influence.

Number of Questions and Length of the Interviews: We first prepared a short list of open questions and formulated the following acceptable amount of questions (see appendix table 1) that would not consume too much of the interviewees' time but would give the chance to go into detail. The list was sent to each interviewee beforehand to give an overview and time for preparation. We tried to find questions that would produce important and applicable answers regarding the educational areas, but would also not restrict the interviewees in their answers. We hoped that the questions could trigger a discussion in which even more valuable information could be obtained. Each of the questions matched an important aspect of the "Educational Areas" of the Darmstadt Model. Six similar interview forms, differing slightly according to the interviewee's level of responsibility were used for this study. We planned to interview each person at his or her place of work. Being in their own environment would allow them to walk around, and get further information (papers, handouts, etc.) if necessary. We planned to audio tape the interviews and asked for permission after a few sentences when the atmosphere got relaxed, but more than half of them did not agree. As a consequence we only have handwritten notes on 12 of the 20 interviews and could only transcribe 8 of them in detail. Only these 8 interviews are the basis for our evaluation. The handwritten notes were only used to re-examine our findings.

Preparing the Analysis and Coping with the Data: After the interviews the audio tapes were first transcribed in Spanish and then translated into English1. While reading and transcribing and translating the interviews, we already started taking notes and decided to connect two axes of the Darmstadt Model. With the axis "Educational Areas" and the axis "Level of Responsibility/Range of Influence" we formed a grid that helped to take notes while reading the interviews and to put information in places where it could easily be found again later. Having a grid with the most important interview data (statements, information, immediate conclusions, ideas, links, etc.) also helped to get an overview of the first findings. This enabled us to decide for which part of the analysis we already had enough data.

All the interviews were conducted in a nice and friendly atmosphere. Some people were very open and keen on fulfilling our need for information. Most of our interview partners wanted to show us the best possible way of implementing Venezuelan CSE. They were motivated by a sense of pride of their country. This, of course, needs to be analyzed because we have to separate the information that is colored by national pride from the real facts.

4 Analysis of Computer Science Education in Venezuela

This chapter is a summary of the results which were achieved in the bachelor thesis [3]. We sorted these resulting information into the following eight categories referring to the second and third dimension of the Darmstadt Model, printed in black in Fig. 1:

4.1 Policies

In order to investigate the education policy in Venezuela, the following questions were not planned for the interview, but emerged spontaneously, during the interview: 1. What were the main developments of the education policy in Venezuela in the last decades? 2. How was CSE influenced by these developments?

This section gives a summary of the general policy of education in Venezuela. It includes some information about reform programs and plans and their political, legal and technical background. The focus lies on the second and third axes of the Darmstadt Model.

CSE and ICT Before the Socialists: Between 1990 and 1998 Venezuela started two programs in the field of CSE and ICT: A Computer for Every School. The objective of the first project was to create a computer lab with 20 computers for every school. The second project was planned for improving the training of teachers [10]. Due to the lack of computers and instructors both projects had very little impact.

Education Policy of the Socialist Party: When the socialist party came to power education policy changed a lot.

1999: The New Government Introduces ICT in Venezuela: When the new socialist government came to power in 1999, they were dissatisfied with the effectiveness of computer science in secondary schools at the time. Therefore they began working with the idea that the computer should be a resource for all subjects and this was when the idea of ICT in schools gained momentum. In addition, teachers were needed to prepare students for the use of the new resources. The teachers had to impart the educational software to the students, but first they had to learn how to use the technology appropriately themselves and not be afraid to use it [3](I2, line 38).

The Legal Background 2000: Decree 825: Decree 825 was written and put into effect in 2000 by the Venezuelan government. It regulated internet usage and internet access for primary and secondary schools as well as for medical institutions [6]. It supported the development of information technology and communication in schools and in the communities [3](I2, line 142).

Plans for the Education Policy in Practice: Primary and secondary education were the levels of the education system selected by the government in 2000 to incorporate ICT. A schedule for this process was created comprising three stages [12]:

- The development of educational content for primary and secondary schools.
- The training of teachers in the use of ICT.
- The development of ICT, telecommunications, infrastructure and internet connectivity.

2004: Bolivarian Missions: In addition to these laws and initiatives other programs play a role in Venezuela, the so-called Bolivarian Missions.. These social welfare programs [13] helped, among other things, in alphabetizing adults and therefore had a significant impact on education. As a result the UNESCO officially declared Venezuela an illiteracy-free territory in 2006.

2010: The Canaima Project: Since 2010, the Canaima Project has been active. It began in 2009 in cooperation with Portugal. It is an educational program to improve technological innovation amongst school children. This project especially provides teachers and all students in primary education with laptops [2]. National public schools, municipal and private schools subsidized by the state participate in the Canaima Project. In 2013 the Canaima Project was extended to include secondary schools. The intention was to promote the development of the students by incorporating the use of technology in in all disciplines and in all grades. Millions of the Canaima laptops have already been given to students of different ages. Venezuela has developed a special operating system for this laptop which is also called Canaima [2]. In the technical centers numerous teachers and scientists work on additional educational software. Section 4.2 offers a detailed description of this project.

The Technical Background: Venezuela has built various technical centers. These centers can be used by students and teachers as well as by academics and the local population. In 2008 the satellite "Simon Bolivar" [5] was put into orbit. This satellite was built with the help of China. This action had been planned and prepared since 2005 because of the government's policy to become more independent of foreign countries, and because of the rapid growth of network usage. This satellite transmits radio and TV and provides access to communication and the Internet for all primary and secondary schools in Venezuela, including the most remote parts of the country.

According to the report "Freedom on the Net 2011", which was made by Freedom House and financed by the United Nations Democracy Fund, 46 per- cent of the Venezuelan population have internet access [8]. This number will continue to rise as it is planned to provide all primary and secondary school students with a laptop. This project has helped to increase the number of internet connections in homes. And it is also instrumental in covering all regions of Venezuela with electrical power supplies because they were now needed for the internet and the laptops in the schools.

The Political and Ideological Background: In 1999 Hugo Chavez, the leader of the socialist party, came to power. He based his new policies on the heritage of Simon Bolivar, a leader of the 19th century. In his tradition Chavez wanted to deal with issues such as economic growth and alleviation of poverty. As one important step in this direction, educational improvements played an important role for the Venezuelan government.

In the area of CS and ICT the Venezuelan socialist party thought that training new generations in ICT would raise awareness of the issues concerning the community and help the people to solve problems [3](I2, line 4).

4.2 Media

This section is oriented towards the second and third dimension of the Darmstadt Model. No questions were planned to address this issue on its own. This category was found in several answers to other questions or came up as a single topic during the interviews.

From 2001 until now Venezuela created a huge countrywide infrastructure of ICT. This infrastructure consists of three different pillars. The first pillar is a dense net of numerous technical centers well equipped with ICT. A lot of these centers are free for everybody to use. The second pillar is a free laptop for every pupil attending a public school, and the third pillar is educational software for every topic. Through this infrastructure every Venezuelan citizen has access to ICT. Moreover, it is guaranteed that ICT is a part of the education for almost every student. It is remarkable that the Venezuelan government does not consider it necessary to integrate private schools, which amount to 30 per cent of all schools, into its technical infrastructure[3](I1, line 77).

4.3 Sociocultural Related Factors in Venezuela

In order to investigate the sociocultural-related factors in Venezuela, questions on the following topics were prepared: gender aspects concerning the students, the influence of the overall economic situation in Venezuela and the influence of public opinion and family socialization on CSE.

Gender Aspects: The Venezuelan government very much wants women to equally participate in technical jobs. They successfully encourage them to take up professions in this field of work.

Additional DM Related Information: The Venezuelan people are generally open-minded towards technical progress, information technologies and CSE. There are some problems with parents, who do not care how their children treat the laptops which they were given by the government[3](I4, line 8). The reasons for this neglect can be indifference or a rejection of the Venezuelan Canaima Project for political reasons. The economic situation of Venezuela has generally improved over the last decade, but in recent months the situation deteriorated a lot. If this economic downward trend should continue, the progress in CSE would be in great danger. Without the economic power of a wealthy Venezuelan middle class, the enforcement of CSE would be a demanding task left to the government[1].

4.4 Educational System

The Venezuelan educational system has many facets. With the public schools, the semi-private schools, the Venezuelan private schools and the foreign private schools it has principally four types of schools. In addition public secondary education is split into two very different branches: the technical schools and the Bolivarian schools.

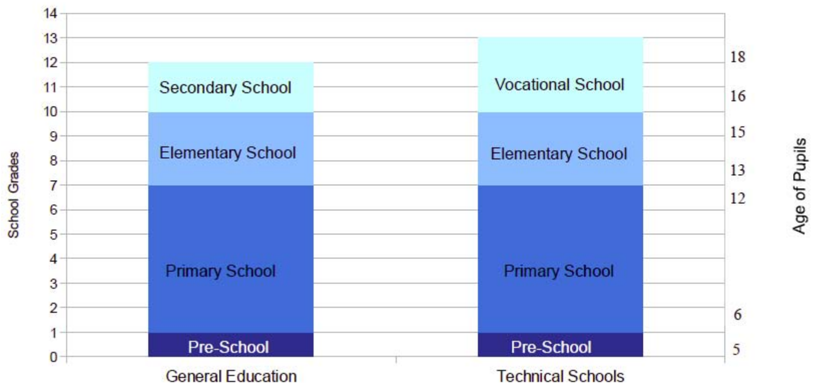


Fig. 3. Educational System in Venezuela

One can look at these schools as school types five and six. The schools are controlled by three authorities: by the Ministry of Education in Caracas, by the respective regional authority and finally by the school itself. Especially in CSE one can see the consequences of this mixed structure. Although CSE is taught

[1] A government, that is weakened by the latest developments since Chavez died just a few days before we conducted the interviews.

in a remarkably high percentage of schools, CSE is not a subject everywhere. Sometimes CSE is compulsory and sometimes it is not, sometimes two lessons are given per week and sometimes three[3](I6, line 12). The schools have different curricula. Moreover the Venezuelan state does not treat the schools equally. For example the Canaima laptop is not given to private schools. Maybe there are good reasons for all these differences. But this kind of split structure makes it impossible to have a homogeneous education on a consistent high level.

4.5 Curriculum Issues

The Ministry of Education does not provide a definitive curriculum for CSE. Neither public nor private Venezuelan schools have a definitive curriculum for CSE. So every school develops its own curriculum. For this reason you can probably find differences concerning the curricula and the content of CSE not only between public and private schools, but also between any two schools in Venezuela. The Venezuelan sate has made only one clear decision: It is not allowed to teach CSE as a subject in primary schools.

4.6 Teacher Qualification

For the analysis of this category, the following axes of the Darmstadt Model are used in this section:

– DM: 2 - Level of Responsibility/Range of Influence: Country, State, Region School and Teacher
– DM: 3 - Educationally Relevant Areas: Teacher Education, Computer Science Education, Certification and Training and Professional Experience

Like other countries, Venezuela needs to have enough well-trained teachers to fulfill the educational mandate of CSE. Therefore, public school teachers have to upgrade their teaching skills in the Canaima Centers. This can cause problems, however, if teachers are forced to do so, especially in their spare time, or if they belong to a low-wage group of professionals. If teachers are persuaded to teach ICT or to integrate computers into their lessons, this can easily lead to poor results. One interviewee told us that some teachers let the students play games instead of teaching the subject properly.

Teachers who are opponents of the leading political party may undermine the political efforts of the ministry of education, whether consciously or subconsciously. Teachers state that they know far less about ICT than their students and that they are expected to learn from their students as well as learn by doing. A teacher who is not confident about a subject is not an ideal partner in qualified education.

The exchange between the student and the teacher about ICT topics may have a certain value, but is problematic in the long run. Students want a teacher who is competent, whom they can rely on and who can give them further inspiration. It is well-known that it is necessary to motivate teachers and students to deal with the subject of ICT and CSE. Venezuela tries to achieve this by combining ICT with regional topics and regional challenges. It remains to be seen if this is going to work and achieve the results politicians would like to see for the educational process itself and for the region.

That you can study CSE only at two universities is also a problem mentioned by one of the interviewees. These two universities alone cannot cover the national demand of all the students who want to attend this university course[3](I3, line 58). Additionally, the number of graduates does not satisfy the national demand for teachers in schools. CSE is a young science in Venezuela's university education, which is still in an early stage of its development. The small amount of universities offering the chance to study CSE reflects this.

4.7 Teaching Methods

Generally, the teaching methods in Venezuela do not differ much from the methods in Europe. The so-called "humanistic approach" just gives teachers and students a different role [3](I1, line 44). The student is supposed to find the knowledge mainly by himself while his personal needs and his emotions are taken into consideration by the school.

The teacher's role is in a way diminished because he is only a consultant and a helping hand. On the other hand he has the responsibility of helping the students to develop and form their personalities. In addition this teaching style requires a great amount of preparation because it is supposed to provide the students with a challenging learning environment. The teacher has to design this learning environment and give the students useful material to work with. The students should then be motivated to use the space to learn in a reasonable way. There is one specific feature that is mentioned in several interviews, the so-called "learning areas". These comprise different subjects, that have been taught separately before. Planning and conducting an interdisciplinary lesson increases the demands on both the students and the teacher, especially when working with the computer is added as an integrated topic. Working with a computer has to be learned beforehand.

The question therefore is if the high level of conducting the lessons can be achieved in everyday school work. A relevant consideration must be the fact that neither a curriculum nor special didactic materials for CSE are available so far. This means that teachers for CSE are more or less left alone concerning their specific teaching methods.

4.8 Intentions

For this category we asked the question what Venezuelan students should know and be able to perform after they have finished CSE classes. Lessons in CSE mostly deal with product knowledge. Students only learn to use the usual programs. Conceptual knowledge and creative thinking are neglected. The popular learning tools that are a big part of Canaimas' support for school lessons for all subjects do not offer children the chance to be creative. All these learning tools were made with the help of teachers and passed four different departments to complete their validation.

5 General Results of the Categories Investigated

Venezuela has carried out 12 educational reforms in 30 years. This high amount of educational reforms asked too much of the Venezuelan educational system, causing it to become overloaded. The projects in CSE/ICT set the priority on building up the technical infrastructure, but the teachers' training, the development of didactics and the construction of a nationwide curriculum were neglected. Almost everything that was done served the field of ICT, whereas in contrast CSE lacked attention and resources. However, the massive building-up of ICT allowed citizens to get access to ICT. The Canaima Project guarantees every student a computer to work with, even if their families are underprivileged. The state of Venezuela attempts to give both genders equal opportunities and support. Venezuelan citizens are open-minded towards technical matters.

The Canaima Project is afflicted with the neglect of some children who do not appreciate a free laptop and do not take care of it. Some families also reject the Canaima Project for political reasons[3](I4, line 8). The project probably will not survive a major political change or a serious economic crisis. The school system comprises many different areas and Venezuelan schools get influenced by different sources, which makes it difficult to set and achieve consistent standards in CSE. There is still no nationwide curriculum for CSE, therefore every school has to design its own school plan.

Teachers try to change their teaching methods towards methods that are action-orientated and focusing on discovery and experimental learning. Lessons should be student-orientated rather than theoretical, and the students' previous knowledge has to be taken into account.

There are three ways in which to become a teacher for CSE/ICT, but all in all they do not train enough teachers to fulfill the nation's requirements. And it is important to understand that, as a result of implementing the Canaima Project, Venezuela needs more than the normal number of CSE teachers. The most demanding way to become a teacher of CSE is to study at university, but there are not enough places available. Therefore, Venezuela does not have enough CSE teachers and therefore the quality of CSE often does not meet international standards.

6 Conclusion

Venezuela's government helped a wide range of citizens to get access to computer technologies. This definitely is a great success. CSE could be improved greatly, however, if conceptual knowledge was much more in the center of attention. To strengthen conceptual knowledge in contrast to product knowledge would also help to get more people to choose the subject CSE.

Venezuela's situation underlines that setting up a technical infrastructure does not automatically lead to more and better-qualified personnel. It is a particular disadvantage if ICT instead of CSE is the main subject. Teacher qualification is essential and should be promoted vigorously.

The results of this study are relevant to politicians, universities and schools. Politicians who want to improve ICT and CSE can consider Venezuela's situation and proceedings. Teachers and students are generally interested in educational approaches of other countries.

Other researchers planning to conduct similar studies will hopefully gain useful information by this study. Our research in Venezuela can only be a beginning. Every aspect of the Darmstadt Model needs to be analyzed further. Traveling through a big country like Venezuela is a great challenge and it would be nearly impossible for one person alone to collect the data needed to answer every aspect of the model satisfactorily.

The method of a qualitative study based on interviews served us well, but it is obvious that it necessarily has its limitations. First and foremost, not everybody wants to give an interview. Secondly it would be necessary for a detailed picture to conduct a great number of interviews, which is not always possible. For these reasons a combination of different methods is probably the best way to conduct a study like this. However, we got structured and hopefully valuable insight into CSE in Venezuela.

References

1. Dirección general de curriculo: Líneas estratégicas en el marco del proceso curricular venezolano (2011), `http://www.me.gob.ve/`
2. Educativo, C.: Proyecto canaima (April 2014), `http://canaimaeducativo.gob.ve`
3. Fecht, N.: Analysis of the situation of computer science education in Venezuela using the Darmstadt Model. University Oldenburg (2013)
4. Fundabit: Fundación Bolivariana de Informática y Telemática (2001)
5. Fundabit: Satélite Simón Bolivar (2007), `http://fundabit.me.gob.ve/descargas/revistas/Edicion-21.pdf`
6. Gobierno de Venezuela: Decreto 825 (2000), `http://www.funtha.gov.ve/doc_pub/doc_194.pdf`
7. Hubwieser, P., Armoni, M., Brinda, T., Dagiene, V., Diethelm, I., Giannakos, M.N., Knobelsdorf, M., Magenheim, J., Mittermeir, R., Schubert, S.: Computer science/informatics in secondary education. In: Working Group Report, ITiCSE, pp. 19–38. ACM, New York (2011)
8. Kelly, S., Cook, S.: Venezuela. In: Freedom on the Net 2011 - A Global Assessment of Internet and Digital Media, pp. 355–367. Freedom House (September 2011)
9. Landsheere, G.D.: Empirical research in education. United Nations (1982)
10. Adrián, M., De Llano, J.: La informática educativa en la escuela. Colección Programa Internacional de Educadores Populares. Caracas: Federación Internacional de Fe y Alegría (2004)
11. Ministerio de Educación: Educación Inicial. Planificación y evaluación. República Bolivariana de Venezuela (2005)
12. Ministerio de Educación: Reformas políticas en la Educación venezolana (2013), `http://www.me.gob.ve`

13. Wiki: Bolivarian mission (2013), https://en.wikipedia.org/wiki/Bolivarian_mission
14. Wikimedia: Mapa de venezuela (2013), http://upload.wikimedia.org/wikipedia/commons/6/6b/Venezuela_Division_Politica_Territorial.svg

A Appendix

Table 1. Interview 1: Form to the Ministry of Education

Nr	**Content**
1	What is the definition which is used in Venezuela to define Computing?
2	Educational System Is computer science education obligatory in school?
3	Educational System In what year do the children start with computer science education?
4	Socio-Cultural related Factors Is there gender equality in computer science education?
5	Gender Aspects What percentage of boys and girls take part in computer science education?
6	Curriculum Issues How is the curriculum for computer science determined? Does it apply to all 23 states or are there differences?
7	Would you be so kind as to give me the curriculum standards?
8	Teacher Qualification Which qualification do teachers need?
9	Teaching Methods Which approaches were used in the past and which approaches are used now to structure the lessons in the subject computer science?
10	Outcomes/Effects What results are obtained by teaching computer science at schools? Do you have any anecdote?
11	Extracurricular Activities Is there a contest planned where pupils can test their computing skills every year?

“Computer Science in Context” and “Learning Fields” in Vocational Computer Science Education – Two Unlike Siblings?

Simone Opel and Torsten Brinda

University of Duisburg-Essen, Essen, Germany
{simone.opel,torsten.brinda}@uni-due.de
http://www.ddi.wiwi.uni-due.de

Abstract. Vocational and general computer science education in Germany use different teaching approaches in some areas to educate their students. Despite all differences of the school types and their goals of education, there are concepts and ideas which have separately developed, but cover similar pedagogical, basic concepts. The concept of “Learning Field-orientated Computer Science Education” (LFCS – “Lernfeldkonzept”) in vocational computer science education and “Computer Science in Context” (CSiC – “Informatik im Kontext”) in secondary education are such related concepts, as both follow the idea of teaching computer science in an activity-orientated and multidimensional way by developing suitable contexts and learning situations. In this paper we first explore the different aspects of both concepts. Afterwards we compare the similarities and differences of the concepts of “Learning Field-orientated Computer Science Education” and “Computer Science in Context”. As result of this comparison, we derive requirements for a general model of these situated and activity-orientated teaching concepts in computer science education.

Keywords: Vocational Education, Secondary Education, Computer Science Education, Learning Fields, Contextualisation, Situated Learning, Computer Science in Context, Activity-orientated Learning.

1 Introduction

Contextualisation plays a major role in modern didactics as well as in computer science education. For this reason, several contextualised approaches have also been developed and implemented in computer science education, such as projects based on anchored instruction [2] or cognitive apprenticeship [15] [31]. So far, there seems to be a lack of comprehensive teaching concepts in which contextualisation is anchored systematically. Existing approaches are the two German developments “Computer Science in Context” (CSiC) for general education [13] and “Learning Field-orientated Computer Science Education” (LFCS) at vocational schools [17]. Whereas CSiC uses contexts from everyday life (e. g. “Email for you (only)?” [11]), the LFCS has been developed to use contexts from the professional life of the students (e. g. developing a marketing game for a fictional company [20]).

These approaches originate from different types of school with miscellaneous aims in computer science education. Nevertheless, they also have some common aspects – like contextualisation as a common principle, or the current lack of high-quality teaching units as yet. These accordances and differences carry a chance to benefit from each other – to improve the respective approach, but also to develop new teaching material.

Up to now, no common approach or general concept has been developed to combine the different context-based approaches of secondary and vocational education. Therefore, our goal is to develop a general model for contextualised computer science education to allow secondary, vocational and, in prospective expansion, even higher education to benefit from each other.

To pursue this objective, we present relevant basic information about the general and vocational education system in Germany in section two. In section three we analyse and compare the approaches of CSiC and LFCS in detail. In the fourth section we describe our resulting requirements for a general model of contextualised computer science education and in the last section we draw our conclusion.

2 General and Vocational Computer Science Education

In contrast to many other countries, Germany offers a variety of secondary schools which students can attend on their way to professional life, such as intermediate secondary ("Realschule") or grammar schools ("Gymnasium"). After attaining school-leaving qualification, students can take up training and education at companies and part-time vocational schools, a concept called "dual vocational education and training"[1]. Based on individual interests and capabilities, the student can decide himself or herself which type of school to choose, but a common aim of all school types is to educate the students to master their social and professional lives.

Computer science is a non-mandatory school subject in *secondary education* in most federal states of Germany [30]. Describing the situation of German computer science education is significantly exacerbated by the existence of different school types in secondary education. Depending on the federal state and school type, computer science as school subject is offered in different grades with different learning content and a wide variety of curricular concepts. Despite the differences, all types of secondary schools in every federal state have the aim to educate the students in order to attend either university or vocational education and training. In the past years, a major conceptual change from input-orientation to output-orientation could be observed in computer science education, which

[1] During this time, students attend part-time schools for two up to three years, mostly for one or two weeks, followed by two up or to four weeks of training at the company. At the end of the training, students attend final examinations by the local chamber of commerce and industry [19].

is seen as a consequence of the poor performance of German secondary school students in PISA 2001 [24]. Resulting advancements have been the development of competence-based and output-orientated curricula (e. g. [18]) and of educational standards in secondary computer science education [10], which are seen as a promising development [4].

Vocational schools are part of the mandatory secondary school system and also include education and training for several professions in computer science, e. g. computer specialists ("Fachinformatiker"). Students at *dual vocational education and training* are apprentices and employees of their training companies as well as students at part-time vocational schools. They are trained by their companies for their profession and acquire a deep knowledge in all technologies used in their companies. Furthermore, they are used to solving problems with the help of self-directed working methods. This is one of the reasons activity-orientated teaching methods are demanded at school by the curriculum [29]. The main aim of vocational education and training is to prepare the students for the challenges of their profession and lifelong learning.

Although the structures and aims of general and vocational computer science education seem to be quite different, the question arises how both educational partners could profit from each other and improve computer science education in general.

3 A Criteria-Orientated Comparison of CSiC and LFCS

"Computer Science in Context" (CSiC) and the curricular concept of learning-fields (LFCS) in vocational education are two different situated and activity-orientated concepts, which seem to follow similar ideas. Although both concepts demand holistic and interdisciplinary context-based lessons, there are several differences between the concepts.

We established a set of criteria in an inductive way by analysing basic documents and descriptions of CSiC and LFCS. Afterwards, we selected which criteria would be necessary for the prospective model: To embed the model into a theoretical basis, it is important to examine the respective competency models (3) and the theoretical principles of both approaches (2). Another central aspect is the contextualisation itself (5), which means how to find contexts and transfer them into teaching units. Supporting acceptance of the model, we needed to include organisational (1, 4) and curricular (1) aspects. Finally, we compared both approaches concerning these criteria:

1. Target group, obligation and foundation in curriculum
2. Underlying theoretical principles and concepts
3. Underlying competency model
4. Orientation on standards and superordinated aims
5. Contextualisation of teaching units

3.1 Learning Fields in Vocational Computer Science Education (LFSC)

Vocational schools in dual vocational education and training are challenged by the heterogeneity of their students concerning age and previous knowledge. To master these challenges, the concept of LFCS has been developed [9] [27] [1], evaluated and improved [3] [26] in the 1990s and represents a substantial part of vocational education today.

Target Group, Obligation and Foundation in Curriculum: Schools in dual vocational education are part of the mandatory secondary school system in Germany. The curricular concept of LFCS is the obligatory basis of each curriculum in dual vocational education, including the profession of computer specialist [29]. For this reason, each curriculum consists of several learning fields (between 10 and 13, depending on the chosen occupation) instead of traditional subjects.

Underlying Theoretical Principles and Concepts: The concept of LFCS is based upon the concept of *activity-orientated lessons* [12] and uses an *activity classification system* instead of a subject classification system [22]. Learning fields (such as "networked IT systems", "application development and programming" or "business processes and operational organisation") are defined as "topical units which contain didactically reduced business and working processes. They define several competencies the students should gain" [25], but they do not include specific aims to be reached or skills to be acquired.

Learning fields describe the competencies which are necessary to master the challenges of the respective occupation, e. g. of a computer specialist. Each learning field contains skills and competencies from one domain of working processes and professional activities. This implies that a learning field can contain content from several topics, as most working processes deal with different questions – e. g. technical, economic or social issues. The aims of learning processes are described as *learning outcome.* Learning outcome is the displayed behaviour and job performance in different complex situations and is highly related to the concept of action competency. The concept of LFCS also distinguishes between learning outcome and *learning output*, which is described as the measurable result of education and learning processes as defined by traditional curricula [25].

Underlying Competency Model: The "Standing Conference of Ministers of Education and Cultural Affairs of Germany" (CMECA) [25] describes how the curricula have to be formulated and which competency dimensions have to be noted. The CMECA uses the competency description by Roth [23]. He describes a multidimensional structural model, which is not domain-specific and follows an action-theoretical approach. Therefore, vocational education is considered as a holistic educational process. According to the CMECA and Roth, the main aim of vocational education for students is to gain a comprehensive professional *action competency* ([25], p. 15). In this context, the CMECA defines action competency as "the individual's willingness and capability to behave appropriately as well as socially and privately responsible in vocational, social and private situations". Action competency consists of the three dimensions self, social and issue competency. The CMECA additionally defines methodological, communicative and the competency how to learn as intrinsic, cross-sectional parts of action competency.

Orientation on Standards and Superordinated Aims: The preamble of each curriculum in vocational education contains CMECA's demands that students should gain "the capability of lifelong learning, personal reflection of own activities, and a capability for flexibility and mobility" [28]. Another basic demand is to strive for international comparability of educational processes [8]. For this purpose, the "European Qualification Framework" (EQF) has been defined as a general description of skills and competencies to be gained in each profession. The EQF is also characterised by *learning outcome orientation* and *competence orientation.*

Contextualisation of Teaching Units: According to the CMECA, the curriculum should be implemented in so-called "learning situations" ("Lernsituationen", e. g. "development process of a game" [20]). Learning situations are curricular elements and small topical teaching units which implement the curricular competence expectations by using professional tasks and working processes ([25], p. 32). When developing learning situations, the teachers should always follow the curricular *situation principle*, which helps to reflect whether the didactic decisions about learning content and teaching methods would be appropriate [14]. These learning situations should enable the students to gain competencies from different competency dimensions as named above. The learning situations should also connect theoretical knowledge with practical lessons to improve the action competency of the students. In addition, the learning situations should be orientated towards the action model of self-contained activity, one of the basic principles of vocational education. Developing meaningful learning situations with relevant contexts can be difficult, even if teachers develop learning situations in groups as supposed by the CMECA [22]. Hence, by enhancing already existing but individually incomplete guidelines, a comprehensive, structured tutorial has been created and evaluated [20] to lead the teachers through the process of designing learning situations.

In summary, the concept of LFCS has been developed for the usage in vocational education. Its multidimensional and not domain-specific competency model as well as several supporting tutorials and guidelines on how to implement a context into learning situations could be a valuable resource, even for general computer science education.

3.2 Computer Science in Context (CSiC)

CSiC has been inspired by context-orientated approaches of other subjects such as physics, biology and especially chemistry, because for "Chemistry in Context" (ChiK) an empirically validated conceptual framework has been developed. For this reason CSiC is being based on situated and student-orientated concepts as well as on basic concepts of chemistry [21].

Target Group, Obligation and Foundation in Curriculum: The concept CSiC has been developed for general secondary education. Since computer science education at general secondary schools is a non-compulsory part of the German school system, the implementation of CSiC in class is voluntary.

Orientation on Standards and Superordinated Aims: CSiC is currently orientated towards "educational standards for computer science in lower secondary education" [10] [4]. Due to missing educational standards for upper secondary education, no teaching units for upper secondary schools have been developed as of now.

Underlying Theoretical Principles and Concepts: A lack of publications on basic concepts of computer science in everyday life of the students leads to the consequence that CSiC has not been based on a substantial theoretical basis in computer science. It is only based on the general concept of situated cognition [13].

Underlying Competency Model: The CSiC concept is not directly based on a specific competency model. Due to the orientation towards educational standards, CSiC is indirectly based on the competency definition by Weinert [32]. He defines competency – following a cognition-theoretical approach – as a domain-specific disposition of accomplishment and capability to solve different problems in a responsible way. This definition only refers to cognitive skills and capabilities, connected to a motivational, volitional and social willingness and capability ([32], p. 17–31).

Contextualisation of Teaching Units: How to find suitable and relevant contexts is still an intensely discussed question in computer science education [7] [5]. According to ChiK, Koubek proposes to analyse a context (which can also include interdisciplinary aspects) on whether it is relevant for the students' life or society. As second step, he suggests to formulate the expectation which competencies from educational standards should be gained during the teaching unit. Based on these results, the teaching units can be developed containing several teaching phases with a huge diversity of methods [13]. Up to now, several useful teaching units such as "email for you (only)?" [11], "cyber bullying" or "chat bots" have been developed and evaluated, but they only cover a minor part of the area of computer science yet.

Considering this analysis of CSiC, the concept suffers from a lack of theoretical foundation. Complemented by the missing concepts, it could be more promising to implement the concept at secondary schools. For this reason the further development of CSiC could be positively influenced by an existing general model of contextualised computer science.

3.3 Synopsis of Approaches

As described in the previous chapters, the approaches of CSiC and LFCS differ in several aspects. However, there is also accordance between the approaches:

- Both concepts are based on the ideas that *contextualisation* would lead to a higher level of motivation and interest as well as learning in contexts would help the students to connect their knowledge to its application. The used contexts should be multidimensional and interdisciplinary to promote the gaining of different competencies.
- The *development of material* is seen as a task for teams of teachers which support each other [6] [22].
- Different *guidelines and proposals* exist on how to develop and design teaching units for both concepts which could be improved and joined.

Nevertheless, there is a substantial number of differences between the two concepts:

- *Target Group:* Whereas CSiC has been developed for general secondary computer science education, LFCS is part of vocational computer science education.
- *Obligation:* In contrast to the compulsory curricular concept of LFCS, CSiC is just a non-compulsory teaching concept for the subject of computer science.
- *Basis for Contextualisation:* LFCS is part of vocational education and training. For this reason, the approach is based on the concept of activity-orientation and contexts should originate directly from the professional life of the students. Due to this professional context, vocational education could benefit from realistic working processes which present a potential resource to find suitable contexts.
  Contexts for CSiC originate from the everyday life or social environment of the students and follow the idea of situated cognition. Although several context ideas have been developed, it seems to be difficult yet to find relevant contexts.
- *Underlying Competency Model:* CSiC is indirectly based on the cognition-theoretical model by Weinert [32], which describes competency as personal disposition and knowledge base, shown as measurable performance. This domain-specific and output-orientated competency model consists of several levels which can be reached by students.
  LFCS has been defined by the CMECA based on the action-theoretical outcome-orientated competency model by Roth [23]. Competency consists of several holistic dimensions and represents a hypothetical definition to classify students' action.

Regarding these differences, it appears that LFCS has a broader theoretical basis than CSiC. For this reason, LFCS could be the main basis for the theoretical framework of a general model of contextualised computer science education, as described in the next chapter.

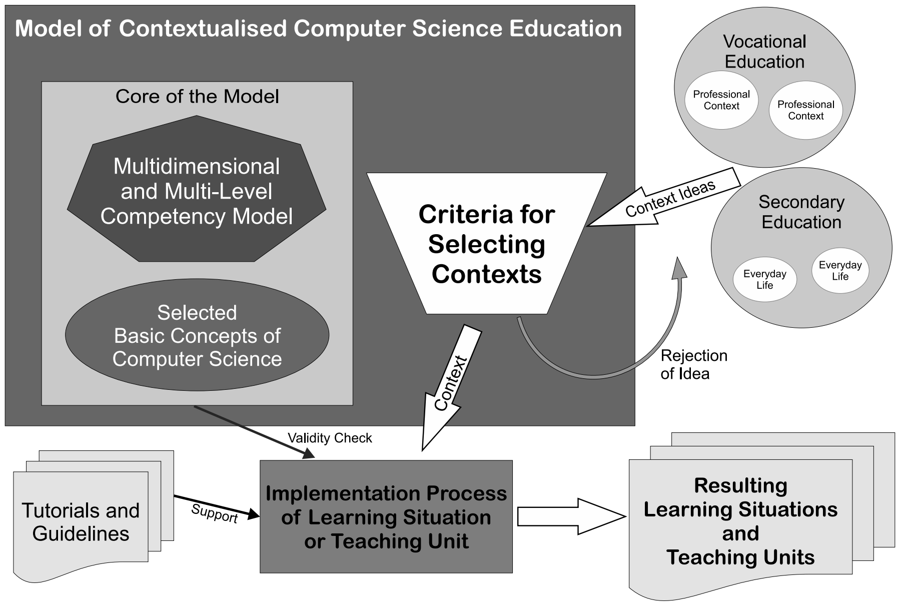

Model of Contextualised Computer Science Education
Core of the Model
Multidimensional and Multi-Level Competency Model
Selected Basic Concepts of Computer Science
Criteria for Selecting Contexts
Vocational Education
Professional Context
Professional Context
Context Ideas
Secondary Education
Everyday Life
Everyday Life
Rejection of Idea
Context
Validity Check
Tutorials and Guidelines
Support
Implementation Process of Learning Situation or Teaching Unit
Resulting Learning Situations and Teaching Units

Fig. 1. Proposed General Model of Contextualised Computer Science Education – Structure of the framework and transfer to application in practice

4 General Model for Contextualised Computer Science Education

The description of the proposed model of contextualised computer science education consists of two parts. First we describe the requirements towards a conceptual framework, in a second step we introduce the usage of the model.

4.1 Requirements

As described in section 3, there are several criteria to be included into the static framework of the model:

– For a theoretical foundation of the model, *basic concepts of computer science* have to be selected. This step is necessary to systematically select the relevant topics – relevant in everyday life – for computer science education from the wide field of computer science. A theory-driven methodology has to be developed to consider the inclusion of existing ideas, standards and conceptual frameworks of computer science education as well as the structures and basic concepts from computer science.
– Additionally, a *suitable competency model* has to be defined. We regard a meaningful competency definition and its thorough implementation as fundamental aspect of the proposed model.

The underlying competency model of LFCS describes a multidimensional structural model which implies a holistic view on competencies to be gained. It emphasises the structure of different dimensions of competence and does not describe any competency levels. As this model defines the curricular basis of LFCS, it is mandatory to include the definition for a general acceptance in vocational computer science education.

The competency definition by Weinert [32] – as an indirect basis of CSiC – focuses on domain-specific skills (which corresponds to "professional competency" in LFCS) and disregards other competency dimensions. However, this staged competency model enables us to define levels which the students should gain. For this reason, competency levels should also be part of our prospective model. Since existing competency models like the "competence model for informatics modelling and system comprehension", developed by the MoKoM project [16], focus on the subject only, we cannot use them directly, but it has to be evaluated whether it could be useful to include ideas from these models.

As shown in fig. 1, the two above-mentioned components establish the *core of the model* and have to be defined first. Based on the model core, a set of *criteria for decision-making* on whether a context idea is suitable to be implemented into a teaching unit or learning situation has to be developed. This criteria set has to be independent of vocational and general education. If required, the criteria set could be expanded by specialised criteria for the purpose of vocational or general computer science education.

The last element – but not an inherent part of the model – consists of a *collection of guidelines* on how to implement a context idea into a teaching unit or learning situation. At the time of publication, a variety of material is available to support the development of learning situations which can be adapted and expanded for the usage in general computer science education.

This description presents the static elements of the proposed model. In the next chapter we describe the resulting usage and application of the model.

4.2 Application of Proposed Model

The main purpose of the model sketched above is to enable a comprehensible and theory-based development of contextualised teaching units.

Each context idea – no matter whether from general or vocational education – has to be reviewed by using the given set of criteria. Depending on the results of this review, the idea can either be accepted and thus seen as suitable for implementation or rejected. During implementation, the guidelines support the process by defining the work flow, offering validation checks or further resources. The result of this work process should be a complete description of a contextualised teaching unit or learning situation.

Following the work flow defined by the general model of contextualised computer science education, the development of teaching material for contextualised lessons could be standardised and optimised.

After successful evaluation and improvement of the model, extensions for higher educations are conceivable.

5 Conclusion and Outlook

We started with the question whether CSiC and LFCS would be "unlike siblings" in order to define the requirements towards a general model of contextualised computer science education. As shown in section 3, both approaches are very different, particularly concerning their theoretical foundation. Nevertheless, in section 4 we could define resulting requirements to develop a theory-based model to describe these concepts. As soon as these requirements are integrated into a formal model description, it will be easier to develop teaching material or to transfer teaching units from CSiC into learning situations and vice versa. For this reason, our next steps will be to transfer these requirements into the description of a prospective "General Model of Contextualised Computer Science Education".

References

1. Bader, R.: Das Lernfeld-Konzept in den Rahmenlehrplänen (German). Die berufsbildende Schule 50(7-8), 211–213 (1998)
2. Bareiss, R., Griss, M.: A story-centered, learn-by-doing approach to software engineering education. In: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2008, pp. 221–225. ACM, New York (2008)
3. Beek, H., Binstadt, P., Hertle, E.M., Kremer, H.-H., Sloane, P.F.E., Zöller, A.: Abschlussbericht. BLK Modellversuch "Neue Unterrichtsstrukturen und Lernkonzepte durch berufliches Lernen in Lernfeldern" NELE (German) (2003)
4. Brinda, T., Puhlmann, H., Schulte, C.: Bridging ict and cs - educational standards for computer science in lower secondary education. In: Proceedings of the 2009 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2009), pp. 288–292 (2009)
5. Diethelm, I., Dörge, C.: From context to competencies. In: Reynolds, N., Turcsányi-Szabó, M. (eds.) KCKS 2010. IFIP AICT, vol. 324, pp. 67–77. Springer, Heidelberg (2010)
6. Diethelm, I., Koubek, J., Witten, H.: IniK – Informatik im Kontext (German). LOG IN (169/170), 97–105 (2011)
7. Engbring, D.: Einige Anmerkungen zum Begriff IniK (German). In: Diethelm, I., Dörge, C., Hildebrandt, C., Schulte, C. (eds.) Proceedings of 6th German Workshop in Primary and Secondary Computing Education, Bonn, Köllen, pp. 119–124 (2010)
8. European Commission. Evaluation of the European qualification framework, EQF (December 2013)
9. Fischer, M., Bauer, W.: Competing approaches towards work process orientation in German curriculum development. European Journal of Vocational Training 40(2007/1), 140–157 (2007)
10. Gesellschaft für Informatik (GI). Grundsätze und Standards für die Informatik in der Schule: Bildungsstandards Informatik für die Sekundarstufe I. Log In (150/151) (2008)

11. Gramm, A., Hornung, M., Witten, H.: Email for you (only?): Design and implementation of a context-based learning process on internetworking and cryptography. In: Proceedings of the 7th Workshop in Primary and Secondary Computing Education, WiPSCE 2012, pp. 116–124. ACM, New York (2012)
12. Gudjons, H.: Handlungsorientiert lehren und lernen. Schüleraktivierung. Selbsttätigkeit (German). Projektarbeit, 6th edn. Klinkhardt, Bad Heilbrunn (2000)
13. Koubek, J., Schulte, C., Schulze, P., Witten, H.: Informatik im Kontext. Ein integratives Unterrichtskonzept für den Informatikunterricht (German). In: Körber, B. (ed.) Proceedings of the 2009 German Conference on Informatics and Schools (INFOS 2009), pp. 268–279. Bonn, Köllen (2009)
14. Kremer, H., Sloane, P.: Lernfelder Implementieren (German). Eusl, Paderborn (2001)
15. Larkins, D.B., Moore, J.C., Rubbo, L.J., Covington, L.R.: Application of the cognitive apprenticeship framework to a middle school robotics camp. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE 2013, pp. 89–94. ACM, New York (2013)
16. Linck, B., Ohrndorf, L., Schubert, S., Stechert, P., Magenheim, J., Nelles, W., Neugebauer, J., Schaper, N.: Competence model for informatics modelling and system comprehension. In: Global Engineering Education Conference, Berlin (2013)
17. Maschmann, A.: Zur Umsetzung des Lernfeld-Konzepts im Kontext fächersystematischer Schulorganisation (German). Lernen & Lehren (2), 64–70 (2013)
18. Ministerium für Schule und Weiterbildung des Landes Nordrhein-Westfalen. Kernlehrplan für die Sekundarstufe II Gymnasium/Gesamtschule (German) (2013)
19. Opel, S., Brinda, T.: Learning Fields in Vocational IT Education – How Teachers Interpret the Concept. In: Diethelm, I., Mittermeir, R.T. (eds.) ISSEP 2013. LNCS, vol. 7780, pp. 147–158. Springer, Heidelberg (2013)
20. Opel, S., Höpfl, A., Brinda, T.: Practical implementation of learning fields in vocational it/cs education - a guideline on designing learning situations. In: Caspersen, M., Romeike, R., Knobelsdorf, M. (eds.) Proceedings of 8th Workshop in Primary and Secondary Computing Education (WiPSCE 2013). ACM, New York (2013) (in press)
21. Parchmann, I., Gräsel, C., Baer, A., Nentwig, P., Demuth, R., Ralle, B., the ChiK Projekt Group: Chemie im Kontext – a symbiotic implementation of a context-based teaching and learning approach. International Journal of Science Education 28(9) (2006)
22. Riedl, A., Schelten, A.: Handlungsorientiertes lernen in technischen lernfeldern (German). In: Bader, R., Sloane, P. (eds.) Lernen in Lernfeldern. Theoretische Analysen und Gestaltungsansätze zum Lernfeldkonzept, Markt Schwaben, Eusl, pp. 155–164 (2000)
23. Roth, H.: Pädagogische Anthropologie (German). Schroedel, Hannover (1971)
24. Schecker, H., Parchmann, I.: Standards and competence models: The German situation, ch. 8. Waxmann, Münster (2007)
25. Sekretariat der ständigen Konferenz der Länder in der Bundesrepublik Deutschland. Handreichung für die Erarbeitung von Rahmenlehrplänen der Kultusministerkonferenz für den berufsbezogenen Unterricht in der Berufsschule und ihre Abstimmung mit Ausbildungsordnungen des Bundes für anerkannte Ausbildungsberufe (German) (2011)
26. SELUBA. Modellversuch "Steigerung der Effizienz neuer Lernkonzepte und Unterrichtsmethoden in der dualen Berufsausbildung": Abschlussbericht zum Modellverscuh SELUBA (2002)

27. Sloane, P.: Lernfelder als curriculare Vorgabe (German), pp. 187–203. Schneider Verlag, Hohengehren (2001)
28. Sloane, P.F.E.: Bildungsstandards in der beruflichen Bildung – Wirkungssteuerung beruflicher Bildung (German). Eusl, Paderborn (2007)
29. Staatsinstitut für Schulqualität und Bildungsforschung. Lehrplanrichtlinie für die Berufsschule. Fachinformatiker/Fachinformatikerin, Fachklassen (German) (2007)
30. Starruß, I.: Synopse zum Informatikunterricht in Deutschland (German). Master's thesis, Technische Universität Dresden, Dresden (2010)
31. Vihavainen, A., Paksula, M., Luukkainen, M., Kurhila, J.: Extreme apprenticeship method: Key practices and upward scalability. In: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, ITiCSE 2011, pp. 273–277. ACM, New York (2011)
32. Weinert, F.: Leistungsmessung in Schulen (German). Beltz, Weinheim and Basel, Switzerland (2001)

Reasoning on Children's Cognitive Skills in an Informatics Contest: Findings and Discoveries from Finland, Lithuania, and Sweden

Valentina Dagiene[1], Linda Mannila[2], Timo Poranen[3], Lennart Rolandsson[4], and Gabriele Stupuriene[1]

[1] Vilnius University, Lithuania
[2] Åbo Akademi University, Finland, Linköping University, Sweden
[3] University of Tampere, Finland
[4] KTH Royal Institute of Technology, Sweden

Abstract. In this paper, we present the results from a multi-national study of students' results in the international IT contest "Bebras". Bebras provides motivating and game-like tasks in the format of multiple-choice questions and interactive problems to students in grades 2–12. Our study focuses on the results of nearly 8 000 students aged 10–13 in Finland (n=852), Sweden (n=201) and Lithuania (n=7 022), using gender, task and country as the underlying variables. In addition to presenting the overall results of the three student groups, we also analyse a subset of tasks in common according to Bloom's taxonomy and put forward detailed results for these tasks with regard to gender and country. The results show that there is no difference in performance between boys and girls in this age group. Our findings also indicate that there was a slight mismatch between the difficulty level of the tasks used in the contest and students' actual abilities; finding more efficient and trustworthy ways of evaluating difficulty levels upfront and choosing a suitable task set is hence important for upcoming contests.

Keywords: Informatics education, computer science education, computing education, competitions, "Bebras" contest, tasks, cognitive skills.

1 Introduction

The current status of informatics[1] education is unsatisfactory in many countries [12]. Although computers, applications and information technology (IT) in general is an increasingly natural part of the everyday work at schools, focus is mainly put on basic digital literacy skills while the underlying principles are left uncovered. This situation has been recognised as a problem in many countries [17] and recently the introduction of computing in the curriculum in e.g. the

[1] The terminology varies between countries, for instance, in the USA "Computer Science" is a widely acknowledged term, while UK started to use "Computing" a few years ago.

UK [4] and Estonia [16] has resulted in an increased debate throughout Europe. Another problematic issue concerns the view of computing as a male-dominated field, with girls losing interest in (or never even considering) computing as a career already at an early stage.

Bringing informatics to schools through curriculum changes in the form of a formal track is essential but this can also be supported in informal ways. Lately, we have witnessed an increased number of initiatives (e.g. code.org, Codeacademy, Hour of Code) aiming at making programming accessible to everyone. Similarly, the number of voluntary activities around informatics grows steadily through e.g. clubs such as CoderDojos, CodeClubs and MakerSpaces. Another similar activity is contests, for instance "Bebras", which is an international informatics contest providing motivating and game-like tasks in the format of multiple-choice questions and interactive problems to students in grades 2–12.

In this paper, we will focus on one specific age group (age 10–13) and analyse and compare the results of students in three countries: Finland, Sweden and Lithuania. Our goal is to bring light on the following questions:

– Are there notable differences in student performance among a) the three countries and b) boys and girls at this age?
– What cognitive skills are addressed in a set of Bebras tasks?

In the following we briefly discuss the status of informatics education in Finland, Sweden and Lithuania respectively. Next we describe the Bebras contest, after which we present the study setting and the results from analysing students' contest results. The paper ends with a discussion and some final words.

2 Informatics Education in Finland, Sweden and Lithuania

Students in all three countries commonly start school at the age of seven, when they enter comprehensive school. The structure and content of the education is based on national core curricula, which are renewed on a regular basis.

In **Finland**, informatics (including e.g. programming) was a compulsory subject at upper secondary level until 1994. Today informatics is not included as an independent subject in the current core curriculum for basic education (grades 1–9) [11], nor for upper secondary school level [10]. Instead IT is to be integrated in a given set of focus areas, which essentially means that students should learn to use technology in a responsible way and to use computers, software and networks for various purposes in different subjects. New core curricula will come into force in 2016 and are currently being drafted. A larger focus on both the use of informatics as well as e.g. programming, is to be expected.

In **Lithuania**, education is divided into three stages: primary (grades 1–4), lower secondary (grades 5–10) and upper secondary (grades 11–12). In 1986–2005 informatics was a mandatory subject at upper secondary level, with a strong focus on programming and algorithmic thinking (e.g. using Logo) [5]. In 2005 the subject was renamed to IT and the revisions resulted in less informatics topics being covered, focusing more on satisfying user needs and developing computer literacy. The curricula does, however, still include mandatory courses including e.g. programming. In grades 5–6 students should have approximately 15 lessons on Logo or Scratch. Similarly, in grades 9–10, there is a mandatory IT subject with several optional modules covering algorithms and programming. In grades 11–12, students can choose to learn several subjects at extended level, for instance programming modules preparing for studies at tertiary level. Students can also take an IT maturity exam, which mainly focuses on programming. New curricula are expected to be developed in 2016, and guidelines for introducing informatics and IT at all school levels are currently being drafted.

The current design of K-12 education in **Sweden** was established during the 1970s. Nine years of comprehensive school (primary and lower secondary education) is followed by three or four years of upper secondary school studies. ICT is commonly used as a tool at primary and secondary level for problem solving in other subjects and literacy. However, there is no such subject in school as computing or IT, as they are offered in Technology education in grades seven to nine. At upper secondary level, education is divided into different programs, e.g. focusing on natural sciences, technology, aesthetics or electronics. Programming courses are mainly offered at upper secondary school, in two separate courses, which are taken by students who attend one of the three programmes in technology, natural sciences or electronics. Hence, only a minority of students take programming courses.

3 Bebras – An International Informatics Contest

Different contests and olympiads [15,13] are arranged with the goal of introducing programming and other informatics domains to students. Contests make teaching of programming more attractive for students [18]. During contests students get to meet and compare their skills with peers from other schools, regions or countries [6,7].

The international Bebras [3] contest on informatics and computer fluency has been arranged since 2004 in a wide range of countries (29 countries took part in 2013). The contest was established and held for the first time in Lithuania in 2004, whereas Finland joined the network in 2010 and Sweden in turn in 2012. The main goals of the Bebras contest are to evoke interest in informatics among all students at an early stage, motivate them to learn and master technology as well as to develop their computational thinking skills [9].

The contest is organised in the second week of November in all participating countries. Contest arrangements vary slightly between countries, commonly the tasks are solved online under teacher supervision in a class room. The contest has five age groups: Little Beavers (grades 2–3 in Finland/Sweden and grades 3–4 in Lithuania), Benjamin (for grades 4–5), Cadet (grades 6–7), Junior (grades 8–9) and Senior for the oldest students. Depending on the country, the contest includes 15–21 tasks for each age group and students have 45–60 minutes time to finish the contest. Some countries use only four age groups, and there might be other small differences as well because participating countries have the freedom to adjust task sets based on their school system.

Bebras tasks are created and discussed in English during an international workshop, and each country then translates the tasks into the local language for use in the local contest.[2] Most of the problems are 4-choice questions related to information comprehension, algorithmic thinking, use of computer systems, combinatorics, discrete structures, puzzles or ICT and society [8]. In particular for the younger age groups, there are also motivating interactive tasks, where students answer by dragging and dropping objects, drawing lines, clicking on items, writing answers in text boxes, etc.

A contest with too many difficult tasks risks discouraging many of the participants, and vice versa, too many simple tasks will provide an incorrect view of informatics. Therefore tasks are categorised according to three difficulty levels - hard, medium and easy - with the intention to offer a balanced set of tasks within each age group. The scoring is in relation to these difficulty levels, as responses are mapped to 5, 4 or 3 points if correct, and -1,25, -1 and -0,75 points if incorrect. An unanswered question does not affect points at all. Initial points were given so that answering incorrectly to all questions gave 0 points. It should be noted that some countries use slightly different scoring systems.

4 Study Settings

4.1 Data Collection

This study is based on an analysis of the results from the Bebras contest held in November 2013 in Finland, Sweden and Lithuania. The total number of participants in the three countries respectively is given in Table 1, together with the corresponding distribution of boys and girls.

In order to answer our research questions (Section 1), we decided to focus our attention on one age group. We chose Benjamin (highlighted with grey in Table 1) for several reasons:

- The gender distribution is most equal for this group in all three countries (if not considering Minis).
- Students are still below the age where attitude changes towards computers and ICT commonly occur [14].
- Lithuanian students of this age should have at least 15 mandatory lessons on Scratch or Logo according to the IT curriculum.

Clearly, the number of Benjamins varied greatly between the three countries: Lithuania had over 7 000 participants, Finland roughly 850 and Sweden around

[2] In this process, it is naturally possible to arrive at somewhat different translations, which still mean the same, but that can be e.g. easier or more complicated to the students due to interpretations or misunderstandings in the translation phase. Finland and Sweden prepared tasks and translations together and consequently there should be only minimal differences between Finnish and Swedish task descriptions.

Table 1. Number of participants in Finland, Sweden and Lithuania in 2013

	Finland (X% boys, Y% girls)	Sweden (X% boys, Y% girls)	Lithuania (X% boys, Y% girls)
Mini	826 (52%, 48%)	262 (49%, 51%)	2 176 (55%, 45%)
Benjamin	852 (50%, 50%)	201 (56%, 44%)	7 022 (54%, 46%)
Cadet	1 294 (55%, 45%)	451 (55%, 45%)	6 550 (57%, 43%)
Junior	1 281 (69%, 31%)	413 (54%, 46%)	6 490 (60%, 40%)
Senior	170 (78%, 22%)	471 (91%, 9%)	3 671 (68%, 32%)
Total	4 423 (58%, 42%)	1 798 (63%, 37%)	25 909 (58%, 42%)

200. Nevertheless, when implementing the same tasks in the countries, we believe the differences in both language and school system contribute to a research setting where specific concepts can be studied.

In Finland and Sweden, Benjamins are aged 10–11 (grades 4–5), whereas Lithuanian Benjamins are a bit older (aged 11–13, grades 5–6). Students in this age group need to solve 21 tasks during 45 minutes in Lithuania and 15 tasks during the same time in Finland and Sweden. This is important to keep in mind when analyzing the data, but on the other hand Lithuanian students were somewhat older and might also have some experience with informatics concepts from their education, which could even out the situation.

4.2 Methodology

Our analysis is divided into two parts: first, we give an overview of the Benjamin results separately for each country as well as compared to each other. Second, we select 12 tasks common to all three countries for closer examination.

As mentioned in Section 3, all Bebras tasks are categorized based on the problem type (algorithm, computer use, puzzle, etc.) and difficulty level during the international workshop. In this study we also wanted to make a first attempt at introducing a common framework for categorizing tasks based on cognitive skill level. We chose Bloom's taxonomy [1], which is widely used for classifying educational objectives, and used content analysis of task descriptions for conducting the categorization. The cognitive domain in Bloom's taxonomy contains six hierarchical levels starting from simply remembering going to more complex cognitive skills:

1. **Remembering:** Recalling previously learnt information.
2. **Understanding:** Comprehending the meaning, translation, interpolation, and interpretation of instructions and problems.
3. **Applying:** Using a concept in a new situation or unprompted use of an abstraction.
4. **Analyzing:** Separating material or concepts into component parts so that its organisational structure may be understood. Distinguishing between facts and inferences.
5. **Evaluating:** Making judgments about the value of ideas or materials.

6. **Creating:** Building a structure or pattern from diverse elements. Putting parts together to form a whole, with emphasis on creating a new meaning or structure.

5 Results

5.1 Overall Performance in the Benjamin Age Group

The distribution of the total scores of Benjamins in the three countries are given in Fig. 1, Fig. 2 and Fig. 3 respectively.

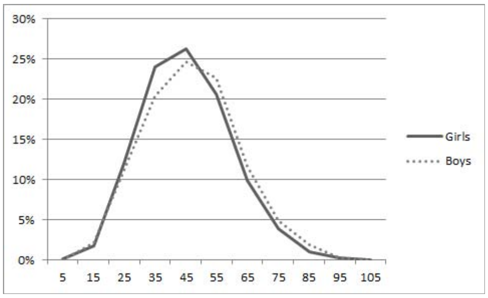


Fig. 1. Proportion of Lithuanian Benjamins achieving a given number of scores (7 022 participants, 21 tasks, maximum score 105)

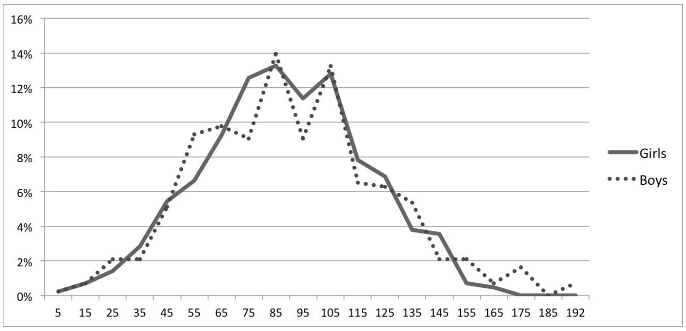


Fig. 2. Proportion of Finnish Benjamins achieving a given number of scores (852 participants, 15 tasks, maximum score 192)

As we can see from the Lithuanian and Finnish diagrams (Fig. 1 and Fig. 2), the results in these countries nearly follow the normal distribution. In Lithuania, the overall difficulty level of the tasks may have been somewhat too high for this age group as the bell curve is shifted slightly to the left. The curve does, however, show that the task set was in good balance: few students received scores around zero and few students reached the highest score. The Finnish curve is more centred around the mid score, but shows some interesting drops in the graph depicting boys' results at this point.

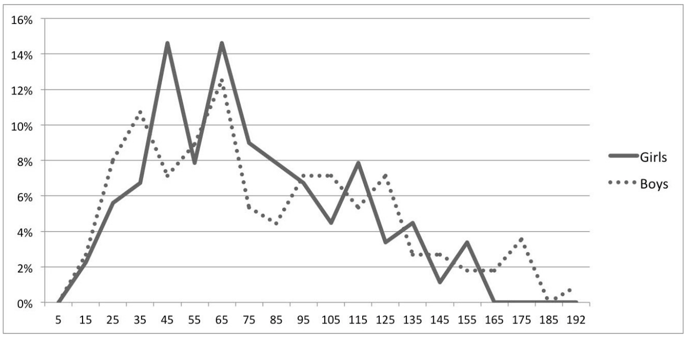


Fig. 3. Proportion of Swedish Benjamins achieving a given number of scores (201 participants, 15 tasks, maximum score 192)

As stated above, there was a large difference in the number of participants in the three countries: Lithuania's diagram is based on contest data gathered from over 7 000 participants and Finland's illustrates the results of over 850 students. In Sweden, however, the number of Benjamins was lower (201). This is quite natural, given that this was only the second time that the contest was held. Despite the low number of participants, the diagram still resembles a normally distributed curve, with some spikes and shifted to the left. In the future, when the contest will attract more participants, the results and the curve can be expected to be smoother.

5.2 Analysis of Selected Tasks Based on Cognitive Domains

Traditionally Bloom's taxonomy has been used to connect a particular teaching or training element with a certain cognitive domain level. For instance, a multiple-choice test is commonly regarded as an example of the first level - remembering or recalling information. When creating multiple-choice questions for Bebras, each task should target a given cognitive skill, focusing on student learning and understanding, not merely on recalling already known facts.

The task sets for Benjamins in Finland, Sweden and Lithuania had 12 tasks in common: 10 multiple-choice and 2 interactive ones. We analysed the descriptions for these tasks using content analysis according to their complexity from three viewpoints: a) what types of informatics concepts are hidden in the task b) how complex is the task in order of understanding, and c) do students need to follow only the given instructions or should they also apply new knowledge obtained from the task description? The results of the analysis are presented in Table 2.

5.3 Closer Analysis of 12 Tasks in Common

The proportion of correct answers for the 12 tasks in common are given in Fig. 4 and Fig. 5. The tasks are listed in order of difficulty level (assigned by the contest organisers) from easy to hard.

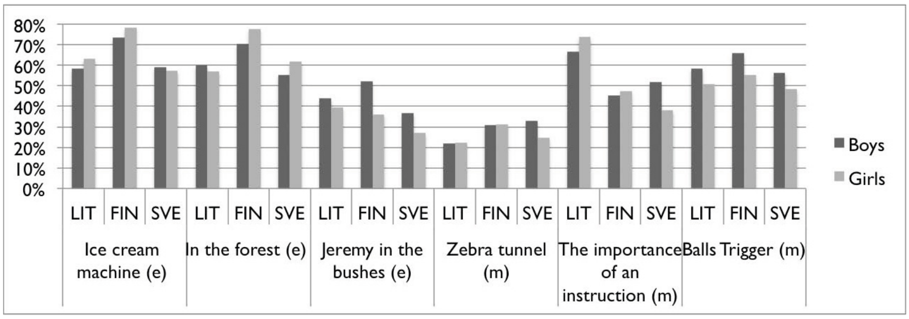


Fig. 4. The difference in relation to gender and country for six of the tasks in common. The difficulties are abbreviated as e = easy, m = medium and h = hard.

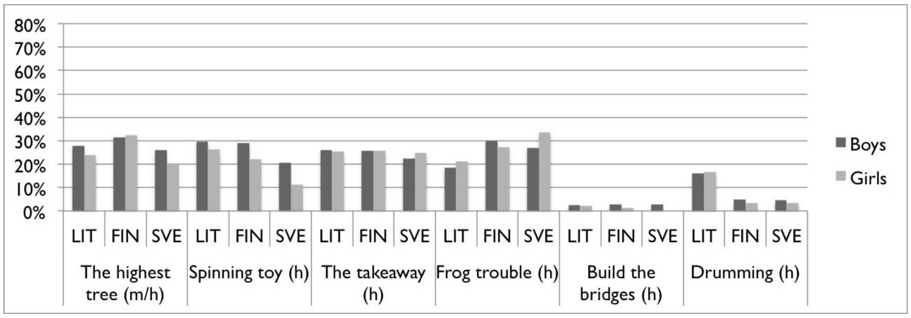


Fig. 5. The difference in relation to gender and country for the remaining six tasks. The difficulties are abbreviated as e = easy, m = medium and h = hard.

As the diagrams illustrate, the assigned difficulty level matches the actual difficulty level for many tasks (e.g. the first two easy ones and all the difficult ones), whereas some tasks seem to have been either easier (e.g. "Balls Trigger") or more difficult (e.g. "Zebra Tunnel") in practice.

The task description for "Zebra Tunnel" was slightly different in Finland and Sweden compared to in Lithuania. In Finland and Sweden the task required students to choose the correct answer from four alternatives (multiple-choice), whereas Lithuanian students were to calculate the answer and submit it in a text box (short answer). Consequently, one can assume that this particular task was more difficult for Lithuanian students as they needed to solve the task precisely without getting any help from alternative answers provided in the task.

Table 2. The 12 Bebras tasks in common, described in terms of cognitive domains (revised Blooms taxonomy)

Task name	Difficulty level	What concepts are involved in the task	Cognitive domain
Ice cream machine	Easy	detecting an algorithm; machine work; sequencing; loop	**understanding** description of non-trivial process; detection the operation of an algorithm; steps of algorithm instruction
In the forest	Easy	graph; tracing; route planning; backward strategy	**understanding** situation and planning a route from the end; separating and organising objects under given rules; distinguishing between input and result
Jeremy in the bushes	Easy	algorithm; robot navigation; tracing	**understanding** given generative rules to an input and situation; following simple; 3–5 steps algorithm instruction
Zebra tunnel	Medium	to follow instructions; algorithm analysis; data structures: FIFO (queue) and LIFO (stack)	**applying** non-trivial rules of behaviour of animals; there are representation of two ways to put data in a structure and retrieve it later; steps of algorithm instruction
The importance of an instruction	Medium	instruction; human machine instruction; unambiquous instruction	**understanding** description of processes and rules of behaviour of your partner; imagine the steps of an algorithms; interpretation of instructions
Balls trigger	Medium	instructions; logics; trigger; logical gate	**understanding** given generative rules and instructions to an initial state; following logical derivation
The highest tree	Medium (Fin/Swe), hard (Lit)	following instruction; repetition; searching algorithm; local optimisation; global optimum	**applying** given few steps non-trivial instructions with repetition; strictly following a list of prescribed instructions
Spinning toy	Hard	binary tree representation; tree traversal; operations abstraction	**applying** - identifying constituent parts and functions of an object; de-construct a process, final state or final product; applying high level abstraction
The takeaway	Hard	memory; management of data structure; stack	**applying** a given complex rule to the process; processing objects as data combinations (data structures)
Frog trouble	Hard	shortest path; breadth-first search algorithm	**applying** given instructions in the process; going from one state to another state; invention of efficient algorithm
Build the bridges (interactive task)	Hard	graph; tree; minimum spanning tree; Kruskal's algorithm; Prim's algorithm	**creating** - reviewing strategic plan in terms of efficacy; building a structure (bridges connecting islands) from diverse elements under rules; putting parts together to form a whole and to count values
Drumming (interactive task)	Hard (Fin/Swe), easy (Lit)	iteration; repetition; loops; following instructions	**analyzing** sequences (rhythms) and understanding repetition; using patterns (component parts) to organise required structure

This difference might explain the sudden drop in performance for Lithuanian students for this task (almost 10% difference in scores between Lithuanian students and Finnish/Swedish students).

The two tasks having the highest proportion of correct answers (70% or more of both boys and girls answering correctly) are "Ice cream machine" and "In the forest", in which Finnish students managed very well. The former of these tasks addresses skills from the understanding domain, whereas the latter one maps onto a higher level domain (analyzing). Interestingly, Lithuanian and Swedish students have done almost equally well on both of these tasks.

The two tasks at the end of the diagram in Fig. 5, "Build the Bridges" and "Drumming", appear to hold some difficulties. These are interactive in all three countries. "Build the bridges" expects students to create, e.g. use a cognitive skill at the highest level. The task involves building a structure (bridges connecting islands) from diverse elements under given constraints and rules, putting parts together to form a whole and to count values.

Similarly, "Drumming" addresses both creation and analysis skills as students are required to build a sequence (rhythm) based on a set of given patterns (components or parts). This task hence also assumes some understanding for concepts such as repetition (iteration) and symbolic language, which can explain the somewhat better results in Lithuania where students are expected to have some background in programming. Another task associated with algorithms and programming is "The importance of an instruction" (Fig. 4), in which students need to interpret and understand how to combine precise instructions. This task was also solved with better results by Lithuanian students.

The results show that there is no notable difference in the performance of boys and girls. Girls performed better than boys on several tasks, for example "Ice cream machine" (Lithuania and Finland), "In the forest" (Finland and Sweden) and "The importance of an instruction" (Lithuania and Finland). Boys, on the other hand, did better on for instance "Jeremy in the Bushes", "Balls Trigger" and "Spinning Toy" in all three countries.

Finally, the diagrams indicate that there is close to no spread between the countries for two tasks: "The takeaway" and "Build the bridges". These two tasks are interesting as the variables under investigation (gender, country and task) seem to have minimal influence on the contest outcome. These tasks could therefore be expected to hold non-biased qualities, in relation to the variables.

6 Discussion and Future Work

The distribution of scores from the three countries resemble the normal distribution to different extents. If the results from all three countries had been merged, we would have ended up with something very close to an exemplary bell curve. In this paper we, however, wanted to keep the student groups separate in order to make it possible to reveal any issues between countries and gender.

As stated in the introduction, informatics is still a male-dominated discipline, but our results suggest that girls aged 10–13 manage equally well (or even better) than boys in this contest. Overall, the results in relation to gender are quite balanced (see Fig. 4 and Fig. 5). Since self-confidence and perceived self-efficacy seem to play a big role for students' choice of further studies [2], the Bebras contest appears to be one way of increasing girls' belief in their own skills and knowledge in informatics. A more detailed analysis of the tasks (type, area, content) solved better by boys and girls respectively would be very interesting, as this could reveal some useful information on e.g. the characteristics of gender-neutral tasks, i.e. tasks appealing to both boys and girls.

When comparing the results from the three countries, we can see a difference in particular for tasks related to programming and algorithms (e.g. "The importance of and instruction" and "Drumming"). Lithuanian students managed notably better on these tasks, which can be explained at least partially by these students having experience in Logo and/or Scratch. Hence, these students can be expected to be familiar with instructions in the form of commands and procedures, whereas Finnish and Swedish Benjamins are most likely not exposed to programming at school.

The tasks included in the 2013 contest seem to have been somewhat too difficult for the Benjamin age group, as the bell curves are slightly shifted to the left for all three countries. The diagrams in Fig. 4 and Fig. 5 give good indications on how well the difficulty level assigned to a given task by the contest organisers corresponds with the actual difficulty level. For instance, "Zebra Tunnel" (Fig. 4) was originally labelled as a medium task, but having seen the results, we can conclude that it seems to have been more difficult than expected. Deciding on a suitable difficulty level for a given task upfront is quite difficult. Hence, coming up with an efficient and valid way of creating good tasks and evaluating them is a main priority both for us as researchers and for the Bebras community as a whole. To some extent the actual difficulty level seems to correlate with the classification in Bloom's taxonomy. It might hence be worthwhile to classify each task picked for a given age group before assigning the difficulty level.

The data presented in this paper do not provide any insight into the number of "guesses", that is, students merely choosing an answer at random instead of leaving it blank. An initial review of the detailed response data from the contests suggests that guessing is something worth further investigations, and we will study this in an upcoming paper. As a selection process is omnipresent in the school system, we will also continue our analysis by looking into the results of other age groups in order to investigate e.g. if, and in that case when, a difference in participation rates and results between boys and girls become notable.

Acknowledgements. The authors thank all members of the international Bebras contest community that took part in task development and in this way influenced the outcome of this paper. In addition, we thank all the teachers and students participating in the contest providing us with valuable data for our research.

References

1. Anderson, L.W., Krathwohl, D.R., Airasian, P.W., Cruikshank, K.A., Mayer, R.E., Pintrich, P.R., Raths, J., Wittrock, M.C.: A Taxonomy for Learning, Teaching, and Assessing: A revision of Bloom's Taxonomy of Educational Objectives. Pearson (2000)
2. Ashcraft, C., Eger, E., Friend, M.: Girls in IT - The Facts. Tech. rep., NCWIT (2012), `http://www.ncwit.org/thefactsgirls`
3. Bebras - international contest on informatics and computer fluency (2013), `http://www.bebras.org/`
4. CAS working group. A curriculum framework for computer science and information technology (2012)
5. Dagiene, V.: Curriculum for introducing information technology in Lithuanian primary education: Role of Logo. In: EuroLogo 2005: the 10th European Logo Conference. Digital Tools for Lifelong Learning, pp. 211–218 (2005)
6. Dagiene, V.: Information technology contests - introduction to computer science in an attractive way. Informatics in Education 5(1), 37–46 (2006)
7. Dagienė, V.: Sustaining informatics education by contests. In: Hromkovič, J., Královič, R., Vahrenhold, J. (eds.) ISSEP 2010. LNCS, vol. 5941, pp. 1–12. Springer, Heidelberg (2010)
8. Dagienė, V., Futschek, G.: Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: Mittermeir, R.T., Sysło, M.M. (eds.) ISSEP 2008. LNCS, vol. 5090, pp. 19–30. Springer, Heidelberg (2008)
9. Dagiene, V., Futschek, G.: Bebras, a contest to motivate students to study computer science and develop computational thinking. In: Proceedings of WCCE 2013: Learning While We are Connected, pp. 139–141 (2013)
10. The Finnish National Board of Education. National core curriculum for upper secondary schools (2003)
11. The Finnish National Board of Education. National core curriculum for basic education (2004)
12. Guerra, V., Kuhnt, B., Blöchliger, I.: Informatics at school - worldwide. Tech. rep., Universität Zürich (2012)
13. Hakulinen, L.: Survey on informatics competitions: Developing tasks. Olympiads in Informatics 5, 12–25 (2011)
14. Ilomäki, L.: The effects of ICT on school: teachers' and students' perspectives. Ph.D. thesis, University of Turku (2008)
15. Poranen, T., Dagiene, V., Eldhuset, Å., Hyyrö, H., Kubica, M., Laaksonen, A., Opmanis, M., Pohl, W., Skupiene, J., Söderhjelm, P., Truu, A.: Baltic olympiads in informatics: Challenges for training together. Olympiads in Informatics 3, 112–131 (2009)
16. Progetiger - programming at schools and hobby clubs (2012), `http://www.tiigrihype.ee/en/programming-schools-and-hobby-clubs`
17. Shut down or restart? The way forward for computing in UK schools, Royal Society report (2012)
18. Verhoeff, T.: The role of competitions in education, future world: educating for the 21st century. In: A Conference and Exhibition at IOI 1997 (1997)

A High-Availability Bebras Competition System

Nataša Kristan[1], Dean Gostiša[1], Gašper Fele-Žorž[1], and Andrej Brodnik[1,2]

[1] University of Ljubljana, Faculty of Computer and Information Science, Slovenia
{natasa.kristan,polz,andrej.brodnik}@fri.uni-lj.si,
dean.gostisa@gmail.com

[2] University of Primorska, Department of Information Science and Technology, Slovenia

Abstract. In this paper we present a new system that can be used for the Bebras and related competitions. The system supports both non-interactive tasks consisting of a question and multiple choice answers and standardized interactive tasks. It also contains a highly versatile administration interface permitting individual teachers/mentors to organize their own competitions (class-wide, school-wide, ...) that can be used in teaching of informatics. Furthermore, the system is also highly scalable and can be distributed across multiple servers.

We have successfully evaluated the system in multiple competitions, some with over 10,000 students. Because it is designed to support i18n, it was easily localized and used in Slovenia and in Serbia.

1 Motivation

1.1 The Bebras Competition

In Slovenia there are three competitions run by ACM Slovenia [17]: *Bober* (*Bebras, Beaver*), *RTK* and *UPM*. Each of them is a part of or reflects some international competition. *Bober* [3] is intended for primary (grades 3-8)[1] and secondary school (grades 9-12) children. *RTK* (grades 9-12) consists of several disciplines from web programming to video production but also includes a national version of an IOI-like (*International Olympiad in Informatics*) competition. Therefore, the winners participate in the *Central European Olympiad in Informatics (CEOI)* and are considered as participants for the *IOI*. The last competition is *UPM* (*Univerzitetni Programerski Maraton*, Slovene for *University Programming Marathon*) and is organized for university students. It is in fact a national version of the ACM ICPC (*ACM International Collegiate Programming Contest*). The best teams from the *UPM* participate in the *CERC* (*Central Europe Regional Contest*), a regional part of the *ICPC*.

Bober [4–7, 12] is a competition in algorithmic thinking, the main purpose of which is to increase the interest of students in Computer Science. The idea is to show them that a computer is not just a tool for communication, surfing the internet, editing text, listening to music or watching movies, but rather

[1] In Slovenia primary schools have 9 grades which we count as K-8 and secondary school 4 grades, 9-12.

an inexhaustible source of interesting problems (tasks). In most cases, to solve a task, algorithmic thinking, logical deduction and problem solving skills are required, but some tasks are also about the use of information technology and its influence on society.

In Slovenia the competition is run in two rounds: school-wide in November and nation-wide in January. In each round there are 15 tasks with a list of multiple choice answers. As the student enters the answer, it is saved automatically. Furthermore, students are not penalized for unanswered tasks, while an incorrect answer is penalized.

Moreover, the competition is divided into four categories: *Bobrček* (grades 3-5, international category Benjamin), *Mladi bober* (6-8, Cadette), *Bober* (9, 10, Junior) and *Stari bober* (11, 12, Senior).

The Beaver competition was first organized in Slovenia in 2010 as a pilot competition with 199 participants. The number of participants in the subsequent years has grown steadily with 3,454 taking part in the competition in 2011, almost 8,200 in 2012 and more than 11,500 in 2013. In 2013 there was also a pilot competition organized in the category *Cicibober* (grades K-2) with approximately 200 participants.

Each school has a coordinator and possibly several mentors depending on the number of participants. The coordinators and mentors are responsible for organizing and running the first round of the competition. The second round (national level) is organized by the Programme committee appointed by the ACM Slovenia. The Programme committee is also responsible for preparing the tasks for both rounds. The national competition is organized with the help of the three largest Slovenian universities. Each year, a different university hosts the competition. In 2014, a model solution booklet [1] was also prepared by the members of the Programming committee Janez Demšar and Špela Cerar. Moreover, all tasks from Slovenian competitions since 2010 are available at the website `http://tekmovanja.acm.si/bober`.

1.2 Existing Systems

For the first few years, we used the Finnish system called *Majava* [18]. Majava was built using Rails, a rapid web development framework [13] in the programming language Ruby [19]. With time, features were added to the system by different developers. Some of these features break the separation between models, views and controllers. For example, although the system was designed to be database-agnostic, its current version occasionally accesses the SQL database directly and is therefore strongly tied to MySQL.

Indeed, there exist multiple alternative Bebras competition systems. Some of them are closed source (e.g. the system used in Germany, Austria, Canada and some other countries), some are simply using Moodle LMS (Italy) and some developed their own system. For example in France, a system with an optimized front-end is used, which reduces the number of requests from clients to the server and consequently the load on a server [14]. While the web workload is distributed, all the web servers access a single relational database. The sessions are implemented using Memcached technology [10], reducing the load on a database while still maintaining a single session across all the web servers. If the web servers become unavailable, competitors are provided with a coded message at the end of their competition. They can send this coded message to the organizers over e-mail and have their results entered into the system. To minimize the communication between the web server and the clients, results are only submitted at the end of the competition with no backup in case a competitor's web browser crashes.

The architecture of the Latvian system is better separated into layers and can be used on Amazon's cloud [2]. It uses the S3 distributed data storage service to serve the images used in the competition, potentially risking that an attacker could access the images given the URL of an image, even if they are not competing. The Latvian system is highly scalable and uses MySQL to store the data regarding a competition, a caching system with periodic updates to reduce the load on the database and DynamoDB (a fast, scalable NoSQL database) to store the competitor's answers. This system should scale to more than 100,000 competitors per hour.

1.3 The Design Requirements of a New System

As mentioned previously, we initially used the Finnish system *Majava*. The prime reasons for this choice were the availability of it's source code and the support we got from our colleagues in Finland. However, we soon ran into some problems with the system. The first one was localization to Slovene, where we noticed that some text messages or words were hard-coded into the programme[2]. Also, the task images were publicly available even for non-competitors (cf. French system). Moreover, during the 2012 competition, the load on our server was so heavy that we came almost to the edge of its capacity. However, the biggest problem we encountered was the design issue that includes the administrative interface. Namely, the *Majava* system was designed as a single round, one year system, while in Slovenia we have a two round system with a heavy load on school coordinators and mentors.

Although it is probably possible to extend the *Majava* system to serve our needs, we decided to create our own system. First we put up a number of functional requirements and complemented them with the system performance requirements. The functional requirements included internationalization (i18n) and later localization (l10n) to a specific language, support for a two round system, and support for school coordinators and mentors. As a consequence, the new system provides teachers with the possibility to prepare their own competition and use it in the educational process. The performance requirements included robustness and scalability to several tens of thousands of competitors.

[2] For example, the levels of task difficulty (easy, medium and difficult) were used as strings in the calculation of a final score for a student.

More details of the system are described in the next section, which also includes an architectural overview of the system. It is followed by the evaluation section and ends with conclusions and directions for future work.

2 A Distributed Competition System

We first describe the system performance requirements in more detail and then present the system architecture that fulfills these requirements.

2.1 System Overview and Design

As the number of participants in Bebras competition steadily grows, the performance requirements also grow. Since we ran into performance problems with the *Majava* system in the 2012 competition, and since we expected even higher participation (over 10,000) in 2013, we had to redesign the architecture of the system. The new system would therefore have to be:

– High performance.
– Scalable.
– Fault-tolerant.

To achieve this, we decided to use a three-layer architecture consisting of a front-end layer, a business logic layer, and a distributed database back-end layer [15]. As shown in Fig. 1, the front-end was implemented using HTML, CSS and Javascript, the business logic layer was implemented in PHP using the Yii web development framework [21], while MySQL Cluster was used as the database back-end.

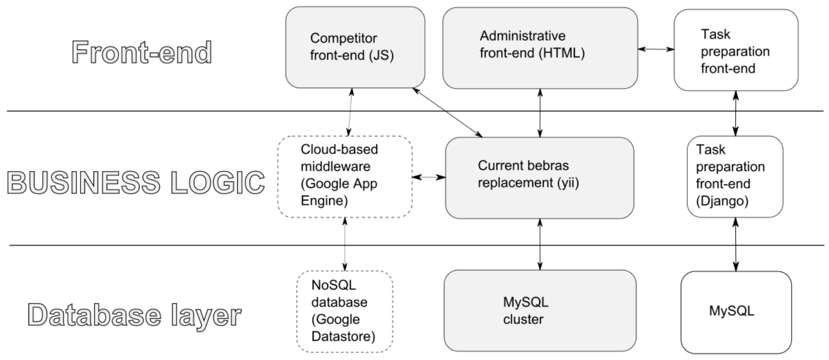

Front-end
Competitor front-end (JS)
Administrative front-end (HTML)
Task preparation front-end
BUSINESS LOGIC
Cloud-based middleware (Google App Engine)
Current bebras replacement (yii)
Task preparation front-end (Django)
Database layer
NoSQL database (Google Datastore)
MySQL cluster
MySQL

Fig. 1. The architecture of our system. The grey-filled boxes are the subject of this article and represent our new system. The empty boxes represent the task preparation system. The dotted boxes represent our future work.

2.2 Front-End

Competitor Front-End. To minimize the load on the web server, most of the processing was moved from the web server to the client. The communication between the web server and the client was reduced as much as possible while still ensuring that possible web browser client crashes would not cause data loss.

After registering for the competition, each competitor is redirected to a page which includes the CSS files and a Javascript application. The Javascript application then loads the images and texts for all the tasks, which minimizes the number of connections to the web server per client. Because the data is transfered in one go, loading times can also be decreased. The Javascript application then downloads the session state which might contain the answers by the competitor from previous attempts and a unique random number. After loading all the data, the first task is displayed. Since the duration of the competition is limited, the start of the competition for a student is counted only from the moment when all tasks have been loaded into their browser and the first task is displayed.

When the competitor switches to the next task, his/her answer is sent to the server. This ensures that no answers are lost if the competitor's computer crashes – (s)he can move to another computer and continue with the competition. Moreover, Javascript application is designed to support the Bebras Task Standard API including interactive tasks.

Administrative Front-End. Among other features, our system provides teachers with competition tokens for students and also with the possibility to check and download their results and diplomas. Moreover, it permits the organization of multiple competitions in multiple countries, regions and schools. To facilitate this, multiple administrative roles are defined:

System administrators have full access, including management of countries, languages, competition categories, task difficulty levels and users.

Country administrators can manage regions, municipalities and schools. Furthermore, they can also add mentors and assign school coordinators, set up country-wide competitions, inspect results and decide which competitors get to advance to the next round of the competition.

Mentors are provided with competition access codes for students that they distribute to their students for a competition. Mentors also have access to the results of their students.

Coordinators are mentors with special privileges with access to the results of all students at a school, even those who have a different mentor. Moreover, a coordinator validates new mentors from their school.

The system also supports guests. Competition results for registered competitors are stored and a mentor can view his or her results from competitions in previous years. Competition results for guests are displayed immediately after finishing the competition. Obviously, only training competitions made up of old, previously used tasks are accessible to guests.

Task Entry. We decided to completely separate task creation from the competition itself. Tasks are created and managed on a separate system built in the Python language using the Django web development framework. This system supports multiple task revisions and online editing. All tasks, including simple multiple choice tasks are converted into interactive tasks implementing the Interactive Task Bebras API. The interactive tasks are exported from the task preparation system as zip files and imported into the competition system.

2.3 Business Logic Layer

The business logic layer is the most complex part of our system. To increase its robustness, we used DNS parallel entries for its several running copies. To ease development and enhance maintainability of the system, we used the Yii web development framework. The framework provides an object-relational mapper. The application is therefore database-agnostic. While we used the MySQL Cluster database for the actual competition, development was done on an ordinary MySQL database using the InnoDB back-end. The back-end could be replaced by any one of the relational databases supported by PHP Data Objects. The list of supported databases includes MySQL, Firebird, Informix, Oracle, ODBC, PostgreSQL, SQLite and others. The business logic layer also restricts access to task files such as pictures to registered competitors only.

2.4 Database Back-End

We ensured the robustness and scalability of the back-end by using MySQL Cluster [20] as the database back-end. MySQL Cluster is a back-end for MySQL using a Network DataBase (*NDB*) cluster as its storage engine. It supports both sharding and failover and is therefore both reliable and scalable. Since version 7.3, MySQL Cluster supports foreign key constraints which greatly eases development. The main downside of MySQL Cluster is that by default it loads all the primary keys in memory to improve performance. This means that memory usage of the database server is greatly increased compared to the old system.

3 Evaluation

We give only a partial evaluation of the system. The reason for this is that the development started pretty late and not all functionalities were available at the time of the November round of the competition. The system is available at URL `http://bober.acm.si/`.

3.1 System Configuration

Due to relatively high memory requirements, we deployed our system across three Linux containers on two physical servers. The servers had two 16-core AMD Opteron$^{\text{TM}}$ 6272 processors (32 cores) each running at up to 2.1GHz with 128GB of RAM. Two of the containers were running on the same server and on the same filesystem. The storage on the system with two containers was 2x3TB Serial AT Attachment 2 (*SATA2*) hard drives (WDC WD30EFRX-6) and 2x256Gbyte SATA3 Solid-State Devices (*SSD*s) (M4-CT256M4SSD2). Logical Volume Manager (LVM) was used on all disks with SSDs configured as RAID1 with 190GB of the SSDs used as cache using bcache [16] for the 3TB drives, while the single-container system used a single 250GB SATA2 hard drive (ST3250310NS).

During the competition the system was continuously monitored for memory consumption, CPU load, network traffic and database parameters on all servers using the Zabbix system.

3.2 Functionalities

After the initial distribution of usernames and passwords to the teachers, all communication went through the system. This includes announcement of competition results and individual results of students to their mentors. Moreover, all diplomas and certificates for students and teachers were distributed through the system as well.

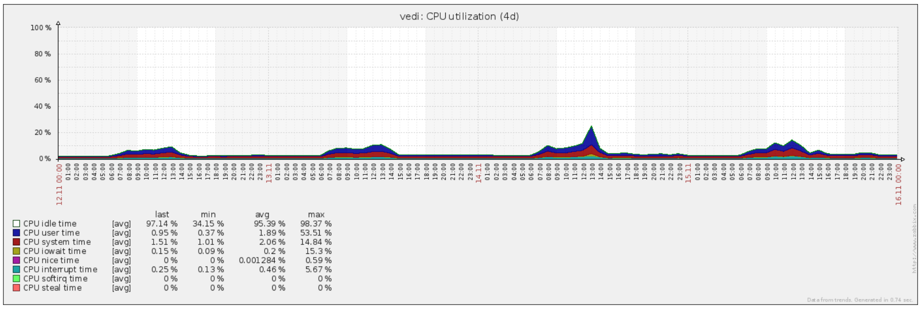
The system was localized to three languages (English, Slovene and Serbian) and used in competitions with the last two languages. Localization proved to be simple and quick and was accomplished in less than a week. The number of included schools grew up to 227 with over 130 mentors for Slovenian competitions and over 70 schools in Serbia. Note that by using a single system, the administrative overhead was greatly reduced.

3.3 System Performance

Our system was successfully used in three large competitions: school-level and finals in Slovenia, and school-level in Serbia. The school-level competition in Slovenia was performed across four days and involved 11,653 competitors. The school-level competition in Serbia was performed across five days and involved 6,280 competitors. The finals in Slovenia had almost 300 competitors competing at the same time. The maximal number of students starting the competition in the same hour was 727 with a maximum of 48 students starting the competition within the same minute and with up to 6 students starting in the same second.

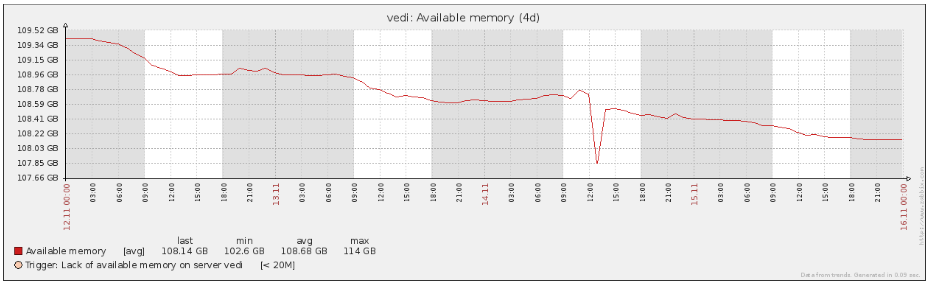
The system performance was measured only during various competitions and we were not able to fully utilize the servers available given the number of simultaneous competitors. During the largest competition (school-level in Slovenia), the load on the server with two containers never exceeded 35%, where the most of the competitors were handled by the first server in our pool. As shown in Fig. 2, the maximal CPU utilization was always under 25%.

Moreover, Fig. 3 shows the maximal memory utilization was 23.13GB of total 128GB for the whole server running two containers. The memory consumption of one distributed database instance was under 7GB while the consumption of the rest of the web server was negligible. Because most of the data is pre-cached, the competition itself had a negligible impact on the memory consumption.



		last	min	avg	max
☐ CPU idle time	[avg]	97.14 %	34.15 %	95.39 %	98.37 %
🟦 CPU user time	[avg]	0.95 %	0.37 %	1.89 %	53.51 %
🟥 CPU system time	[avg]	1.51 %	1.01 %	2.06 %	14.84 %
🟨 CPU iowait time	[avg]	0.15 %	0.09 %	0.2 %	15.3 %
🟪 CPU nice time	[avg]	0 %	0 %	0.001284 %	0.59 %
🟦 CPU interrupt time	[avg]	0.25 %	0.13 %	0.46 %	5.67 %
🟩 CPU softirq time	[avg]	0 %	0 %	0 %	0 %
🟥 CPU steal time	[avg]	0 %	0 %	0 %	0 %

Fig. 2. The CPU load of a server running two instances of our system during a competition involving over 10,000 students



		last	min	avg	max
🟥 Available memory	[avg]	108.14 GB	102.6 GB	108.68 GB	114 GB
● Trigger: Lack of available memory on server vedi	[< 20M]				

Fig. 3. The memory consumption on a server running two instances of our system during a competition involving over 10,000 students

Minimal System Requirements. The new competition system could be used on any computer with PHP and MySQL. Based on performance data acquired during competitions, we expect that a quad-core server with 8GB of RAM should be enough to handle the number of competitors we had participating in our competition. To ensure robustness and high availability two or three such systems would need to be set up.

Anecdotally, during the finals in Slovenia, the load on the servers was so low that the staff in charge of monitoring the system environment had to call the organizers to check if the competition was still going on.

Potential Problems. Due to shortage of time and late development we ran no detail load tests on our system. The only serious load test was the competition itself and we encountered virtually no problems. Indeed, we are aware of two problems only. One situation happened during the Slovenian finals where a small number of client machines had an older version of InternetExplorer without Javascript support for the front-end to function properly. The problem was resolved in less than five minutes at no cost to the competitors as the competition is timed individually per competitor. The other situation happened during Slovenian first round when a MySQL Cluster failed unexpectedly for unknown reasons. The server was restarted automatically without the operator's intervention and the competition continued with no problems about ten minutes later.

4 Future Work

We built a robust and scalable system used in Slovenian and Serbian Beaver competition. The system was built according to a three-layer architecture, each layer on its own being designed to support scalability and robustness. Moreover, the built system provides a sufficient level of administration support including the possibility for teachers to access the data of their students and even organize their own competitions. Nonetheless, the system is easy to localize to other languages, which was exhibited by the localization to Serbian.

There are still a number of future directions. Due to late development, the most obvious ones are to perform detail load and failover testing of the system, and to prepare a more thorough documentation of the system [11]. Furthermore, the user experience in an administrative interface in particular needs an improvement.

On the other hand, to bring even more scalability and robustness into the system would require an architectural redesign of the competition part along the lines of the Latvian system [2]. As already presented in Fig. 1, we could introduce a temporary non-relational data store to keep the competitor's answers during the competition and later aggregate them into the central database. The technology used would probably look like *Google App Engine* [8] for the business logic layer and *Google Datastore* [9] for the database layer.

Last but not least, we would like to encourage further use of our system either by joining to the installed version or take the code and build your own system. All the code is available under a GPL license.

Acknowledgement. We gratefully acknowledge the help of Niko Colnerič with the *Majava* system, Grega Pompe for writing the task creation system, Bojan Pikl with implementing the application for automatic generation of diplomas and certificates, and Milutin Spasić for helping to design the system, localizing it to Serbian and helping to organize the competition in Serbia.

References

1. ACM Slovenia: ACM tekmovanja – Bober (2012), `http://tekmovanja.acm.si/bober` (accessed February 03, 2014)
2. Balodis, M.: Mākonī izvietotas sistēmas slodzes test (2012)
3. Bebras: Bebras, international contest on informatics and computer fluency (2004), `http://bebras.org/` (accessed February 03, 2014)
4. Bell, T., Curzon, P., Cutts, Q., Dagiene, V., Haberman, B.: 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 (2011)

5. Dagienė, V.: Information technology contests: Introduction to computer science in an attractive way. Informatics in Education 5(1), 37–46 (2006)
6. Dagienė, V.: Teaching information technology and elements of informatics in lower secondary schools: Curricula, didactic provision and implementation. In: Proceedings of the 3rd International Conference on Informatics in Secondary Schools - Evolution and Perspectives: Informatics Education - Supporting Computational Thinking, pp. 293–304 (2008)
7. Dagienė, V., Futschek, G.: Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: Mittermeir, R.T., Sysło, M.M. (eds.) ISSEP 2008. LNCS, vol. 5090, pp. 19–30. Springer, Heidelberg (2008)
8. Google Developers : Google app engine: Platform as a service, `https://developers.google.com/appengine/` (accessed February 03, 2014)
9. Google Developers: Google cloud datastore, `https://developers.google.com/datastore/` (accessed February 03, 2014)
10. Dormando: Memcached, `http://memcached.org/` (accessed February 03, 2014)
11. Gostiša, D.: Visokorazpolojiva storitev tekmovalnega sistema v oblaku. Diploma thesis, University of Ljubljana, Faculty of Computer and Information Science, Ljubljana (2014) (Engl. translation: A high-availability cloud-based competition system service)
12. Haberman, B., Cohen, A., Dagiene, V.: The beaver contest: Attracting youngsters to study computing. In: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, pp. 378 (2011)
13. Hansson, D.H., et al.: Ruby on rails (2009), `http://www.rubyonrails.org` (accessed February 03, 2014)
14. Hiron, M.: Personal correspondence (2013)
15. Kate, M.: Scalable web architecture and distributed systems, `http://aosabook.org/en/distsys.html` (accessed February 03, 2014)
16. Kent, O.: Bcache, `http://bcache.evilpiepirate.org/` (accessed April 24, 2014)
17. Kristan, N., Brodnik, A.: Integrated ACM competitions website. In: Proceedings of the 10th IFIP World Conference on Computers in Education: Learning While We are Connected, pp. 174–175 (2013)
18. Majava: Majava kilpailu, `http://www.majava-kilpailu.fi/` (accessed February 03, 2014)
19. Matsumoto, Y., Ishituka, K.: Ruby programming language. Addison Wesley Publishing Company (2002)
20. MySQL: MySQL Cluster CGE, `http://www.mysql.com/products/cluster/` (accessed February 03, 2014)
21. yiiframework: The fast, secure and professional php framework, `http://www.yiiframework.com/` (accessed February 03, 2014)

Competence Orientation in Vocational Schools – The Case of Industrial Information Technology in Austria

Markus Brunner[1] and Monika Di Angelo[2]

[1] HTL Krems, Department for Information Technology, Krems, Austria
markus.a.brunner@gmail.com
[2] Vienna University of Technology,
Institute for Computer Aided Automation, Vienna, Austria
monika.diangelo@tuwien.ac.at

Abstract. As part of the project "Educational standards in vocational schools" the responsible Federal Ministry compiled competence models, descriptors and didactic examples for Informatics in technical high schools (HTL). The implementation of this project implies a paradigm shift for the vocational school system, which requires a new pedagogic foundation. This paper presents a didactical concept, that meets the requirements of the educational standards in general and competence orientation in particular. A proposal for the implementation of these educational standards in the competence area industrial information technology (INIT) is presented. The specification is based on selected teaching sessions for micro controller technique. Learner-oriented teaching methods are applied, along with procedures to promote the learners' intrinsic motivation and creativity in general. It became apparent that the competence area INIT conveys remarkably demanding learning outcomes and its implementation proved challenging on multiple levels.

Keywords: vocational schools, competence orientation, educational standards, technical high school, secondary education.

1 Introduction

Austria's school system is in a flux, educational standards and competence orientation are the two key topics. The declared objectives in this process are a sustainable increase of quality and a better comparability of educational attainment.

This change is already partly implemented: in 2011, the Federal Ministry of Education, Arts and Culture (bm:ukk) released new educational standards for the vocational school system [1] which are orientated towards the development of well-defined competences. To this end, the ministry compiled competence models, descriptors and didactic examples for Informatics in technical high schools (HTL) as part of the project "educational standards in vocational schools".

These new curricula induce a twofold change: new contents and an orientation towards competence attainment. While the adaptation of the contents is caused by the dynamics of IT, the change towards competences bears challenges on multiple levels. The predominant method in the class rooms is still ex-cathedra teaching where the students assume a rather passive role. This is in conflict with the characteristics of competence orientation which calls for a paradigm shift in teaching. Therefore, new didactic concepts have to be developed that cater to the new contents and the demands of competence orientated teaching.

This paper deals with the development and implementation of a didactic concept for the competence field "Industrial Information Technology" (INIT) within the vocational schools for information technology (HTL for IT) as a case study. The pursuit of this new concept entails the following central questions:

Q1: In which way should teaching be designed as to ensure the educational standards for the competence field INIT?

Q2: In which way can be ensured that individual learning goals are attained in combination with personal and social competencies?

Q3: To what extent does a specific school influence the implementation of the new curricula with respect to activity-oriented teaching methods?

Q4: In which way should teaching be designed as to foster creativity and intrinsic motivation?

Section 2 of this paper discusses Austria's approach towards competence orientation within the vocational schools for IT. In section 3 the case study is presented. Finally, an evaluation and discussion can be found in section 4.

2 Competence Orientation at Vocational Schools for IT

Educational standards can be seen as precise and binding expectations regarding competence attainments of the learners. A so-called competence model (c.f. section 2.2) serves as basis for the definition of the aspired competences, which include operational competences with respect to the learning content. These competences comprise expertise and methodological competences, as well as social and personal competences. Competence orientation in education systems means a focus on fostering the desired competences.

One of the reasons for a shift towards competence orientation lies in the comparability of education systems. To that extent, the EU developed the European Qualifications Framework (EQF) [2] in 2008.

In order to improve the quality of informatics education by means of standards and competences, there have been initiatives in Germany, like MoKoM [3] or the GI standards [4].

The current initiative in Austria is based on Weinert's definition of competence [5]. The legal basis for competence orientation in vocational schools in Austria is the enactment of the respective curricula [6]. Details for the implementation can be found in the ministry's report "educational standards for vocational schools" [7].

2.1 Teaching Competencies

Austria's new educational standards define the desired learning attainments in terms of competencies. Adequate teaching methods are required for the learners to acquire these competencies effectively [8, 9, 10]. These have to go beyond the conveyance of factual knowledge.

The ministry's report "teaching competencies" [11] still defines the main components of teaching as planning, delivering, and assessment, which accounts for traditional teaching as well. New territory has to be conquered with regard to the teaching methods: They should aim at establishing operational competence, skills and abilities, along with personal and social competence. A major difference lies in the learner-centered approach as opposed to traditional subject-centered or teacher-centered methods. More specifically, learners are no longer mostly passive consumers of conveyed information, but are put to action.

In this sense the teacher becomes a learning coach, while the learners are encouraged to actively work through the contents, thereby acquiring the desired competences. The teacher's responsibility shifts towards providing suitable conditions, materials and a productive atmosphere. In addition to factual knowledge, an interdisciplinary embedding of knowledge and skills plays a central role. For teachers this implies a coordination and alignment with other teachers with respect to content and timing.

The ministry's report [11] further identifies the following criteria as essential to successful competence orientated teaching: Structured teaching with clearly defined objectives; Diversity of methods, variable forms of learning; Assessment-free space for exercises; Facilitation of experiencing competence increase; Motivating environment; Development of the ability to accept criticism; Sufficient time for learning processes.

2.2 Competence Model

The ministry's competence model defines the desired learning outcomes by means of competences. It consists of the two dimensions "content" and "action", as depicted in fig. 1.

For a vocational school for information technology (HTL IT) the dimension "content" defines the contents that are taught in IT. These include software development, IT projects, and systems engineering as competence fields. For the case study in this paper the competence area industrial information technology (INIT) has been selected which belongs to the competence field "systems engineering". Other areas in this flied are: electrical engineering and electronics for IT, basics of informatics, operating systems, system integration and infrastructure, and decentralized systems.

The dimension "action" defines the cognitive effort. This is based on the revised Bloom's taxonomy [12] where the cognitive area of learning goals has been subdivided into six categories: Remember, Understand, Apply, Analyze, Evaluate, and Create. For Austria's educational standards, the ministry combined the first two categories [13]. The ministry's report "educational standards for vocational schools" [7] serves as a basis for the dimension "action", together with the more specific "educational standards for information technology" [14].

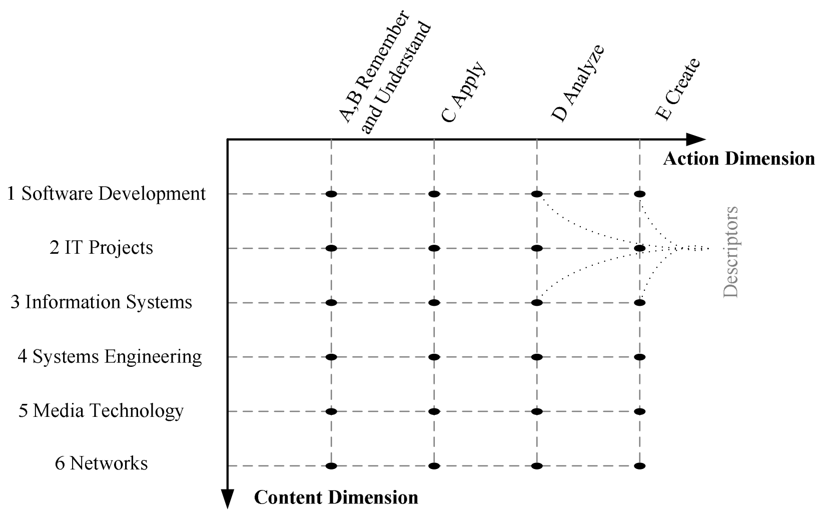


Fig. 1. Competence model of a vocational school with focus on IT

2.3 Descriptors

As can be seen in fig. 1, descriptors are the intersection points of the dimensions "action" and "content". They describe the expectation in the form of resulting competences. The ministry defined a coding syntax [13] for the descriptors in order to uniquely assign them to competence areas:

<school type><school focus> - <content> . <index> - <action>

For the school type vocational school for information technology (HTL IT), the code is IT, for the school focus "system engineering" the code is S. For basic topics the code for the school focus is generally omitted. The code for content is a numerical value indicating one of the main educational topics (in our case "systems engineering" is represented by 3). The index is used to distinguish multiple descriptors within an educational topic. For basics the index can run up to 100. Indices with a value over 100 indicate focus topics. The code for actions is a letter, where A represents the category "remember and understand". To illustrate this coding, a few descriptors [13] are listed below:

IT-5.1-B	I am able to explain the main scalar data types of a high-level programming language.
IT-3.23-D	I am able to assess an operating system and to choose an appropriate one for a given purpose.

According to [13] IT-5.1-B is assigned to the competence area software development. The student should understand and be able to describe the taught contents "scalar data types". IT-3.23-D represents the 23rd competence in the area of systems engineering at the level of analyzing.

3 Case Study at HTL Krems

For the case study the vocational school HTL Krems (http://www.htlkrems.ac.at) was chosen. The school's intent is to follow the state-of-the-art as closely as possible, which was demonstrated by their great interest in participating in this case study. The implementation of the developed didactic concept was done at the school's department for information technology.

The recently enacted new curricula [6] had to be applied for the first time in the school year 2011/2012 for all first grades in the vocational school.

3.1 HTL (Technical High Schools)

Vocational Schools. Vocational schools take five years (from year 9 to 13) and award a diploma that not only certifies professional qualifications but also allows university access. This type of school is deeply rooted in the Austrian education system with a fine tradition. According to [15] about half of the roughly 8000 diploma holder per year opt for tertiary education. Those who enter the job market are very welcome by employers [16].

In an international comparison it is hard to classify those diplomas. Within the ISCED (International Standard Classification of Education) [17] it could be classified between 4A and 5B [18]. To help with the international comparison of educational qualification, Europe developed the European Qualification Framework (EQF) [2] which defines attainment in the categories "knowledge", "skills", and "competencies" at eight levels.

Technical High School for Information Technology (HTL IT). Within the range of vocational schools there are the technical high schools, the so-called HTL, which come in several different flavors, e.g. for Information Technology (IT). These school types devote about half of the weekly hours to general subjects, and the remainder to IT specific ones.

Among the IT specific subjects, the competence field "systems engineering" covers the most weekly hours, namely 31 over the five years. Systems engineering comprises the competence areas: electrical engineering and electronics for IT, basics of informatics, operating systems, system integration and infrastructure, decentralized systems, and industrial information technology (INIT).

Competence Area INIT. For the case study, the competence area INIT was selected which is taught in years 4 and 5. The desired competences for those years [14] are listed in the tables 1 and 2.

Table 1. Descriptors for competence area INIT, competence levels A, B, and C

Content	**Remember/Understand**	**Apply**
Structure and functionality of SPC and micro controller systems	… know the basic structure and functionality of SPC and micro controller systems	… are able to apply SPC and micro controller systems for technical tasks in typical settings within INIT
Development of typical applications		
Industrial field bus systems	… know the structure of typical industrial field bus systems including their technologies and transmission methods	… are able to apply technologies and methods of industrial field bus systems

Table 2. Descriptors for competence area INIT, competence levels D and E

Content	**Analyze**	**Create**
Data processing, visualization, and communication of processes	… are able to plan, handle, document and supervise IT infrastructure for processes in INIT	… apply SPC and micro controller systems for development of networked and real-time systems in industrial scenarios, and to implement suitable mechanisms for process communication for these systems
Advanced topics in SPC and micro controller		
Development of specialized systems		

These competences are not assigned to a school year, neither in the curriculum nor in the educational standards. So it is obviously left to the teacher's responsibility.

3.2 Requirements

Based on the given constraints the following requirements for the teaching concept could be derived. These requirements are intended for the school years 4 and 5 (overall years 12, 13). In order to increase clarity, four clusters have been formed:

Table 3. Requirements for pedagogy and learning psychology

	Cluster A: Pedagogy and learning psychology
A1	Consider aspects of natural, reasonable and incidental learning
A2	Harness different learning speed
A3	Set clear teaching goals and identify contents

Table 4. Requirements for competence-orientated teaching

	Cluster B: Competence-orientated teaching
B1	Put student to action by employing activity-oriented methods
B2	Consider individual performance
B3	Foster intrinsic motivation and creativity
B4	Use a range of diverse teaching methods
B5	Leave room for learning processes and reflection
B6	Support networked thinking and multidisciplinarity

Table 5. Requirements for legal and institutional constraints

	Cluster C: Legal and institutional constraints
C1	Descriptor "remember and understand" for micro controllers
C2	Descriptor "remember and understand" for SPC
C3	Descriptor "remember and understand" for industrial field buses
C4	Descriptor "apply" for micro controllers
C5	Descriptor "apply" for SPC
C6	Descriptor "apply" for industrial field buses
C7	Descriptor "analyze" for processes
C8	Descriptor "create" for micro controllers
C9	Descriptor "create" for SPC
C10	Foster development of social and personal competence
C11	Consider school specific implementation of INIT
C12	Practical instruction with focus on deepening knowledge

Table 6. Requirements for embedded systems

	Cluster D: Embedded Systems
D1	Consolidation of previous knowledge
D2	Consider 3S (science, skills, state-of-the-art)
D3	Introduction to embedded C (essential for practical exercises)
D4	Modular and open topics for projects with flexible and creative solutions
D5	Impede plagiarism, foster individual attainment
D6	Use free software
D7	Use evaluation kits
D8	Flexible grading – grade is based on several elective contributions
D9	Assignments with diverse levels of difficulty and degree of fulfilment

3.3 Didactical Concept

Based on the above listed requirements the didactical concept for INIT for year 4 in a HTL IT has been derived and is presented in fig. 2.

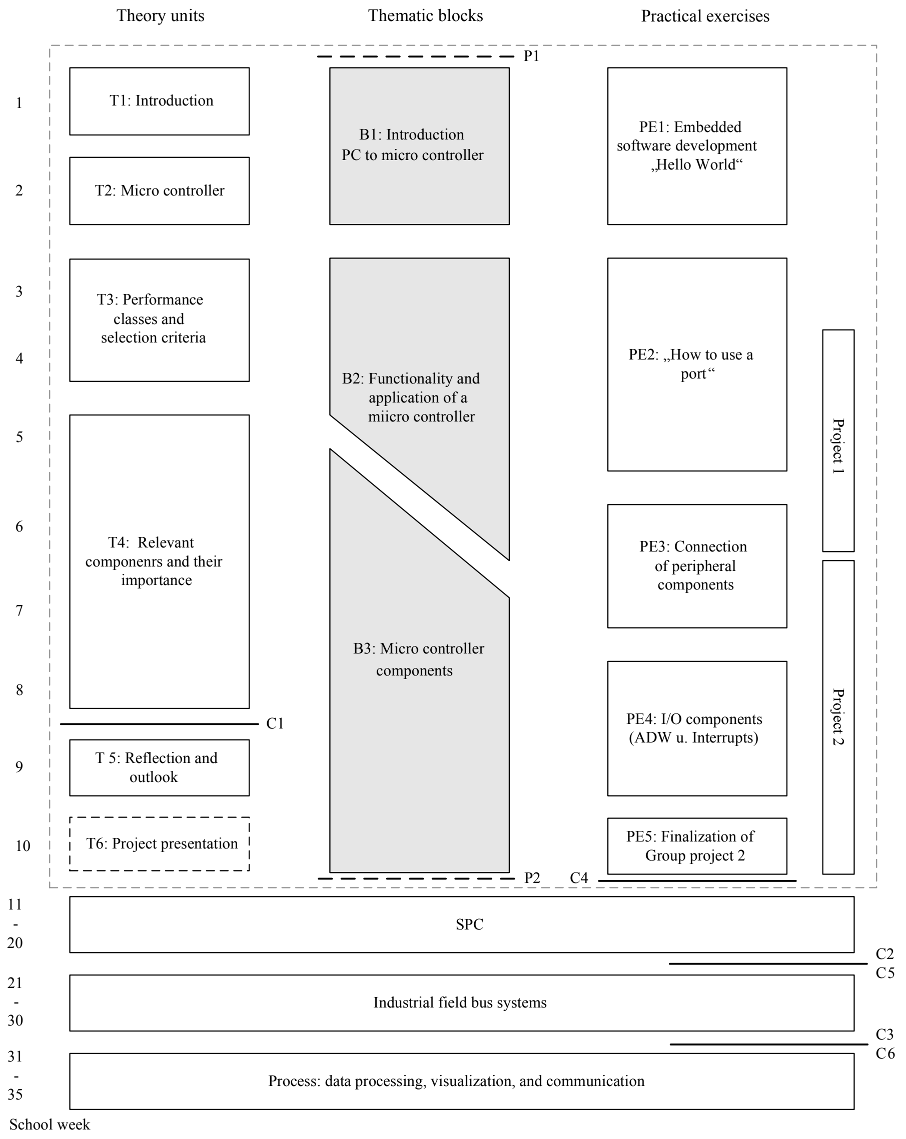

Theory units
Thematic blocks
Practical exercises
P1
1
2
3
4
5
6
7
8
9
10
T1: Introduction
T2: Micro controller
T3: Performance classes and selection criteria
T4: Relevant componenrs and their importance
C1
T 5: Reflection and outlook
T6: Project presentation
B1: Introduction PC to micro controller
B2: Functionality and application of a miicro controller
B3: Micro controller components
PE1: Embedded software development „Hello World“
PE2: „How to use a port“
PE3: Connection of peripheral components
PE4: I/O components (ADW u. Interrupts)
PE5: Finalization of Group project 2
Project 1
Project 2
P2
C4
11 - 20
SPC
C2
C5
21 - 30
Industrial field bus systems
C3
C6
31 - 35
Process: data processing, visualization, and communication
School week

Fig. 2. Teaching concept for Industrial Information Technology (INIT) in a HTL IT in year 4

As can be seen, an emphasis has been put onto micro controllers. Furthermore, not all requirements are taught in year 4. The remaining requirements are rather taught in year 5.

4 Evaluation and Discussion

4.1 Evaluation

The change of paradigm towards competence orientation proved challenging for both, teachers and students. Apart from obstacles in administration and organization, the change to student-centered teaching makes great demands on teachers. Teachers are used to being the center and to immediately react to fuzziness in the learning process. This is in conflict with student-centered teaching which should allow sufficient space for the student's own exploration. Aside from necessary self-discipline, phases for correction of the instructional process are required. This has to be adequately considered during planning.

Concerning methods which reflect the student-centeredness (e.g. group puzzle[9]) it must be reported that students do not necessarily appreciate them. Especially students with a performance level well above average moan about the lack of balance regarding the cost-benefit ratio. They spend a lot of time instructing others with little gain for themselves. These students prefer ex-cathedra teaching because of the condensed information supply. Nevertheless, they appreciate the social aspect of the new teaching methods. It would be interesting to see a long-term evaluation for differently gifted groups of students.

With regard to the implementation it may be noted, that for grade 4 in HTL (overall grade 12) competencies are limited by the level "Apply". Even for that, the intended period of 11 weeks of instruction has been overly ambitious. More realistically, 15 weeks should be planned. The general division of the topics into three major blocks proved useful. It turned out that a major difficulty lies in the strict two hour limit for practical exercises. Especially, if specialized hardware was used which had to be reassembled every time from scratch, valuable time was lost. The students, however, did not complain extensively as the reassembly reinforced the general understanding of the circuits. Alternatively, pre-assembled hardware with peripheral components could be used (e.g. buttons, LCD).

In order to accommodate for individualization within the practical exercises, individual learning goals have been agreed upon. Especially students who work faster chose their own additional tasks. Interestingly, this turned out to be motivating for other students, as well. A possible reason for this effect could be the inspiring demonstration of the creative possibilities of the employed evaluation kit. Generally, it could be observed that the chosen contents and exercises were conducive to the intrinsic motivation of the students. It can be concluded that the employment of suitable hardware kits seems essential.

Concerning multidisciplinary aspects, the competence areas "IT projects" has been successfully coupled with the competence area "project management". For "project management" which is taught 4 hours per week, the students had to accomplish projects with contents from INIT. Results show that the students' choices for INIT contents were highly ambitious and far beyond the content requirements.

In summary it can be concluded, that the presented didactical concept served as a solid basis for developing the desired competencies up to the level "apply". There is potential for improvement concerning the theory part "micro controller" which was too ambitious for one teaching unit.

4.2 Discussion

The initially posed central questions can be answered as follows:

Q1: In which way should teaching be designed as to ensure the educational standards for the competence field INIT?

The presented concept had the educational standards as requirements, and therefore shows a possible implementation strategy, but definitely not the only one. School specific requirements were considered, as well. These might differ from other schools, and again represent one of many possible implementations. The distinction into theory and practice blocks was a school requirement, but proved useful. In order to support the combination of theory and practice, the contents have been subdivided into three major topics. Furthermore, the theory sessions have been designed in an open way as to facilitate constructivist learning processes. The application of activity-oriented teaching methods has been favored, as well as conceptual knowledge. Contrary to the theory session, the practical exercises (workshops) were more restrictive with respect to the activity and expected results.

It has to be mentioned that there is only a single example in [14] for the competence area INIT. This is problematic since the educational standards are fairly general and leave plenty of room for interpretation. Furthermore, it becomes evident that both, the curriculum as well as the educational standards, are rather ambitious in view of the available time frame. Therefore it should be discussed as to what extent the area of SPC correlates with the profile of a HTL IT in general. Instead, it might be more advisable to emphasize micro controllers.

Q2: In which way can be ensured that individual learning goals are attained in combination with personal and social competencies?

The didactical concept is based on the idea of achieving personal and social skills mainly during the theory sessions. In contrast, the practical exercises focus on individual attainment as they demand the student's own effort to find solutions. Later on, the practical exercises include group work as well, where social interaction is a necessary part of the task, but personal outcomes are defined, as well.

Q3: To what extent does a specific school influence the implementation of the new curricula with respect to activity-oriented teaching methods?

The school specific requirement to distinguish theory and practice blocks bears advantages as well as disadvantages. Most students appreciate the concept "Theory deals with general concepts, while practical exercises deal with a specific technology." Problems mainly arise from the organizational limit of 50-minutes sessions. The consequences of splitting tasks over several sessions remain to be seen. Concerning activity-oriented teaching methods, the average class size seems to be challenging.

In general, there is little flexibility for schools to influence class sizes; a deviation from 50-minutes sessions is possible, but requires a huge organizational effort.

The requirement for sufficient time for reflection had to be planned ahead, but does not pose a problem in general. Also multidisciplinary aspects have to be planned in advance as it requires intensive coordination efforts by the participating teachers.

Q4: In which way should teaching be designed as to foster creativity and intrinsic motivation?

The open approach to theory sessions provides space for personal development. Activity-oriented methods are generally conducive to intrinsic motivation, as students tend to acquit themselves well when asked to actively contribute.

An evaluation kit was used for the practical exercises. Thereby the students could test their solutions on physical hardware which could be adjusted to the individual tasks. Both, testing and individualization increased the motivation. There was a choice of possible tasks, with the possibility of individual flair. Tasks were formulated as questions or hypotheses with the intent to arise the student's interest.

Furthermore, it can be stated that, due to the complexity of teaching competences, teachers are required to exhibit an appropriate willingness, as well as sufficient skills for this approach. This starts with a fundamental acceptance of competence orientation.

With the definition of descriptors, it seems that essential steps towards educational standards for IT have been made, while some details are still missing. This can be derived from the fact that the descriptors in [13] for the educational basics are a draft version. Also, while the competence model in [13] uses five dimensions, the one in [14] applies six dimensions. A synthesis of these two documents [13, 14] is advisable. Additionally, to implement these educational standards, the completion of the prototypical instruction examples in [14] is of great importance. Furthermore, the coding syntax of [13] was not applied in [14], which only represents a minor problem.

As stated by Reinbacher [19] as well, time will prove whether this approach to competence orientation in vocational schools actually is an important step towards an increase in quality. To achieve this long-term goal it is indispensable to provide a fully developed competence model with descriptors that reflect the state-of-the-art.

In any case, the comparison between educational programs should be more feasible than before.

References

1. Educational standards for vocational schools, Austria, `http://www.bildungsstandards.berufsbildendeschulen.at/en/leitideen.html`
2. EQF – European Qualifications Framework (2008), `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32008H0506%2801%29:EN:NOT`

3. Linck, B., Magenheim, J., Nelles, W., Neugebauer, J., Ohrndorf, L., Schaper, N., Schubert, S.: Empirical refinement of a theoretically derived competence model for informatics modelling and system comprehension. In: Proceedings of IFIP-Conference Addressing Educational Challenges: The Role of ICT (AECRICT), Manchester Metropolitan University, July 2-5 (2012)
4. Grundsätze und Standards für die Informatik in der Schule. LOG IN Nr. 150/151 (2008)
5. Weinert, F.E.: Concept of Competence: A Conceptual Clarification. In: Rychen, D., Salganik, L. (eds.) Defining and Selecting Key Competencies. Hofgrefe & Huber (2001)
6. Curricula for HTL (Bundesgesetzblatt Nr. 300/2011), Austria, `http://www.htl.at/fileadmin/content/Lehrplan/HTL_VO_2011/BGBl_II_Nr_300_2011.pdf`
7. Bildungsstandards in der Berufsbildung – Projekthandbuch, `http://www.bildungsstandards.berufsbildendeschulen.at/fileadmin/content/bbs/Handbuch_BIST_25.03.2013.pdf`
8. Antonitsch, P.: On Competence-Orientation and Learning Informatics. In: Bezakova, D., Kalas, I. (eds.) Proceedings of ISSEP 2011, Informatics in Schools: Situation, Evolution and Perspectives, Selected Papers (on CD). Library and Publishing Centre Comenius University, Bratislava (2011)
9. Reich, K. (ed.): Methodenpool, `http://methodenpool.uni-koeln.de`
10. Lersch, R.: Kompetenzfördernd unterrichten – 22 Schritte von der Theorie zur Praxis. Pädagogik 12 (2007)
11. Fritz, U.: Kompetenzorientiertes Unterrichten – Grundlagenpapier. bm:ukk (ed.) (2011), `http://www.berufsbildendeschulen.at/fileadmin/content/bbs/KU/Grundlagenpapier_KU_Maerz2011.pdf`
12. Anderson, L.W., David, R., Krathwohl, D.R., et al. (eds.): A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Allyn & Bacon, Pearson Education Group, Boston (2001)
13. Höhere technische Lehranstalt für Informationstechnologie – Bildungsstandards in der Berufsbildung, `http://www.berufsbildendeschulen.at/fileadmin/content/bbs/AGBroschueren/IT_V8.pdf`
14. Höhere technische Lehranstalt für Informationstechnologie – Fachrichtungsstandard, `http://www.bildungsstandards.berufsbildendeschulen.at/fileadmin/content/bbs/AGBroschueren/Fachrichtungsstandard_IT_08-09-2011_Druck.pdf`
15. Schneeberger, A.: Mittelfristige Perspektiven der HTL - Erhebung und Analysen zur Sicherung und Weiterentwicklung der Ausbildungsqualität (Juni 2008), `http://www.ibw.at/html/rb/pdf/rb_43_schneeberger.pdf`
16. Schneeberger, A., Petanovitsch, A.: HTL und Qualifikationsbedarf der Wirtschaft - Analysen zur Arbeitsmarktlage und europäischer Vergleiche (March 2009), `http://www.ibw.at/de/pruefungsunterlagen?page=shop.getfile&file_id=306&product_id=288`
17. UNESCO. International Standard Classification of Education (1997), `http://www.unesco.org/education/information/nfsunesco/doc/isced_1997.htm`
18. Schneeberger, A.: Internationale Einstufung der österreichischen Berufsbildung - Adäquate ISCED-Positionierung als bildungspolitische Herausforderung (May 2010), `http://www.ibw.at/de/pruefungsunterlagen?page=shop.getfile&file_id=404&product_id=365`
19. Reinbacher, T.: Lehre von Software-Verifikation in der Berufsbildung unter dem Aspekt von Bildungsstandards. Master thesis, Vienna University of Technology (2010)

Measuring Student Competences in German Upper Secondary Computer Science Education

Jonas Neugebauer[1], Peter Hubwieser[3], Johannes Magenheim[1], Laura Ohrndorf[2], Niclas Schaper[1], and Sigrid Schubert[2]

[1] University of Paderborn, D-33102 Paderborn, Germany
`http://ddi.uni-paderborn.de`
[2] University of Siegen, D-57068 Siegen, Germany
`http://www.die.informatik.uni-siegen.de`
[3] Technische Universität München, D-80335 München, Germany
`http://www.ddi.tum.de`

Abstract. Within the interdisciplinary research project "Measurement Procedure for Informatics in Secondary Education (MoKoM)", conducted at the Universities of Paderborn and Siegen with help from Peter Hubwieser, Technical University of Munich, researchers aimed to develop a theoretically and empirically sound competence model for the domains of system comprehension and system modelling, alongside an evaluated measurement instrument to assess competences of students in upper secondary computer science education in German schools. The competence model was thoroughly developed by conducting several theoretical and empirical steps. Based on the model measurement items were constructed and compiled into an instrument with 87 items of varying complexity. The instrument was divided into six booklets and distributed to over 800 computer science students in three German states. The returned 583 tests were analysed by means of the multidimensional item response theory, to assess the item difficulty on the one hand and the student abilities on the other. The results are used in several ways: to evaluate the competence model, to revise the measurement instrument and to define proficiency levels in a competence level model by using the method of scale anchoring.

Keywords: Competence Modeling, Competence Measurement, Informatics System Application, Informatics System Comprehension, Informatics Modelling, Secondary Education.

1 Motivation

The international discussion of educational standards triggered various projects of competence modeling. For the majority of subjects normative competence models were developed, which structure the competences of learners into dimensions and sub-dimensions. But only for the main subjects (languages, mathematics) empirically refined competence models were officially presented. For these concepts, empirical research and standardization of measurement are indispensable. In the area of Didactics of Informatics the interdisciplinary research project “Measurement Procedure for Informatics in Secondary Education (MoKoM)” developed such a model for German secondary education. The MoKoM project started in 2009 and was funded by the German Research Foundation (DFG). At that time no competence model for upper secondary computer science education in Germany existed. Our research focused on two selected parts of the curriculum: informatics system comprehension (ISC) and object-oriented modeling (OOM). This selection offered the opportunity to investigate the levels of application and the complexity of media functions of the informatics system in the learning process of our target group. The specific objectives of our research to measure ISC and OOM competences are:

1. To create a theoretically derived competence model (TCM);
2. To refine the TCM into an empirically derived competence model (ECM);
3. To design a measurement instrument on the basis of the ECM;
4. To evaluate the validity of the test instrument in a pre-test;
5. To apply the test instrument in a comprehensive test sample to examine the psychometric quality, especially the validity according to the developed competence model;
6. To develop a competence level model with descriptive proficiency levels.

In this paper we will summarize the development of our measurement instrument and how it was applied to the specified target group. Then we will describe the analysis of the gathered data and how the results can be interpreted. Finally, we will outline the steps required to accomplish the last objective of creating a competence level model.

2 Development of a Competence Measurement Instrument

The MoKoM competence model was created in two steps: The theory-driven analysis of curricula and the refinement by means of empirical methods.

The sources for the analysis were carefully selected with the goal to consider the most influential national and international curricula and syllabi. This includes the ACM/IEEE Computing Curriculum 2001 [23] and the ACM Model Curriculum for K-12 Computer Science [24]. The resulting model consisted of three cognitive dimensions: K1 *Basic Competences* (including system application, system comprehension and system development), K2 *Informatics views* and K3 *Complexity*, and a non-cognitive dimension K4 [15].

To refine this model, 30 expert interviews were conducted with experienced persons from the fields of computer science education, computer science in general and computer science teachers. This was done using the Critical Incident Technique [3], presenting each interviewee with four problem scenarios and asking them how they would solve the given task. The interviews were recorded, transcribed and then analysed using the qualitative content analysis according to Mayring [18]. The results were used to refine and restructure the theoretical model [13, 15, 16]. This led to the final MoKoM competence model containing five competence dimensions: K1 *System Application*, K2 *System Comprehension*, K3 *System Development*, K4 *Dealing with* *system complexity* and K5 *Non-cognitive competences* (for a detailed description of the model see [14]).

Based on the empirical competence model the test instrument was developed following the principles of Situational Judgment Tests (SJT) [26]. We intently regarded experiences gained in large competence measurement studies like TIMMS [17], PISA [2] and DESI [12]. Furthermore we followed expert knowledge from the field of work and organizational psychology. Based on detailed competence descriptions, tasks for every single competence item were created. After this, the answering format was chosen following the cognitive level (e.g. multiple choice or open questions). Multiple tasks were grouped in a context suiting the target group to help the test takers to identify with the tasks. To allow an objective and balanced evaluation, a comprehensive grading manual was created alongside the test items. This contained different sample solutions as well as approaches to grade answers [21].

The test instrument was initially used in a preliminary test with students from local secondary schools. In addition, student computer science teachers from didactical courses at the universities of Paderborn and Siegen were asked to review the instrument. The main issues found during this pre-test were ambiguous wording and careless mistakes on the one hand and the difficulty of applicability of the tasks on the other. Rewriting or extending the context of the tasks could easily fix the former. One common example is the use of specific technical terms, which might not be familiar to some of the test takers. One common problem was the misjudgment of the complexity of a task and the resulting longer durations to solve them. To circumvent this, we simplified some tasks by changing the format (e.g. creating a multiple-choice question instead of an open question).

Due to the large amount of items, the test instrument was not applicable in a classroom setting. Usually a German computer science class lasts 45 minutes, with combining two lessons being common practice. To adapt the instrument to a 90-minute timeslot, the items were divided into six blocks. Next, six booklets were compiled from three item block each. Every one of these can be answered in about 80 minutes. Together with a non-cognitive questionnaire on attitudinal, motivational and volitional competences, the test can be accomplished in two school lessons. The application of such arrangement called "matrix design" is possible due to the use of the Item Response Theory to analyse the test results (see section 4). Though not all students answer every task due to not having them in their booklet and thus produce a lot of "missing values" in the final data, the IRT allows for the estimation of student abilities in combination with the overall item difficulty. This methods allows for coherent results even if the students worked on different subsets of items [5, 22].

3 Applying the Measurement Instrument

To get a large enough population we asked teachers from North Rhine-Westphalia to participate. 17 teachers showed interest and contributed 26 classes with 522 students. Since we couldn't expect all booklets to be returned, we also asked 5 teachers from Berlin, Hesse and Lower Saxony, who added 6 classes with a total of 82 students. In Bavaria we had contacted 6 teachers at different gymnasiums and asked their students to complete our test booklets. 244 students from 11 different classes (6 of grade 10 and 3 of grade 11) took part in the test. All this data was collected in the context of the compulsory subject “Informatics” that had started at Bavarian Gymnasiums in 2004. A detailed overview of this subject was presented in [10]. According to the curriculum, the current learning content of all responding students was focused on object-orientation. The students of grade 10 had just finished the first half of introduction into object-oriented programming, while those of grade 11 had already implemented recursive data structures like lists and trees and were starting to learn the basic concepts of software engineering and project management. The learning objectives of the object-oriented course were described in [9]. In the grades before 10, the students had learned to describe standards software by object-oriented models, to program robots, to implement data flow diagrams on spread sheets and to design relational data bases (see [8, 9]).

The test was carried out as a pen- and paper-test. The booklets were printed and sent to teachers who volunteered to apply them in their classes. To prevent the students from cheating, each teacher received two to three different booklets to distribute, rather than just one. From more than 800 send out tests we received 583 completed and evaluable booklets back. 86% male and 14% female students with an average age of 17.53 years worked on the tasks. 17% of them had an immigrant background. Their self-assessed proficiency in computer science on a scale from 1 to 6 averaged at 2.65 points. They had participated in computer science classes for a mean of 3.46 years. Only 3.34% had dropped the subject in the interim.

The coding of items was done according to the aforementioned grading manual. While the manual intended for the items to be coded as aggregate scores, this approach proved to be too inaccurate, since afterwards we were unable to reconstruct what exactly was the problem within the item. Therefore, each task was separated into multiple sub-items that could be graded as dichotomous codes (wrong/correct). Since this was not possible for some of the more complex tasks, those items still were coded with aggregate scores. Missing values had to be treated in two ways: items that were missing from the booklet a student worked on were coded as normal missing value, while items that were present, but hadn’t been worked on, were coded as ‘99’ to reliably separate them from each other.

4 Analyzing the Test Data

The gathered data was analyzed according to the Multidimensional Item Response Theory (MIRT). The main goal was to evaluate the competence model and the measurement instrument. To do so, several different IRT models were used to analyze the empirical data and the results were compared to assess the best fitting model.

4.1 Multidimensional Item Response Theory

Compared to classical test theory, IRT models assume that personality traits cannot be measured directly and test results can only be interpreted as an indicator for the existence and intensity of such a trait. Therefore, IRT models differentiate between latent variables, that can’t be measured directly, but influence the response to a test item, and manifest responses that are assumed to be the observable manifestations of the latent traits [6, 20, 22]. Thus, the ability of the tested person can be inferred from the responses. Furthermore it is assumed that any subject has a certain probability to answer any item right or wrong. The difficulty of the item and the ability level of the subject determine this probability. For example, a student with a high level regarding a certain competence dimension is more likely to answer an item with a medium difficulty in that same dimension, than a student with a lower level. Prominent large scale studies like PISA [2] and TIMSS [17] used IRT scaling methods to evaluate student competences.

IRT has several advantages for the assessment of competences. For once, the estimation of the item difficulties and student abilities does not require for every participant to work on every task of the test instrument. This allows for the use of a matrix design with different booklets as described above. Furthermore, the estimated parameters can be interpreted on the same scale and easily related to each other [22].

Since competence structures are complex constructs, they often times result in multidimensional competence models. In our case this applies to the cognitive dimensions K1 to K4 with the additional non-cognitive dimension K5. To evaluate the dimensionality of the empirical data, multidimensional IRT models can be utilized, which assume multiple latent variables (one per dimension) cause the deterministic responses to a test. Furthermore, MIRT allows for the comparison of different models, by analyzing the conformity of the theorized model to the empirical data [20].

Because of the utilized matrix design the data had to be recoded to deal with the missing values. There are two possible options to replace the aforementioned '99' codings: substitute them with a 0 (speed test) or as a missing value (power test). There are reasonable arguments for both variants. Therefore, we analyzed both. Since the conclusions are very similar though, this article will concentrate on the results of the speed option. To calculate the MIRT analysis we used ACER ConQuest Version 2 [27].

4.2 Evaluating the Competence Dimensionality

To evaluate the structure of the competence model, we analyzed four different IRT models with one to four assumed dimensions respectively. Since the test items were crafted with the intent to test for one specific competence, a between-item multidimensionality model was used in all cases [7]. Because not all items could be coded as dichotomous responses, the partial credit model was applied to analyze dichotomous and polytomous data alike. Starting with the one-dimensional model, for which it was assumed that all items loaded on the same latent trait, every model added one additional dimension by theoretical reasoning along the competence model. The assignment of the competence dimensions to the IRT models can be seen in table 1.

Table 1. Final deviance, estimated parameters and reliability for evaluated models

Model	**Final Deviance**	**Estimated Parameters**	**Reliability for dimension 1 to 4 (if available)**
1-Dim	87379.09538	316	0.872 (K1,K2,K3,K4)
2-Dim	86695.99173	319	0.831 (K1) / 0.831 (K3)
3-Dim	86403.83657	323	0.749 (K1) / 0.806 (K2,K4) / 0.812 (K3)
4-Dim	85891.85717	328	0.779 (K1) / 0.763 (K2) / 0.861 (K3) / 0.759 (K4)

To compare the models, the final deviance – an indicator of how well the empirical data fits the IRT model - and count of estimated parameters reported by ConQuest can be used [20, 22, 27]. Usually, both parameters should be as low as possible. If it is not possible to choose the better model by comparing the values alone (because one value is lower, while the other is bigger than the second models), a Chi-Square-Test can be calculated, using the difference in deviance and the difference in estimated parameters as the degrees-of-freedom. If the result is significant, the model with the smaller deviance is selected. Otherwise the model with the lower amount of estimated parameters is deemed the better one. The parameters for each evaluated model can be seen in table 1.

Table 2. Chi-Square statistics for model comparisons with difference in deviance and difference in estimated parameters as degrees of freedom

	2-Dim	**3-Dim**	**4-Dim**
1-Dim	$\chi^2_{(3)}$=683.1, p<.001	$\chi^2_{(7)}$=975.26, p<.001	$\chi^2_{(12)}$=1487.24, p<.001
2-Dim		$\chi^2_{(4)}$=292.15, p<.001	$\chi^2_{(9)}$=804.13, p<.001
3-Dim			$\chi^2_{(5)}$=511.98, p<.001

Since with increasing dimensions the deviance decreases and the number of parameters increases, a chi-square-test was calculated for every combination of models (see table 2). Every time the result was statistically significant and since the models with a higher number of dimensions have a lower deviance, it can be assumed that they better match the empirical data than the models with fewer dimensions. Thus, the four-dimensional model has the best model fit overall.

4.3 Item Fit and Reliability

ConQuest calculates the EAP/PV reliability for each dimension, which can be compared to Cronbach's Alpha [22]. Table 1 shows the reliability for all dimensions in each model. All values exceed 0.7 and can be considered acceptable. Unfortunately, the reliability index decreases for the four-dimensional model with only the third dimension having a value well over 0.8. The current measurement instrument seems to measure latent trait levels for this dimension with notably more certainty than for the other dimensions.

To further evaluate the models, the item fit for individual items can be examined. The fit compares the predicted probabilities for each item within the model with the observed responses. To do this, ConQuest calculates the weighted mean squares (wMNSQ), which are expected to be 1 for perfectly fitting items. The wMNSQ for a good fitting item should fall between 0.8 and 1.2, and the corresponding t-values, that should not be greater than 1.96 [1, 27]. Furthermore, the discrimination parameter shows how an item correlates to the overall test results. With the discrimination close to 0, an item may not be useful to differentiate between students with high levels of a trait and those with low levels. Values between 0.4 and 0.7 are considered good while values above 0.3 can be considered as acceptable [19].

The data for all models (see table 3) showed a good item fit overall, but the percentage of unfit items increased for models with more dimensions, from below 1% (2 items out of 292) for the one-dimensional to 4.7% (14 items) for the four-dimensional model. In addition, the number of items that might have a bad fit according to the t-values increased from 27 to 37 items. Unfortunately the discrimination parameters are not very good for a large part of the items. Just 22.6% (66 items) are above the 0.4 threshold and even if we adjust the point at which an item is considered to have a too small discrimination to 0.3, roughly 43.8% (128 items) remain under that line. Only one item had a negative discrimination, which was close to 0.

Table 3. Range of mean squares, t-values and discrimination values for all models

Model	**wMNSQ**	***t***	**Discrimination**
1-Dim	0.86≤wMNSQ≤1.3	-2,9≤t≤4.4	-0.04≤Disc. ≤0.58
2-Dim	0.77≤wMNSQ≤1.42	-4.1≤t≤5.2	-0.04≤Disc. ≤0.58
3-Dim	0.76≤wMNSQ≤1.42	-4,2≤t≤5.2	-0.04≤Disc. ≤0.58
4-Dim	0.65≤wMNSQ≤1.42	-5.7≤t≤5.3	-0.04≤Disc. ≤0.58

The low discrimination necessitates a throughout examination of the items in the measurement instrument and how they fit to their corresponding dimension.

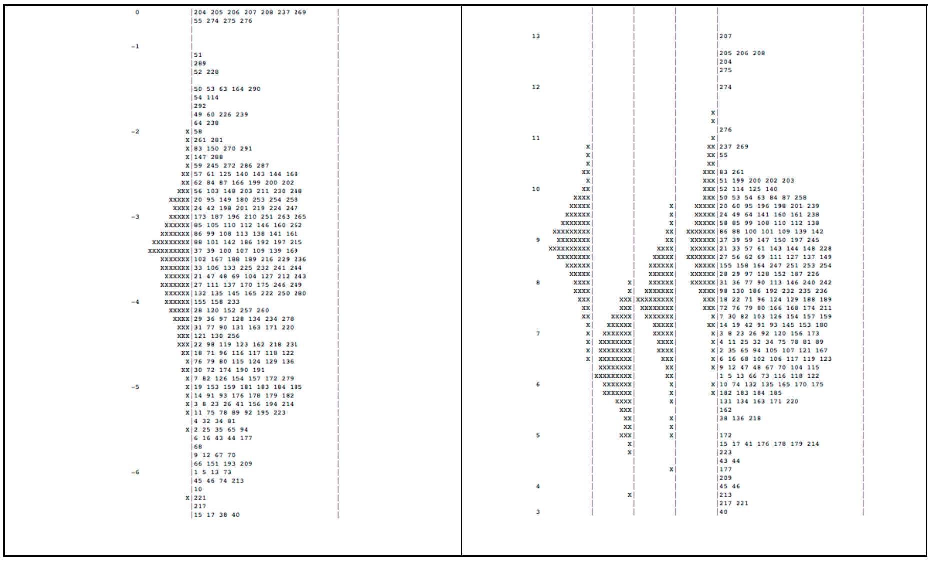
4.4 Difficulty Parameters and Latent Abilities

The main goal of IRT analysis is the estimation of two parameters: the item difficulty, that denotes the probability of answering an item correct given a certain level of the measured construct, and person parameters, that assess the level of the latent trait for individual students. One advantage of IRT analysis is that both estimates can be arranged on the same scale and easily compared. The item-person-map for each model visualizes the item difficulties on the right, by ordering them from more difficult (top) to less difficult (bottom), and the latent trait levels on the left (grouping persons with the same values together). Ideally, the item difficulties should be well dispersed around the mean, having the most items in the medium difficulty range, but also providing items with high and low difficulties [20]. Additionally, the latent traits are separated by dimension. Table 4 shows the maps for the on- and four-dimensional models. As can be seen, the item difficulties are well distributed along the axis, though there are some observations to be made.

First, there are some outliers in the upper part of each map. This indicates, that some items are way to difficult for the targeted student groups, since no person was estimated to have a high enough proficiency to solve these items with an adequate probability.

Second, the latent traits in the different dimensions are somewhat uneven dispersed. While the one-dimensional model indicates, that the overall difficulty of the test matches the ability of the population, the four-dimensional model reveals, that only the third dimension can be considered well matched. Dimension 1 and 4 lack items in the upper difficulty range, while dimension 2 necessitates less difficult items to adequately assess its competences.

Table 4. Overview of the estimated item parameters for the one- and four-dimensional model



```
 0              |204 205 206 207 208 237 269         |
                |55 274 275 276                      |
                |                                    |
                |                                    |
-1              |                                    |
                |51                                  |
                |289                                 |
                |52 228                              |
                |                                    |
                |50 53 63 164 290                    |
                |54 114                              |
                |292                                 |
                |49 60 226 239                       |
                |64 238                              |
-2             X|58                                  |
               X|261 281                             |
               X|83 150 270 291                      |
               X|147 288                             |
               X|59 245 272 286 287                  |
              XX|57 61 125 140 143 144 168           |
              XX|62 84 87 166 199 200 202            |
             XXX|56 103 148 203 211 230 248          |
           XXXXX|20 95 149 180 253 254 258           |
            XXXX|24 42 198 201 219 224 247           |
-3         XXXXX|173 187 196 210 251 263 265         |
          XXXXXX|85 105 110 112 146 160 262          |
         XXXXXXX|86 99 108 113 138 141 161           |
       XXXXXXXXX|88 101 142 186 192 197 215          |
      XXXXXXXXXX|37 39 100 107 109 139 169           |
         XXXXXXX|102 167 188 189 216 229 236         |
         XXXXXXX|33 106 133 225 232 241 244          |
          XXXXXX|21 47 48 69 104 127 212 243         |
         XXXXXXX|27 111 137 170 175 246 249          |
          XXXXXX|132 135 145 165 222 250 280         |
-4        XXXXXX|155 158 233                         |
           XXXXX|28 120 152 257 260                  |
            XXXX|29 36 97 128 134 234 278            |
             XXX|31 77 90 131 163 171 220            |
             XXX|121 130 256                         |
             XXX|22 98 119 123 162 218 231           |
              XX|18 71 96 116 117 118 122            |
               X|76 79 80 115 124 129 136            |
              XX|30 72 174 190 191                   |
               X|7 82 126 154 157 172 279            |
-5             X|19 153 159 181 183 184 185          |
               X|14 91 93 176 178 179 182            |
               X|3 8 23 26 41 156 194 214            |
               X|11 75 78 89 92 195 223              |
                |4 32 34 81                          |
               X|2 25 35 65 94                       |
                |6 16 43 44 177                      |
                |68                                  |
                |9 12 67 70                          |
                |66 151 193 209                      |
-6              |1 5 13 73                           |
                |45 46 74 213                        |
                |10                                  |
               X|221                                 |
                |217                                 |
                |15 17 38 40                         |
```

```
              |         |         |       |                                  |
              |         |         |       |                                  |
13            |         |         |       |207                               |
              |         |         |       |                                  |
              |         |         |       |205 206 208                       |
              |         |         |       |204                               |
              |         |         |       |275                               |
              |         |         |       |                                  |
12            |         |         |       |274                               |
              |         |         |       |                                  |
              |         |         |       |                                  |
              |         |         |      X|                                  |
              |         |         |      X|                                  |
              |         |         |       |276                               |
11            |         |         |      X|                                  |
             X|         |         |     XX|237 269                           |
             X|         |         |     XX|55                                |
             X|         |         |     XX|                                  |
            XX|         |         |    XXX|83 261                            |
             X|         |         |    XXX|51 199 200 202 203                |
10          XX|         |         |    XXX|52 114 125 140                    |
          XXXX|         |         |    XXX|50 53 54 63 84 87 258             |
         XXXXX|         |        X|  XXXXX|20 60 95 196 198 201 239          |
        XXXXXX|         |        X|  XXXXX|24 49 64 141 160 161 238          |
       XXXXXXX|         |        X|  XXXXX|58 85 99 108 110 112 138          |
     XXXXXXXXX|         |       XX|XXXXXXX|86 88 100 101 109 139 142         |
9     XXXXXXXX|         |       XX| XXXXXX|37 39 59 147 150 197 245          |
    XXXXXXXXXX|         |     XXXX| XXXXXX|21 33 57 61 143 144 148 228       |
      XXXXXXXX|         |    XXXXX|XXXXXXX|27 56 62 69 111 127 137 149       |
        XXXXXX|         |   XXXXXX|  XXXXX|155 158 164 247 251 253 254       |
         XXXXX|         |   XXXXXX| XXXXXX|28 29 97 128 152 187 226          |
8         XXXX|        X|   XXXXXX| XXXXXX|31 36 77 90 113 146 240 242       |
          XXXX|        X|  XXXXXXX|   XXXX|98 130 186 192 232 235 236        |
           XXX|      XXX|XXXXXXXXX|    XXX|18 22 71 96 124 129 188 189       |
            XX|      XXX| XXXXXXXX|    XXX|72 76 79 80 166 168 174 211       |
            XX|    XXXXX|  XXXXXXX|      X|7 30 82 103 126 154 157 159       |
             X|   XXXXXX|    XXXXX|     XX|14 19 42 91 93 145 153 180        |
7            X|  XXXXXXX|    XXXXX|      X|3 8 23 26 92 120 156 173          |
             X| XXXXXXXX|     XXXX|      X|4 11 25 32 34 75 78 81 89         |
             X| XXXXXXXX|     XXXX|      X|2 35 65 94 105 107 121 167        |
             X| XXXXXXXX|      XXX|      X|6 16 68 102 106 117 119 123       |
              | XXXXXXXX|       XX|      X|9 12 47 48 67 70 104 115          |
              |XXXXXXXXX|       XX|       |1 5 13 66 73 116 118 122          |
6             |  XXXXXXX|        X|      X|10 74 132 135 165 170 175         |
              |  XXXXXXX|        X|      X|182 183 184 185                   |
              |     XXXX|        X|       |131 134 163 171 220               |
              |      XXX|         |       |162                               |
              |       XX|        X|       |38 136 218                        |
              |       XX|        X|       |                                  |
5             |      XXX|        X|       |172                               |
              |        X|         |       |15 17 41 176 178 179 214          |
              |        X|         |       |223                               |
              |         |         |       |43 44                             |
              |         |        X|       |177                               |
              |         |         |       |209                               |
4             |         |         |       |45 46                             |
              |        X|         |       |213                               |
              |         |         |       |217 221                           |
3             |         |         |       |40                                |
```

5 Discussion

The results from the MIRT analysis show promise for our further work, though it also points to some areas that need to be improved. In this section we will discuss the implications for our research.

5.1 Analyzing the Results

The results of the model comparison described in section 4.2 are a strong indication that the four dimensional structure of the (cognitive part of the) MoKoM competence model is an appropriate representation of the competences in question. The overall results of the MIRT analysis confirm the multidimensional competence structure from our model, since it has the best fit to the empirical data. The thoroughly conducted steps to construct the competence model (see section 2) seem to have yielded a plausible result. Especially the third dimension K3 *System Development* appears to be well measured with regards to item difficulty and item fit. This is also evident in the reliability of the four dimensions, which is the highest for the third. The item fit denoted by the weighted mean squares and the t-values identifies exceptions in the item pool, that don't fit the empirical data as well as the other items and will have to be revised.

5.2 Implications for the Measurement Instrument

From the comparison of the item difficulties and latent abilities in section 4.4 some consequences for the overall instrument have been drawn. The overall distribution of the estimates shows which dimensions have to be reinforced with easy or hard items. This can either be done by making some items for that dimension easier/harder, or by replacing items with a low model fit. To adjust item difficulties, the definition of criteria that are known to influence the item difficulty can be utilized. Since such a set of criteria was developed as a necessary step for the characterization of proficiency levels (see section 6), it should be possible to adjust the item difficulty within certain parameters.

An examination of items with a low model fit shows a set of items that belong to the same task in the measurement instrument. Within this task the students have to structure the problem field of creating an online tool in smaller problems and select the best tools for solving these. During the coding process of the student responses these tasks were coded as dichotomous, one per correctly identified sub-problem. The low fit of theses items might be a result of the close connection between the responses to each individual item. For example, if one task requires the students to split a task into two sub-problems and then to split one of the new problems into two sub-sub-problems, the task would be coded as two dichotomous items. If a student splits the first task immediately into three sub-problems, the first item would be coded as 1 and the second as 0. Though he or she arrived at the correct solution of three sub-problems, he or she skipped a step and does not get full points. Though this is reflected correctly in the coding of the items, the answer to the second item is somewhat depended on the answer to the first. For this reason it might be advisable to code both items together with three levels instead of two, giving 2 points for a correct solution and 1 for the solution that skips one step.

The examination of items with a low discrimination reveals a number of multiple-choice items that appear to be too ambiguous to be answered reliably by the students. For example, the item "All available users are displayed in the chat room" was considered to be a useful requirement for the development of a chat system. The student responses suggest that especially those with a high ability estimate thought otherwise. A possible explanation might be that those students had privacy concerns regarding this feature. Therefore, the item should be changed to "All available users, whose status is set to 'visible', are displayed in the chat room", to compensate for that. Also, some of the ambiguity seems to stem from simple mistakes in the use of terminology. For example, one item in the same context as above read: "All students of the school can participate in the forum". Here "forum" should have been "chat" and therefore confused the students that worked on this task. This item should be removed from the evaluation of the current results and be corrected for future assessments.

5.3 Student Competences

Though the main goals of this research were to evaluate the competence model and measurement instrument and gather data to define proficiency levels (see section 6), the distribution of item difficulties can show tendencies in the competences present in upper German high school education. Without a refined competence level model the description of student competencies will be improper, but at least we can have a descriptive look at the student abilities in the different dimensions alongside the difficulty of typical items in that area.

To be able to interpret the results more easily, the item difficulties and person estimates were normalized using the PISA scale [2]. This way, the estimates have a mean of 500 and standard deviation of 100. Looking at the normalized item difficulties, the majority of items fall in the range of 400 to 600. In the range above 650 points there is a steep increase in difficulty with 13 items having up to 800 points on the scale (3 standard deviations above the mean). In the range under 400 points the difficulty also decreases fast with the easiest item having 228 points.

The five hardest items all belong to the competence dimension K4 *Dealing with system complexity*. All items belong to the same assignment, which tasks the students to deal with different components of a system. Most of the hardest items are from the two dimensions K4 and K3, with 17 of the 20 items with the highest difficulty belonging to them. A substantial part of these deal with some form of UML notation. It seems that this form of modeling systems was neglected in the assessed computer science courses. On the other hand, 18 out of the 20 easiest items are from the dimension K2 *System Comprehension*. It seems like the evaluation of the internal and external workings of a system are well taught. Especially the assessment of software qualities and dealing with data structures was an easy task for most of the students. The described trends continue for most of the items. A large part of the 50 hardest items are from dimensions K4 and K3, while on the other end of the scale dimension K2 dominates. This is also reflected in the means of the estimated difficulties for each dimension. These are for K1 to K4 respectively: 501.64, 418.3, 531.06, 561.66. The mean for Dimension K2 is almost one standard deviation below the global mean of 500 points, while K4 clearly seems to be the most difficult dimension.

6 Conclusions and Further Work

The evaluation of the competence model and the measuring instrument was a big step for our research. The next important goal is the definition of descriptive proficiency levels to form a competence level model. For this purpose, we will utilize an approach that already proved useful in similar studies like TIMSS [11, 25] and DESI [12]. The empirical data and the results from the MIRT analysis will be used to identify significant thresholds in the ability levels of the tested students, where a predefined percentage of students was not able to solve the next difficult items. The items adjacent to these anchor points can be used to describe the demands to a competence level. To get consistent and meaningful descriptions all items will also be rated by experts in several theoretically derived criterions to denote the demands of each item. By using regression analysis, the criteria with the biggest influence on the difficulty can be determined and used to calculate expected difficulties for certain combinations of criteria. This way we will end up with more consistent proficiency levels, that are independent from the actual items [4]. To be able to rate every item the test instrument was split into four parts, each rated by two experts. Two additional experts then discussed disagreements between the two raters and decided upon a final rating for every criterion. The expert ratings have been completed by now and work on the definition and description of proficiency levels is under way.

References

1. Adams, R.J.: Scaling PISA cognitive data. PISA Programme for International Student Assessment (PISA). PISA 2000 Technical Report, pp. 99–108. OECD Publishing, Paris (2002)

2. Adams, R.J., Wu, M.L. (eds.): PISA Programme for International Student Assessment (PISA) PISA 2000 Technical Report. OECD Publishing, Paris (2002)
3. Flanagan, J.C.: The critical incident technique. Psychological Bulletin 5(4), 327–358 (1954)
4. Hartig, J.: Skalierung und Definition von Kompetenzniveaus. In: Klieme, E., Beck, B. (eds.) Sprachliche Kompetenzen Konzepte und Messung DESI-Studie (Deutsch Englisch Schülerleistungen International), pp. 83–99. Beltz, Weinheim (2007)
5. Hartig, J., et al.: Methodische Grundlagen der Messung und Erklärung sprachlicher Kompetenzen. In: DESI-Konsortium (ed.) Unterricht und Kompetenzerwerb in Deutsch und Englisch. Ergebnisse der DESI-Studie, pp. 34–54. Beltz, Weinheim (2008)
6. Hartig, J., et al.: Modellierung von Kompetenzen mit mehrdimensionalen IRT-Modellen. Projekt MIRT. In: Klieme, E., Leutner, D., Kenke, M. (eds.) Kompetenzmodellierung: Zwischenbilanz des DFG-Schwerpunktprogramms und Perspektiven des Forschungsansatzes, pp. 189–198. Beltz, Weinheim (2010)
7. Hartig, J., et al.: Multidimensional IRT models for the assessment of competencies. Studies in Educational Evaluation 35(2), 57–63 (2009)
8. Hubwieser, P.: A smooth way towards object oriented programming in secondary schools. In: IFIP (ed) Informatics, Mathematics and ICT: A Golden Triangle: Proceedings of the Working Joint IFIP Conference IMICT 2007 (2007)
9. Hubwieser, P.: Analysis of Learning Objectives in Object Oriented Programming. In: Mittermeir, R.T., Sysło, M.M. (eds.) ISSEP 2008. LNCS, vol. 5090, pp. 142–150. Springer, Heidelberg (2008)
10. Hubwieser, P.: Computer Science Education in Secondary Schools – The Introduction of a New Compulsory Subject. Transactions on Computing Education (TOCE) 12(4), 41 (2012)
11. Kelly, D.L.: Interpreting the Third International Mathematics and Science Study (TIMSS) achievement scales using scale anchoring (Doctoral dissertation). Boston College Graduate School of Education (1999)
12. Klieme, E., Beck, B. (eds.): Sprachliche Kompetenzen. Konzepte und Messung. DESI-Studie (Deutsch Englisch Schülerleistungen International). Beltz, Weinheim (2007)
13. Lehner, L., Magenheim, J., Nelles, W., Rhode, T., Schaper, N., Schubert, S., Stechert, P.: Informatics Systems and Modelling – Case Studies of Expert Interviews. In: Reynolds, N., Turcsányi-Szabó, M. (eds.) KCKS 2010. IFIPAICT, vol. 324, pp. 222–233. Springer, Heidelberg (2010)
14. Linck, B., et al.: Competence model for informatics modelling and system comprehension. In: Proceedings of the 4th Global Engineering Education Conference, IEEE EDUCON 2013, Berlin, pp. 85–93 (2013)
15. Linck, B., et al.: Empirical refinement of a theoretically derived competence model for informatics modelling and system comprehension. In: Proceedings of IFIP-Conference "Addressing Educational Challenges: the Role of ICT (AECRICT)", Manchester (2012)
16. Magenheim, J., et al.: Competencies for informatics systems and modeling: Results of qualitative content analysis of expert interviews. In: Proceedings of the 1st Global Engineering Education Conference - Educon 2010, pp. 513–521. IEEE Computer Society, Madrid (2010)
17. Martin, M.O., Mullis, I.V.S.: Overview of TIMSS 2003. In: Martin, M.O., et al. (eds.) TIMSS 2003 Technical Report. Boston College, Chestnut Hill, MA, pp. 3–20 (2004)
18. Mayring, P.: Qualitative Content Analysis. Forum: Qualitative Social Research 1, 2 (2000)
19. Moosbrugger, H., Kelava, A. (eds.): Testtheorie und Fragebogenkonstruktion. Springer, Heidelberg (2008)

20. Osteen, P.: An Introduction to Using Multidimensional Item Response Theory. Journal of the Society for Social Work and Research 1(2), 66–82 (2010)
21. Rhode, T.: Entwicklung und Erprobung eines Instruments zur Messung informatischer Modellierungskompetenz im fachdidaktischen Kontext (Doctoral dissertation). University of Paderborn (2013)
22. Rost, J.: Lehrbuch Testtheorie–Testkonstruktion. Huber, Bern (2004)
23. The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society: Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM, New York (2013)
24. Tucker, A. (ed.): A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee. Association for Computing Machinery, New York (2003)
25. Watermann, R., Klieme, E.: Reporting Results of Large-Scale Assessment in Psychologically and Educationally Meaningful Terms. European Journal of Psychological Assessment 18(3), 190–203 (2002)
26. Weekley, J.A., Ployhart, R.E. (eds.): Situational judgment tests theory, measurement, and application. Lawrence Erlbaum, Mahwah (2006)
27. Wu, M.L., et al.: ACER ConQuest Version 2.0: Generalised item response modelling software. ACER Press, Melbourne (2007)

Pupils in the Virtual World and Education

Majherová Janka, Palásthy Hedviga, and Janigová Emília

Faculty of Education, Catholic University in Ružomberok, Ružomberok, Slovakia
{janka.majherova,hedviga.palasthy,emilia.janigova}@ku.sk

Abstract. In this article, we analyse the use of the virtual world of the Internet within secondary education in Slovakia. We investigate the current use of emerging Internet technologies, known as Web 2.0, in practice. We compare the opportunities for sharing educational materials, as well as the possibilities of accessing the Internet and its use by pupils. We recommended the digital version of Bloom's taxonomy for establishing educational objectives for teaching and learning with the support of ICT.

Keywords: virtual world, secondary education, digital competencies, digital taxonomy.

1 Introduction

Currently, teachers and pupils are increasingly using the virtual world of the Internet for obtaining and sharing educational materials. New possibilities are related to the use of Web 2.0 technologies. An introduction of the concept of Web 2.0 is associated with the emergence of blogs, social networks and websites, which have dynamic content and allow content sharing and interactivity among website visitors [12]. Web 2.0 allows users to add their own content to the web, which they have control over. Web 2.0 is built not only on the content, which is created by its owner, but also on the content, which is created by the community of web users. Examples of Web 2.0 are blogs, Wikipedia, social media such as Facebook, Google+, Twitter, YouTube and others.

The ability to understand information and use it in various formats from different sources, presented through ICT, is part of the digital literacy of the young generation. According to a study by the Institute for Public Affairs [17], the average value of the index of digital literacy in Slovakia has increased from 0.33 points (on a scale from 0 to 1), in 2005, to 0.43 points, in 2011. The study points out that the school has the most influence on the increase of digital literacy. In this research, young people achieved approximately 60 percent better results than the overall average population, while during the past ten years they have improved their abilities and skills in the majority of ICT that is commonly available.

Monitoring the development of digital literacy, according to Velšic [17], has revealed also two contradictory characteristics in the young generation. The first points to the fact that 18-26 year olds have improved their knowledge and skills in the majority of ordinary information technologies and services. Young people have improved in working and operating hardware, in working with multimedia, the Internet, in the administration and installation of software, in the use of various Internet services, communication technologies such as chat, IP telephony, discussion groups or social networks.

Young people cope with some ICT with more difficulties. The greatest shortcomings are related to “work on the network”. For example, the skill of transferring or copying data via a local area network (LAN) is below average. Young people also have similar problems in the case of searching for information and data in local computer networks or when working with databases.

The development of digital literacy is related to the immediate objectives and tasks of Informatics, as a compulsory school subject at primary and secondary schools. The key competences, which pupils have acquired while studying Informatics and are able to use them in other subjects, are for example [11].

- To present the acquired knowledge and apply it in everyday life.
- To creatively process information and use it while studying.
- To know how to analyse and use credible sources of information from the Internet.
- To locate, sort and process information.
- To apply knowledge to specific problems.
- To apply the acquired knowledge and skills through projects and cross-cutting themes in other subjects taught.

Teachers of Informatics should take into account, while teaching, the fact that today's pupils belong to a generation, which exalts new technologies and is experimenting with them. New gadgets employ intuitive learning by the method of “trial and error!”. They are not afraid to make mistakes, because they learn faster this way. They use devices according to their experience and they have no problem getting help online. They obtain information very quickly, favouring parallel processing and multi-tasking. They give precedence to working with graphics over text, as well as random access (hypertext), they work best when they are within a network. For them, we use the well-known term "Digital Natives" (Generation Y).

Many teachers then belong to the group that Prensky in [16] labelled as “Digital Immigrants” (Generation X). They must, while using ICT in teaching, think about how to teach their students in the language of the “Digital Natives”. In this paper, therefore, we attempt to describe the current state of Internet usage for the obtaining and sharing of information within education, in Slovakia. We recommended the digital version of Bloom's taxonomy for creating the educational objectives for teaching and learning with ICT.

2 The Development of Digital Competences and Web 2.0

In Slovak schools, there is a visible effort concerning the modernisation of education. Teachers are engaged in further education and implement a number of projects with the support of ICT. They implement school equipment with modern didactic techniques, in order that the teachers are able to implement teaching with ICT support for different subjects. The onset of interactive techniques can be seen, especially in science subjects, mathematics, foreign languages and, of course, at the lessons of Informatics and Informatics’ Education. Technique alone, however, is not enough. What is needed is also good quality digital content by [2] and [13].

Web 2.0 technology is currently the standard and at the same time is also necessary for almost all websites, networks and education portals. An example is the Interactive Tests portal (fig. 1, 2), which allows teachers to create and share learning materials and interactive tests for their subjects (`http://www.interaktivnetesty.sk/`) [20]. The portal offers teachers of Informatics, a variety of educational materials (thematic plans, methodologies of lessons, curriculum presentations, tests, etc.).

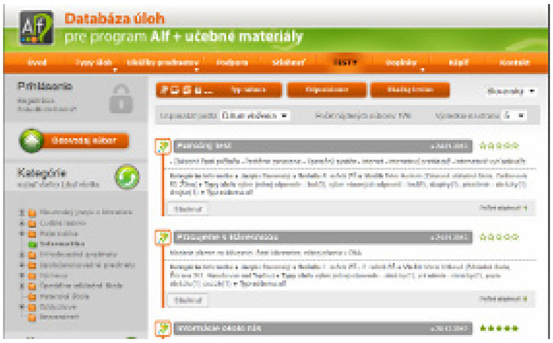
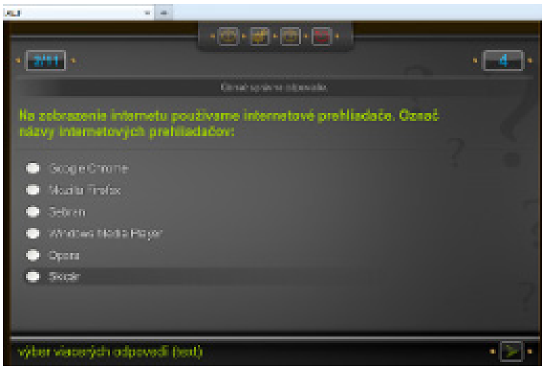


Fig. 1. Teaching materials from Informatics



Označ správne odpovede

Na zobrazenie internetu používame internetové prehliadače. Označ názvy internetových prehliadačov:

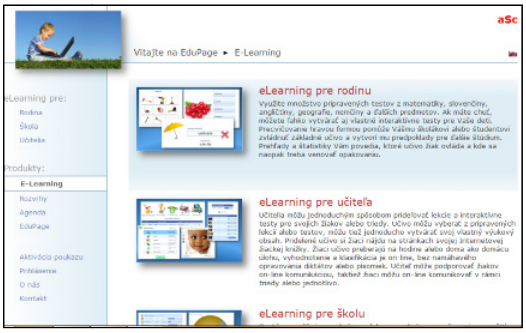
- Google Chrome
- Mozilla Firefox
- Sebran
- Windows Media Player
- Opera
- Skicár

výber viacerých odpovedí (text)

Fig. 2. ALF interactive test

Increasingly, teachers and students create their own websites. They use online tools such as the portal webnode.sk, to do it. This portal offers scope also for educational activities for schools. Students can upload and share whatever is beneficial for learning to their site - text, images, videos, documents or galleries. Teachers can create any amount of student accounts and thus have an overview of the activities of their students. They can also set which presentations will be open access and which will be accessible only to selected visitors (`http://edu.webnode.sk/`) [21].

As stated by Polčin [15], teachers can already find plenty of interactive materials, tests, worksheets and teaching aids at www.zborovna.sk [25], where new materials that are immediately useful in the classroom, are being updated daily. The portal contains very good search options, user contributions, transparent categorisation of content and a number of other options. Available here are various formats of documents, for example, .doc, .odt, .ppt, .pdf and links to other websites. Materials from the virtual library zborovna.sk are used by many teachers. Teachers use the Hot Potatoes program for creating interactive exercises, where they may form tests with a choice of the correct answer, as well as complimentary exercises and scrolling interactive exercises (fig. 3).



eLearning pre rodinu

Využite množstvo pripravených testov z matematiky, slovenčiny, angličtiny, geografie, nemčiny a ďalších predmetov. Ak máte chuť, môžete ľahko vytvárať aj vlastné interaktívne testy pre Vaše deti. Precvičovanie hravou formou pomôže Vášmu školákovi alebo študentovi zvládnuť základné učivo a vytvorí mu predpoklady pre ďalšie štúdium. Prehľady a štatistiky Vám povedia, ktoré učivo žiak ovláda a kde sa naopak treba venovať opakovaniu.

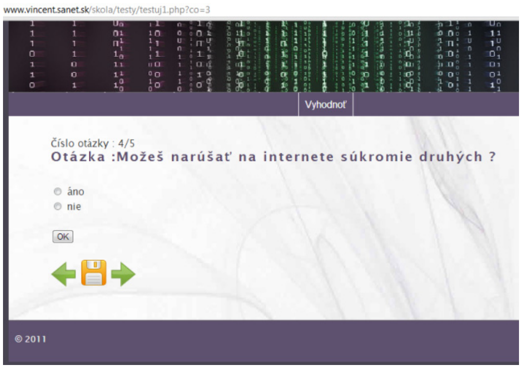
eLearning pre učiteľa

Učitelia môžu jednoduchým spôsobom prideľovať lekcie a interaktívne testy pre svojich žiakov alebo triedy. Učivo môžu vyberať z pripravených lekcií alebo testov, môžu tiež jednoducho vytvárať svoj vlastný výukový obsah. Pridelené učivo si žiaci nájdu na stránkach svojej Internetovej žiackej knižky. Žiaci učivo preberajú na hodine alebo doma ako domácu úlohu, vyhodnotenie a klasifikácia je on-line, bez namáhavého opravovania diktátov alebo písomiek. Učiteľ môže podporovať žiakov on-line komunikáciou, takisto žiaci môžu on-line komunikovať v rámci triedy alebo jednotlivo.

eLearning pre školu

Fig. 3. E-Learning at portal edupage.org

Teachers of informatics often use the LMS Moodle system, as well as their own system of electronic tests, created in the PHP language with the support of the jQuery library (fig. 4). The teacher prepares a test according to the content of the curriculum, and pupils may sufficiently prepare for the test, at home. After solving the test, the pupil is evaluated by a mark, which is written into a database of marks, which is accessible to the pupils themselves, as well as their parents. It is thus a sort of electronic pupil's book. In contrast to other commonly used electronic pupil's books, the teacher acquires the mark directly from the test and he does not have to write it into the system [24].



Číslo otázky : 4/5
Otázka :Možeš narúšať na internete súkromie druhých ?

- áno
- nie

Fig. 4. Interactive test from Informatics

Social networks are a separate chapter about using Web 2.0 in education. The phenomenon of the Internet, such as Facebook, which is used by primary school pupils (Generation Y), is for teachers (Generation X) often an unknown “territory”. Its opportunities in education are still only a little explored in our country.

2.1 Survey Results

We conducted survey using a questionnaire, in which we investigated the use of the Internet and social networking by pupils, in preparation for school. Our sample consisted of 66 pupils, 28 girls and 38 boys, from two primary schools, in Bratislava and Ružomberok. The average age of the respondents was 13 years. The questionnaire consisted of 11 questions, some of which have been concluded with the selection of one or more answers, some were open ended.

The results were compared with the results of research from all over Slovakia, conducted in 2010 [7]. The sample then consisted of 303 pupils, with an average age of 14 years. In the research results, almost all of the respondents from the sample (99.7%), at least occasionally connect to the Internet, at least at school. 84.5% of the respondents have an Internet connection at home. In each connection they most frequently spend on average 1-2 hours (31.8% of the respondents), or 2-3 hours (24.8% of the respondents). The girls did not differ from the boys in these characteristics. In some indicators, however, the authors found statistically significant differences between the boys and the girls. That has been shown also by our survey.

In our sample, 98% of the respondents already have an Internet connection at home. Most frequently, girls spend 1-2 hours on the Internet, boys spend a longer time on the Internet, 34% spend more than 3 hours on the Internet (see fig. 5 and 6).

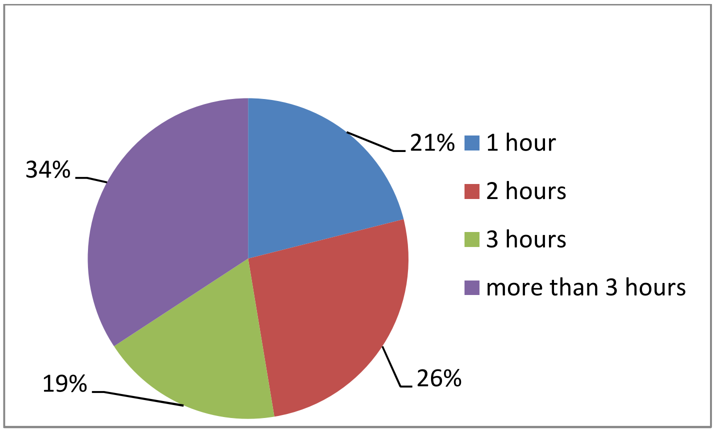


Fig. 5. Time spent on the Internet - boys

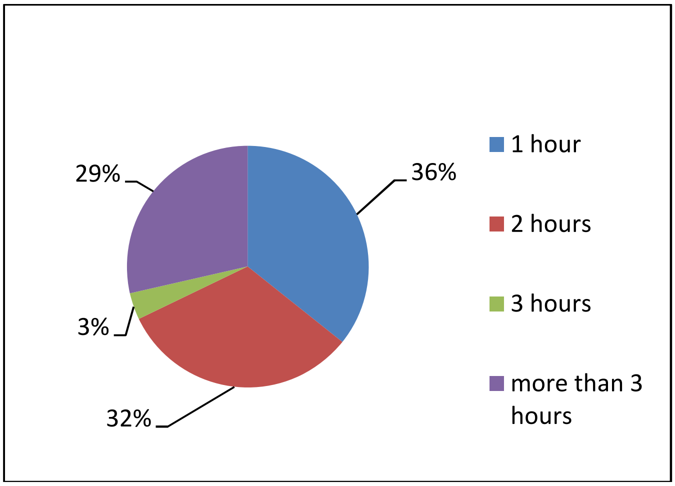


Fig. 6. Time spent on the Internet - girls

According to the results from the year 2010, the most popular Internet activities of adolescents were, in first place, chatting (77%); watching videos, YouTube (66%); downloading music, movies, software (63%), followed by social networks activity, Facebook (51%), and playing games (51%). When evaluating the selection of activities, the differences between girls and boys can be seen.

In the sample of pupils in our survey, several indicators have changed. Among the girls, the most popular activity is that of downloading movies and music (78%), and social network activities (68%). In boys, in first place, are watching videos on YouTube (89%), and downloading (84%). Compared to the research in 2010, there was a decrease in surfing the Internet, using e-mail, as well as in playing online games (in boys and girls). The use of social networking and the downloading of music and movies has increased. In our view, these changes are related to the development of the use of Web 2.0 technologies in Slovakia.

Table 1. Activities of pupils on the Internet (as a %)

Activity	2010		2014	
	girls	boys	girls	boys
Chat	87	66.0	25	71.05
Games	38.9	66.0	10.71	52.63
YouTube	65.4	67.4	46.43	89.47
Facebook	60.5	41.8	67.86	71.05
The downloading of music, movies and software	66.0	61.0	78.57	84.21
email	35.2	35.0	25	18.42
Surfing the Internet	34.6	43.3	7.14	28.95

Furthermore, in our survey we were interested in what kind of files the pupils are downloading most frequently from the Internet, and whether they are providing some materials themselves, e.g., from school. Girls and boys are most frequently downloading music and movies. Only 10% of boys and 13% of girls are not downloading anything from the Internet.

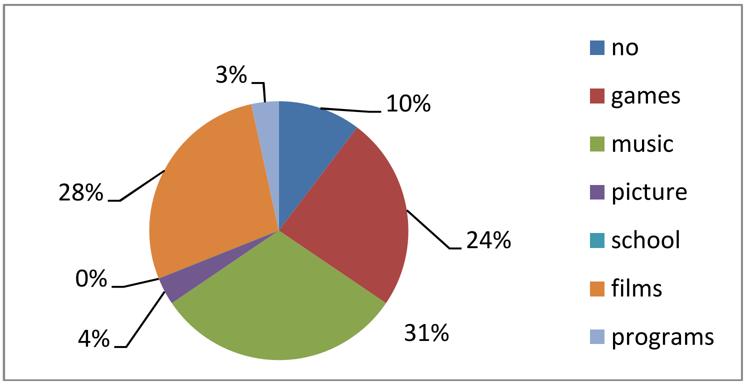


Fig. 7. Downloading files from the Internet - boys

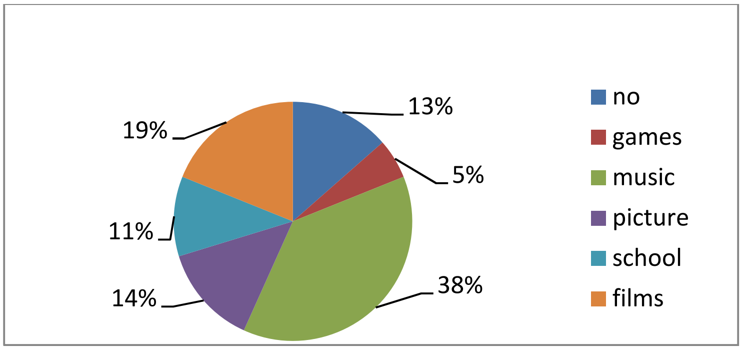


Fig. 8. Downloading files from the Internet - girls

Up to 66% of boys and 82% of girls stated that they do not provide any material for the Internet. Many of the pupils appear to not realise that published photos or statuses on the Internet can be also seen as disclosed information.

In the questionnaire, we also focused on the use of the Internet and social networking in school. 98% of pupils stated that they use the Internet and a computer in order to prepare for school. Teachers provide students with learning materials via the Internet and they use it practically for all the subjects.

In the last question, we investigated whether or not pupils use Facebook when preparing for school. The feedback showed that boys use Facebook, especially for communication when doing their homework (93%), only a small percentage (7%) do not have, or do not use, Facebook. In contrast, 65% of girls stated that they do not use Facebook when preparing for school, the remaining 35% use it for communicating and when doing their homework.

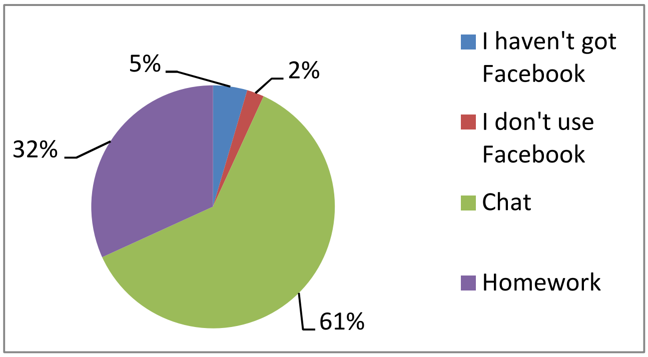


Fig. 9. The use of Facebook while preparing for school - boys

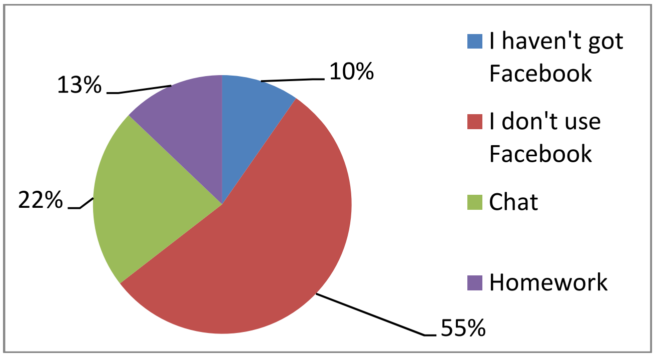


Fig. 10. The use of Facebook while preparing for school - girl

The pupils know and use mostly the portal Google to search for information, many also know Wikipedia. We were interested whether pupils consider information received on the Internet to be true and correct. More than half responded: “I do not know”. Approximately one third of pupils do not believe the information on the Internet.

To the question of which school subjects they need the Internet for, the primary school pupils reported mainly the science subjects (the preparation of a project). The sharing of teaching materials in electronic form (the presentation of the curriculum, laboratory practice protocols), or electronic test solutions, becomes a matter of course.

3 Educational Goals and Digital Taxonomy

When setting goals for education in the subject of informatics, in the context of the development of the digital competences of pupils, we can be motivated by new approaches, according to a revised Bloom's taxonomy by Anderson [1]. A digital application of Bloom's taxonomy, for the conditions of learning in today's digital world, was created by Churches [3, 4]. It lies in the linking of different levels of BT to the specific activities of pupils, when using ICT.

The category of Remembering, in the revised Bloom's taxonomy, represents the usage of memory for the designation and acquiring of knowledge, facts and definitions. We focus on obtaining information, when using ICT at this basic level. The ability to remember everything needed, we can formulate by computer-oriented words, to search ("to google"), to select, to cut, to save, and so on. It is also important that we do not lose the discovered information, which in practice means using bookmarks (favourites). An example of activity for the application of this level, when working with ICT, can be, in addition to work with files and folders, also the knowledge of basic search on the Internet (e.g. through the Google portal), and the ability of a pupil to solve an online test.

Understanding, represents constructing the importance of different types of knowledge. Churches explains this level of digital competence as the ability to process the information identified. The first sign of understanding may be that we know how to structure the information in an appropriate manner. An appropriate activity with pupils is, for example, the creation of a mind map.

The level of Applying connects and refers to situations where the student acquires knowledge through products like models, presentations and interviews. When working with ICT, this level is represented by computer literacy, thus the ability to apply computer skills through the most appropriate way. That means to apply the right tool in the form of hardware and software. A typical process is the editing of various types of information (graphics, text, numbers, etc.), and their publication within an online environment.

The level of analysing lies in the fact that, pupils were able to show some signs of functional literacy. This means that they should understand the meaning of information, with which they work, and that they will be able to assess the credibility of the obtained information. We teach them to be able to recognise the structure of information content, as well as to identify the origin of the individual components and evaluate the seriousness of the source.

The ability to assess is directly intertwined with analysis. Typical examples are school blogs, electronic journals and the presentation outputs of project teaching. Pupils then mutually react to their contributions and they learn to not only assess the work of others, but even their own work.

Finally, this is followed by the category of Creating. Digital technologies now play a major role in implementing most of the required tasks in teaching. Many teaching outputs are produced by pupils, on the computer. It can be text, graphics, audio or video. It is important that the result will be a unique product and the pupil will be able to pass on something to the other classmates. A typical procedure applicable for this creation, is the creation of presentations or the telling of digital stories. The creation and sharing of the work done by pupils online, may be achieved, for example, through Google Docs.

In the following table, we present examples of pupil activities with their extension for the use of digital technologies according to Churches [4]. The implementation of individual activities varies, according to the particular subject. The selection of appropriate educational programs, the creation of presentations and other electronic materials for teachers, as well as technical support using an interactive white board, is also connected to it, as stated by Gunčaga [9].

Table 2. Activities of pupils concerning digital taxonomy

Level	Activities	Digital addition
Creating	designing, constructing, planning, producing, inventing, devising, making	programming, filming, animating, Blogging, Video blogging, mixing, remixing, wiki-ing, publishing, video casting, podcasting, directing/producing, creating or building mash ups
Evaluating	Checking, hypothesising, critiquing, Experimenting, judging, testing, Detecting, Monitoring	(Blog/vlog) commenting, reviewing, posting, moderating, collaborating, networking, reflecting, testing, validating.
Analysing	Comparing, organising, deconstructing, Attributing, outlining, finding, structuring, integrating	Mashing, linking, reverse-engineering, cracking, mind-mapping
Applying	Implementing, carrying out, using, executing	running, loading, playing, operating, hacking, uploading, sharing, editing
Understanding	Interpreting, Summarising, inferring, paraphrasing, classifying, comparing, explaining, exemplifying	Advanced searching, boolean searching, blog journaling, tagging, categorising and tagging, commenting, annotating
Remembering	Recognising, listing, describing, identifying, retrieving, naming, locating, finding	Bullet pointing, highlighting, bookmarking, social networking, Social bookmarking, favouriting/local bookmarking, Searching, googling,

To achieve the stated educational objectives, we can use the curriculum of several thematic areas of the subject Informatics, for example, Information Around Us (the processing of textual and graphical information, spreadsheets, the presentation of information); Communication Through ICT (searching for information on the Internet, sharing),and Procedures and Problem Solving (the creation of algorithms and programming) [12]. For teachers of Informatics, it is also important to know the process of creating and assessing learning objectives, in accordance with new knowledge, in order for teaching not to be formal, but to prepare pupils for practice [5] and [10].

4 Conclusion

The virtual world of the Internet has many advantages, which can be used in education, such as an unlimited dissemination of information, speed, access to services at anytime and anywhere. Using the virtual world of the Internet also brings certain risks. Teachers (not only) of Informatics, should know also this side of the movement of pupils on the Internet. Experts are exploring the possibility of addiction of children and young people from the virtual world. Gregusová [7], in her research, indicates that not only addiction from computer and video games can be observed in children, but also virtual relationships from surfing on the Internet and other addictions. Addictions from the Internet represent an increased risk, especially for children of a school-age. All Internet addiction has one main common feature, and by that they are quickly and easily obtained pleasant feelings. A major risk of the virtual world is also making contact with inappropriate people or cyberbullying.

How can teachers of Informatics help pupils with orientation in the maze of the virtual world? Extending the learning objectives about the digital dimension of the proper use of ICT can increase the effectiveness of teaching. In cooperation with the development of other types of pupil literacy - reading, mathematical and natural science, thus helps pupils with their inclusion in the information society. In teaching the subject of Informatics, we have scope for the development of the following skills of the pupils at a higher level:

- to locate, collect and process information and use it in a critical and systematic way, to visit websites, or create them on their own,
- to assess their importance and distinguish between real and virtual information
- to use tools for the creation, presentation and understanding of complex information,
- to acquire, search and use Internet services.

The taxonomy of the digital competences of pupils may be an incentive for the teachers of Informatics at primary and secondary schools, while updating the contents of the subject of Informatics. Pupils from lower grades are already coming with greater skills within the work with ICT, as from a few years ago. The linking of Informatics' competencies with other key competencies will become an important aspect of teaching Informatics at all stages of education.

It is necessary to also prepare the future teachers of Informatics, on these aspects. They often have conceptions about education from the time of their studies at primary and secondary school. The benefit is when they experience pedagogical practices in a modern school of the 21st Century and they will also use their own experiences with the virtual world in pedagogical practice.

Acknowledgements. This work is supported by the Scientific Research Fund KEGA 010 TTU-4/2012 Development and validation of virtual excursions in high school and VEGA 1/0087/13 Internet dependence and personality of type D.

References

1. Anderson, L., et al.: A Taxonomy for the Learning, Teaching and Assessing of Educational Objectives. Longman, New York (2001)
2. Černák, I.: The importance of the quality of education of current and future teachers: a virtualisation learning environment and the new role of the teacher. In: Učiteľ, žiak a motivácia vo vzdelávaní včera, dnes a zajtra (The Teacher, Pupil and Motivation in Education Yesterday, Today and Tomorrow), pp. 173–189. Raabe, Bratislava (2012) (in Slovak)

3. Churches, A.: Bloom's digital taxonomy, http://edulibpretoria.files.wordpress.com/2009/05/blooms-digital-taxonomy.pdf
4. Churches, A.: Edorigami, http://edorigami.wikispaces.com/Bloom%27s+Digital+Taxonomy
5. Gazdíková, V.: Počítačové zručnosti žiakov základných škôl, potrebné pre e-vzdelávanie (The computer skills of primary school pupils, necessary for e-learning). In: Acta Facultatis Paedagogicae Universitatis Tyrnaviensis, ser. C, Trnava, vol. 11 (2007)
6. Gregussová, M., Drobný, M.: Deti v sieti (Children in the network). eSlovensko 2013 (2013)
7. Gregussová, M., Tomková, J., Balážová, M.: Teens in cyberspace. Iuventa, Bratislava (2011)
8. Gülseçen, S., Çelik, S., Özdemir, S., Uğraş, T., Özcan, M.: Education in smart cities. In: New Challenges in Education, pp. 126–129. Verbum, Ružomberok (2013)
9. Gunčaga, J.: GeoGebra in Mathematical Educational Motivation. Annals: Computer Science Series, Tome 9, Fasc. 1, 75–84 (2011) ISSN 1583-7165
10. Jablonský, T., Kolibová, D., Matúšová, S.: European Values and Cultural Heritage - a New Challenge for Primary and Secondary School Education, p. 201. University of Debrecen, Debrecen (2012)
11. Kalaš, I.: Learning activities for students in the 21st Century. In: Kalaš, I. (ed.) DIDINFO 2012, pp. 35–46. UMB, Banská Bystrica (2012)
12. Majherová, J.: Revised Bloom taxonomy and competencies for using ICT. In: Gunčaga, J., Jablonský, T., Nižnanský, B. (eds.) Didactics – Interdisciplinary Dialogue 2010, Verbum. Ružomberok (2011)
13. Mandić, M., Ivanović, M.: Experience of applying a Wiki in university courses. In: Kalaš, I. (ed.) DIDINFO 2012, pp. 141–146. UMB, Banská Bystrica (2012)
14. Polčin, D., Majherová, J.: Rozvoj digitálnej gramotnosti žiakov v predmetoch fyzika a informatika (The development of the digital literacy of students in Physics and Informatics). In: Gunčaga, J., Nižnanský, B. (eds.) Didactics – Interdisciplinary Dialogue, Verbum. Ružomberok (2011)
15. Polčin, D.: Objectives, tasks and the importance of the "Support Programme Digitalisation of Schools" as an initiative of the centre of modern educational technologies EDULAB. In: Gazdíková, V. (ed.) Didactics – Interdisciplinary Dialogue, pp. 177–183. Verbum, Ružomberok (2012)
16. Prensky, M.: Digital Natives, Digital Immigrants. In: On the Horizon, MCB University Press, 9(5) (October 2001), http://www.marcprensky.com/writing/prensky%20-%20Digital%20Natives,%20Digital%20Immigrants%20-%20Part1.pdf
17. Velšic, M.: Digital literacy through the optics of the young generation. Research report. Microsoft Slovakia, s.r.o. & the Institute for Public Affairs, Bratislava (2013)
18. Velšic, M.: Digital literacy and the labour market. Research report. Institute for Public Affairs, Bratislava (2010)
19. http://en.wikipedia.org/wiki/Web_2.0
20. http://edu.webnode.sk
21. http://interaktivnetesty.sk
22. http://www.datakabinet.sk
23. http://edupage.org
24. http://vincent.edupage.org
25. http://www.zborovna.sk

Introducing Students to Recursion: A Multi-facet and Multi-tool Approach

Maciej M. Syslo[1,2] and Anna Beata Kwiatkowska[2]

[1] Institute of Computer Science, University of Wroclaw
F. Joliot-Curie str. 15, 50-383 Wroclaw, Poland
syslo@ii.uni.wroc.pl
[2] Faculty of Mathematics and Informatics, Nicolaus Copernicus University
Chopin str. 12/18, 87-100 Torun, Poland
aba@mat.uni.torun.pl

Abstract. In this paper we discuss a number of results and advices coming from our observations and didactical experience gathered when teaching about recursion in different contexts and on various education level (K-12 and tertiary). Knowing the difficulty in introducing, explaining and using recursion, we differentiate our approach, tools, and methods. Recursion can be introduced as a 'real-life topic' – see Section 2, and then software for visualization of recursive computations (Section 3) can be very helpful to overcome some difficulties by novices. Section 4 is on developing recursive thinking – we use two popular topics – generating Fibonacci number and printing digits of a number – to explain how to introduce students to different aspects of recursion. Section 5 is addressed to complexity of recursive computations – we discuss how to use recursion in a most effective way.

We do not teach recursion as a separate topic or subject, it is rather a method and a tool, the way of thinking, used in various situations. It is redundant in some cases (Section 5.1), can be used as an approach alternative to iteration (Sections 2 and 4.2), but our main focus is on its properties as a concept which has a computational power in designing solutions of problems and running such solutions on a computer.

Keywords: recursion, iteration, induction, algorithm visualization, Fibonacci numbers, Horner's rule, printing digits of a number, fast exponentiation, divide and conquer.

1 Introduction

Recursion is one of the main concepts in computer science and still remains a challenge for both teachers and students – recursion has been found to be one of the most difficult topics in discrete mathematics and in programming to master for students. We present here various facets of recursion together with various tools and examples which help teachers to explain and students to understand recursion as a discrete mathematics concept and as a programming tool.

Most of the textbooks focus on classical examples of recursion such as factorial, the Fibonacci numbers, the Towers of Hanoi, Euclid's algorithm, permutations, a binary search, and quicksort in a very traditional way which do not really illustrate the full potential of recursion and recursive thinking. There have been a number of papers focused on some real-life examples of recursion such as parking cars [30], task delegation [5], Cargo-Bot game [26] to contextualize learning of recursive operations and recursive thinking. Recursive thinking is one of the competences within computational thinking, which comprises competences expected to be mastered in some range by all citizens of the knowledge society, see [16], [3].

Recursion is a tool for problem-solving by decomposition of a problem into subproblems of the same kind – one has to specify two methods: decomposition of a problem into subproblems and composition of the solutions of the subproblems into the solution of the problem. Such approach is used in traversing a tree, in some sorting algorithms, and generally in a divide and conquer method. In some other cases, recursion is used to reduce a problem to the same problem but with reduced (smaller) parameters, e.g. in factorial, binary search, Euclid's algorithm.

Recursion and recursive algorithms can be categorized according to the type and the number of recursive calls. We can distinguish the following recursions: linear, multiple, nested, and mutual. In a linear recursion (e.g. factorial, summation of a series) at most one recursive call is allowed at a level of recursion, and a multiple recursion may contain more than one recursive call at a level (e.g. the Fibonacci numbers, the Tower of Hanoi, a binary tree traversals). A nested recursion appears in the Ackerman function and a mutual recursion in its simplest form is a system of two recursions in which recursive calls in one recursion are to the other recursion, see [19] for some interesting mutual recursions and their applications. A multiple recurrence is called also exponential since its computational complexity is non-polynomial (see Section 5, where we consider a system of two recurrence relations for the Fibonacci numbers which are mutual as well as multiple.

There are several methods and tools for explaining recursion to students; see [13]:

- induction – a recursive function is defined in terms of itself (recursive calls) and base case(s), for instance as a computer implementation of a recursive algorithm or a function;
- runtime stack – implementation of recursion in a high level programming language involves a stack for frames which contain local variables, parameters, return address and other information; when a procedure is invoked a corresponding frame is pushed onto the stack and it is popped off the stack when a called procedure returns control to its calling procedure;
- the trace – when a procedure is called, a procedure name with its input parameters is listed;
- the recursion tree – the tree in which nodes correspond to procedure calls: the parent of a node is the procedure which called the node and the children of a node are the procedures which that node calls; see [14] for an extended version of the recursion tree called a recursion graph (RGraph).

Quite often more than one of these methods is used when introducing and explaining a particular recursion or a recursive function, especially in visualization systems, see Section 3.

There exists a close relation between two notions: (mathematical) induction and (computational) recursion. The authors of [17] have summarized their investigations as follows: ‘recursion is an executable version of induction’. To the question ‘which of the two topics is simpler? which should come first in the learning sequence?’ they answer: ‘recursion is initially more accessible than induction … it is not recursion *per se* that is easier than induction. Rather, it is *recursive activities with the computer* that are easier than *inductive proofs with pencil and paper*.’ We refer also to [23] for discussion how informatics education could contribute to mathematics education.

In this paper we focus on recursion as a mental tool for computing – to perform some computations and to understand computational processes which stay behind recursion and recursive objects. Moreover we illustrate that recursion naturally appears in formulation of some problems and then as a method for their solutions. The main objective in this paper is to illustrate how we introduce our students to various aspects of recursion and explain the most important properties and facets of this concept and its application in computing. With regard to recursion as a tool for computations, we also focus on complexity of using recursive procedures, see Section 5. Recursion is a very powerful concept in problem solving and designing algorithms, but from the computational point of view, it is always profitable to implement a recursive solution as an iterative process, since recursion may be considered as another way of looking at iteration.

In Poland, some students in K-12 may first encounter recursion in primary or middle schools when using Logo or Pascal, mainly for drawing recursive pictures (e.g. fractals), and then in high schools when they choose informatics as an elective subject. Most of the examples explained in this paper are included in the textbook [11] for students in high schools, which meets the curriculum and evaluation standards for school informatics (at the high school level) approved by the Ministry of Education. One can also find these topics in Level III (*Computer Science as Analysis and Design*) in *A Model Curriculum for K-12 Computer Science*, published by ACM in 2003 [1] and also at Stage IV (*ICT Specialization*) in *A Curriculum Structure for Secondary Schools*, published by IFIP/UNESCO in 2002 [15].

The example presented here were also used in the outreach project Informatics + [25], addressed to more than 1000 high schools in 3 regions – more than 17 000 students participated in this projects, at least half of them took some courses on introductory algorithmics and programming. At a tertiary level, students of informatics and informatics related studies learn recursion as a mental and algorithmic concept and tool in a course on discrete mathematics and as an algorithmic and programming tool in programming courses. Recursion appears also in a course on algorithms and data structures. Students encounter recursion as a problem-solving approach, technique, heuristic, and tools (e.g. divide and conquer), and also as recursive structures (e.g. binary trees) and schemes (e.g. search and traversal schemes, and sorting schemes).

2 Computer Science Unplugged

In the beginning, when introducing recursion to students, we usually follow an idea of Ershov [7] and describe some activities, which consist of a number of steps, in a ‘recursive way’ of performing these steps – the problem reduces to a simpler (smaller) one. Other such problems are the Russian dolls and Open a present [2]. Ershov proposed to eat porridge:

```
procedure eat_porridge;
  if the plate is empty then Stop
  else
    eat a spoonful of porridge;
    eat_porridge
```

This example shows that in the successive calls of the eating procedure, the ‘size’ of the problem becomes really smaller, and smaller, reduced by a spoonful of porridge at each turn (step). The recursive calls stop when the plate is empty.

Since eating in a classroom is usually not allowed, we can use dancing for the same purpose, moreover this can be done in the classroom by all students:

```
procedure dance;
  if the music is not played then Stop
  else
    make a step;
    dance
```

The stopping condition in this example also corresponds to lack of something – music. Note here however that this stopping condition is external in a sense that the music does not depend on the actions taken in the procedure – making steps to dance.

Both examples show that the stopping conditions as the base case in the procedural approach to recursion need not to correspond to the end of decomposing problems into smaller similar problems, as assumed in [12]. In fact, instead of decomposition we have here a simple reduction of problems into smaller ones by removing one element (a spoonful of porridge or a step).

If recursion is introduce when students are familiar with some basic program constructions for programming an iteration, we demonstrate that there is a simple way to transform a recursive procedure to iteration, as shown in Fig. 1.

Eating porridge:
`while the plate is not empty do`
`  eat a spoonful of porridge`
	Dance:
	`while music is played do`
	`  make a step`

Fig. 1. Eating porridge and dancing as iterations

A similar correspondence between conditional iteration and recursion is shown in Fig. 2 for approximating the square root of `a` with accuracy `eps` for the initial value `x`.

`while abs(x*x - a) > eps do`
`   x:=(x+a/x)/2;`
	`procedure sr_r;`
	`if abs(x*x - a) > eps then`
	`   x:=(x+a/x)/2;`
	`   sr_r`

Fig. 2. Iterative and recursive ways of calculating square root of a

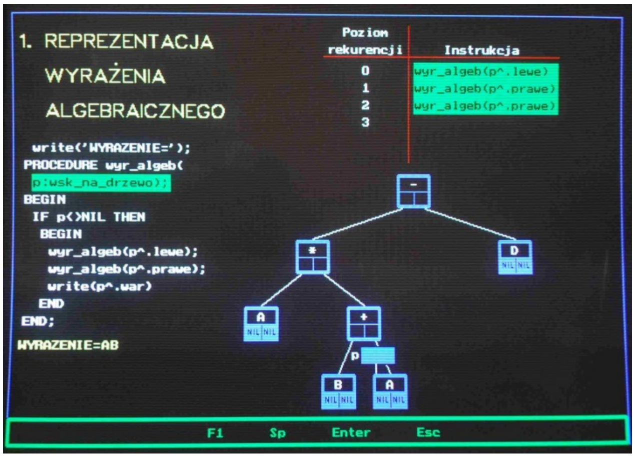
3 Visualization of Recursion

A number of visual representations have been developed for recursive procedures – they provide a visual aid to illustrate the concept of recursion.

In 1993, a software package – EI (Elements of Informatics) – was designed for IBM PC by a team supervised by the first author to support teachers of informatics and of related subjects, see [6]. The software, comprehensively tested and accompanied by documentation and educational materials (more than 1 000 pages), was then produced in 1 300 copies and delivered to high schools in which informatics was taught as an elective subject. The package is still used in some schools.

Package EI consists of 10 educational systems: We list here only those systems which are of some use in teaching and learning recursion:

- **SB** – a system for constructing and executing flow-charts of algorithms.
- **TP-TOOL** – a set of tools for supporting learning of programming in Turbo Pascal.
- **DISC-MATH** – a system consisting of 6 programs for supporting learning algorithms and data structures: operations on list and tree data structures, sorting algorithms, operations and algorithms on graphs, backtracking algorithms, a model of a universal computer (RAM).
- **ASD** – a system for demonstrating and analyzing algorithms and data structures.
- **MET-NUM** – a system for supporting learning numerical methods (computer realization of mathematical calculations) and for performing some numerical calculations.



1. REPREZENTACJA WYRAŻENIA ALGEBRAICZNEGO

```
 write('WYRAZENIE=');
PROCEDURE wyr_algeb(
 p:wsk_na_drzewo);
BEGIN
 IF p<>NIL THEN
  BEGIN
   wyr_algeb(p^.lewe);
   wyr_algeb(p^.prawe);
   write(p^.war)
  END
END;
WYRAZENIE=AB
```

Poziom rekurencji	Instrukcja
0	wyr_algeb(p^.lewe)
1	wyr_algeb(p^.prawe)
2	wyr_algeb(p^.prawe)
3	

Fig. 3. A screen snapshot of Package EI – demonstration of a recursive procedure on a tree data structure

Package EI, supported by very rich written educational materials, could be used in a number of ways: by a teacher, during a lecture or exercises – to illustrate concepts and their properties, and to demonstrate methods and algorithms; by students – during a teacher's demonstration, when working in groups and individually. Depending on the preparation and expertise level of students, package EI could be gradually used:

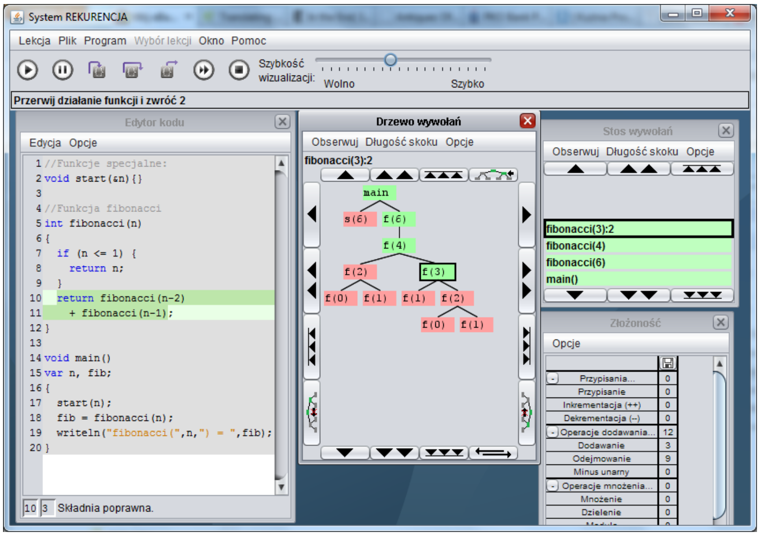
- as a toll for demonstrating concepts, their properties, methods and algorithms, e.g. recursive sorting algorithms, searching in binary trees;
- as a demonstration tool in which the users can make experiments with different values of parameters of their own choice, e.g. how sorting methods work on sorted sequences;
- to plan and design exercises supported by the package, e.g. to compare efficiency of various sorting algorithms;
- to write own programs utilizing some software modules available in the package, e.g. write a complete program for transforming an expression to the RPN format.

In our vision, when we were designing and implementing Package EI, similarly to Seymour Papert: *the student programs the computer instead of the computer is being used to program the student.*

Some of the systems in the package EI have been redesigned and implemented for the Windows environment, and now are available as open educational software, [6].

The screen snapshot in Fig. 3 shows an animation in which for a given (randomly generated) expression tree, the expression in the Reverse Polish Notation (RPN) format is produced. The demonstration consists of: the trace (in upper right corner) which is a list of the procedure name and input parameters, the full recursion tree, and the recursive procedure (on the left) which recursively traverse the expression tree in the post-order fashion and generates the expression in the RPN format.

Another system for visualization of recursion has been proposed in [4], see Fig. 4. For each problem and its solution in the form of a recursive program in C++, presented are: the recurrence tree, the trace, the runtime stack and the complexity table for a run of the program for parameters chosen by the user, who may watch its own speed performance of the program with highlighted instructions and all data updated at each step. Complete description of this system will appear elsewhere.


System REKURENCJA
Lekcja Plik Program Wybór lekcji Okno Pomoc
Szybkość wizualizacji: Wolno Szybko
Edytor kodu
Edycja Opcje
Drzewo wywołań
Obserwuj Długość skoku Opcje
main
s(6)
f(6)
f(4)
f(2)
f(3)
f(0)
f(1)
f(1)
f(2)
f(0)
f(1)
Stos wywołań
Obserwuj Długość skoku Opcje
Złożoność
Opcje

Przerwij działanie funkcji i zwróć 2

```
//Funkcje specjalne:
void start(&n){}

//Funkcja fibonacci
int fibonacci(n)
{
  if (n <= 1) {
    return n;
  }
  return fibonacci(n-2)
    + fibonacci(n-1);
}

void main()
var n, fib;
{
  start(n);
  fib = fibonacci(n);
  writeln("fibonacci(",n,") = ",fib);
}
```

10 3 Składnia poprawna.

fibonacci(3):2

fibonacci(3):2
fibonacci(4)
fibonacci(6)
main()

- Przypisania...	0
Przypisanie	0
Inkrementacja (++)	0
Dekrementacja (--)	0
- Operacje dodawania...	12
Dodawanie	3
Odejmowanie	9
Minus unarny	0
- Operacje mnożenia...	0
Mnożenie	0
Dzielenie	0

Fig. 4. A screen snapshot of another system for visualization of recursion – computing Fib(5)

4 Developing Recursive Thinking

There are a number of simple tasks we use to teach recursive thinking which result in better understanding of recursion as a process of leading to recursive solutions in the form of recursive algorithms and/or recurrence relations in the case when the goal is to calculate some quantities.

4.1 Fibonacci Numbers

Recursion is most often introduced using the Fibonacci numbers, which originally came out as the answer to the question about the population of rabbits in a rather non realistic birth model. We used to introduce these numbers using another ‘story’ (which can be found in many textbooks) together with some real-world situations, in which these numbers occur.

Here is the ‘story’ [23]: A professor S. has his office on the second floor and the staircase from the first floor to the second floor consists of 12 (in general n) steps. He is a very chaotic person and he takes one or two steps at a time. In how many ways can he reach his office?

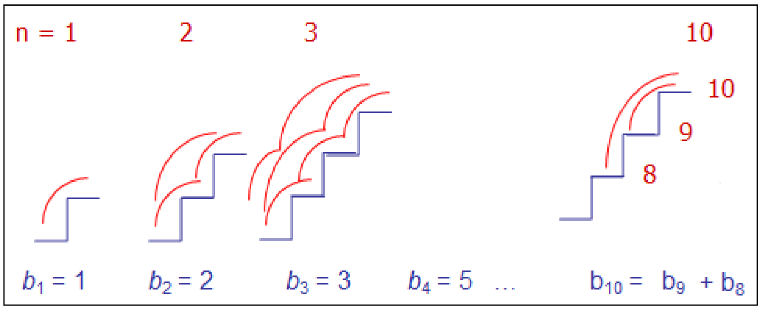


Fig. 5. Another context for introducing … the Fibonacci numbers

Students usually start to make pictures and calculations for staircases with 1, 2, 3, 4 … steps and obtain correct answers: 1, 2, 3, 5,… After a while, we ask them to guess the next number. If they did not hear about Fibonacci numbers before, they have no idea how these numbers are changing. Then we suggest them to look at the last step of the staircase and try to answer, how Professor S. can reach this step – in this way we move our students to think recursively. They suddenly see the rule – there are two ways to reach a given step: taking either one or two steps at the end, so the following relation is satisfied (see Fig. 5):

$$b_i = b_{i-1} + b_{i-2}, \qquad \text{for } i > 2$$

with $b_1 = 1$, and $b_2 = 2$, where b_i denotes the number of ways Professor S. can reach step i. We therefore obtain $b_i = F_{i+1}$, where F_i is the i-the Fibonacci number.

In a similar way we can develop with students an answer to the following question: how many subsets does the set $\{1, 2, 3, \ldots, n\}$ have that contain no two consecutive numbers? In this case, a subset of $\{1, 2, 3, \ldots, n\}$ with no two consecutive numbers can be obtained from a smaller subset ether by adding number n to a subset of $\{1, 2, 3, \ldots, n-2\}$ or taking a subset of $\{1, 2, 3, \ldots, n-1\}$.This again ends up with the Fibonacci recurrence relation.

Fibonacci numbers 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, … look quite strange to students – except the above relation, there is no regularity in this sequence. We usually avoid to present Binet's formula for b_i, since it contains square roots and hence does not help in understanding these integer numbers. However, as a means of contextualization, we are very enthusiastic about real-world objects in which one can find Fibonacci numbers, such as: pine cones, sunflowers, leaf arrangements in some plants, vegetables and fruits, etc. Interesting is also the relation between Fibonacci numbers and the golden ratio – we refer students to a chapter in [22] on application of Fibonacci numbers.

The solution method illustrated above, resulting in a recursive procedure or in a recurrence relation, Ginat [10] calls 'going backwards' and treats as the core of recursive thinking. He illustrates this approach with another nice problem: for two natural numbers X and Y, such that X < Y, find the smallest number of operations +1 (add 1) and x2 (multiply by 2) that are required to obtain Y from X.

The formula for the Fibonacci numbers can be used to explain that a difference between iteration and recursion lies in the way a formula is interpreted, as shown in Fig. 6. In general, iteration is a repetition of the same procedure and recursion is a reduction of a solution of a problem to simpler versions of the same problem. Recursion may be interpreted in some cases as implementation of iteration.

Iteration	**Recursion**
$b_1 = 1, b_2 = 2,$ $b_i = b_{i-1} + b_{i-2},$ for $i = 3, 4, 5, \ldots$	$b_1 = 1, b_2 = 2,$ $b_i = b_{i-1} + b_{i-2},$ for $i > 2$

Fig. 6. Iteration versus recursion

Iteration means that numbers b_i are calculated in the order: b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8,… On the other hand, recursion is interpreted as follows: if we want to calculate b_i, for a given i, then if $i = 1$ then $b_1 = 1$, if $i = 2$, then $b_2 = 2$, and if $i > 2$, then $b_i = b_{i-1} + b_{i-2}$, therefore to calculate b_i we have to find b_{i-1} and b_{i-2} in a similar way using the recursive formula.

4.2 Printing a Number

A quite popular programming task is to print an integer, digit by digit, starting with the most important digit. This task sounds strange to some students since they can easily print any integer `n` by using a simple instruction `write(n)`. Nevertheless it is worth to convince students that the best of this task is to come at the end.

There are two questions to be answered when attacking this task: (1) how to extract a digit from an integer stored in a computer memory and (2) how to extract a digit at a given position. Regarding the first question, students who are not familiar with the computer representation of numbers think that integers are stored digit by digit in a computer memory. Therefore it is important to make yet another assumption that we have access to the value of a number but not to its digits, those should be extracted from the value of a number.

To answer both questions it is important that students know how to calculate the less important digit of a number – it is equal to the reminder when dividing a number by 10 and it can be calculated using operation `mod`. Regarding the second question – it could be answered with the help of the answer to the first question: if we want to find the *k*-th digit of a number we have to cut off $k - 1$ right most digits. This can be done recursively digit by digit, starting with the less important position. To this end, another operation is useful – `div`, which produces integer value of division and can be used to cut off a right (that is, a less important) part of a number. A complete solution is given in Fig. 7 (the left column).

```
procedure Digits(n:int);
  if n < 10 then write(n)
  else begin
    Digits(n div 10);
    write(n mod 10)
  end
```

```
procedure Digits(n,p:int);
  if n < p then write(n)
  else begin
    Digits(n div p,p);
    write(n mod p)
  end
```

Fig. 7. Extracting digits from numbers represented in the decimal system (on the left) and represented in the system with respect to the base *p*

There are a number of other interesting questions and tasks which can be answered by small modifications of the program we have obtained for the original question. Let us list some of them:

- print digits of a number starting from the less important one (hint: change the order of two instructions);
- print digits of a decimal number in the representation with respect to the base *p*, ($2 \le p \le 10$) – see Fig. 7 (the right column);
- compute the number of ones in the binary representation of a number (or the number of any digit in the representation of a number with respect to any base *p*);
- compute the sum of digits of a number in its representation with respect to any given base *p*.

5 Complexity of Recursive Calculations

We focus here on a very important issue of any computations – complexity – with regard to recursion. In general, recursion is used to considerably speed up computations.

5.1 A Recursive Version of Horner's Rule

In the beginning we shortly explain to our students that computers (processors) perform only 4 basic arithmetic operations and therefore any other operation or function must be calculated using these four operations. Polynomials, whose values can be computed using only these four operations, play a very important role as functions which can be used to approximate any other continuous function, such as sin, cos, tan, log (we usually present some polynomial approximation formulas for trigonometric functions).

Polynomials appear in school mathematics mainly in linear and quadratic equations. Indirectly, polynomials of higher degrees are used to represent numbers in **a positional system** with respect to any base – the representation of a number is **a polynomial** in the base of the system p with coefficients from the set of digits $\{0, 1, 2, \ldots, p-1\}$, see [21] and [22].

For a polynomial of degree n: $w_n(x) = a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n$, expressed as: $w_n(x) = (\ldots((a_0x + a_1)x + a_2)x + \ldots + a_{n-1})x + a_n$ we get the following computational procedure know as Horner's rule to find $y = w_n(z)$:

$$y = a_0,$$
$$y = y \cdot z + a_i \text{ for } i = 1,2,\ldots,n.$$

We can also use the following expression:

$$w_n(x) = (a_0x^{n-1} + a_1x^{n-2} + \ldots + a_{n-1})x + a_n$$

to derive the following recursive procedure for computing the value of a polynomial:

$$w_0(x) = a_0$$
$$w_n(x) = w_{n-1}(x) \cdot x + a_n \text{ for } n \geq 1.$$

We now ask students which of the above formulas for polynomial value calculations is more efficient? To answer this question we ask students to write down all operations when $y = w_3(z)$ is calculated using the recursive algorithm, performed during the recursive call for $n = 3$. It is demonstrated in Fig. 8 that all the recursive calls are nothing else as preparation to execute the usual Horner's rule. Therefore it is no advised to use recursion here.

$$\begin{tikzcd}
\boxed{\text{Recursive calls}} & & & & \boxed{\text{Returns from recursive calls}} \\
w_3(z) = w_2(z) \cdot z + a_3 \arrow[dr] & & & & y = y \cdot z + a_3 \\
& w_2(z) = w_1(z) \cdot z + a_2 \arrow[dr] & & & y = y \cdot z + a_2 \arrow[u] \\
& & w_1(z) = w_0(z) \cdot z + a_1 \arrow[dr] & & y = y \cdot z + a_1 \arrow[u] \\
& & & y = w_0(z) = a_0 \arrow[r] & {} \arrow[u]
\end{tikzcd}$$

Fig. 8. Recursive realization of Horner's rule

5.2 Fast Exponentiation

Fast calculation of a power x^n is a crucial step in real computations, for instance in the RSA systems, where x and n are really very big numbers. In a simple exercise students verify, how long it will take to calculate x^n for a ‘small’ exponent consisting of 35 digits using the ‘school’ method which depends on performing $n - 1$ multiplications. Using the Windows calculator students may calculate that for n consisting of 35 digits, e.g. n = 12345678912345678912345678912345, when a 1 PFlops super computer is used (it performs 10^{15} multiplications per second), it will take more than 10^8 years to find the value of x^n.

We now discuss with students how we can perform exponentiation faster. To direct them to a right answer, we ask how to decrease the number of multiplications when the exponent is an even integer. They quickly come up with the formula $x^{2k} = (x^k)^2$. And then, how to deal with an odd exponent – students also come up quite quickly with the formula $x^{2k+1} = (x^{2k})x$, which transforms this case to the case of even exponent. Now we ask students to use these observations to find how to calculate for instance x^{43}. Repeated application of these rules lead to the following transformations of the power:

$$x^{43} = (x^{42})x = ((x^{21})^2)x = (((x^{20})x)^2)x = ((((x^{10})^2)x)^2)x = (((((x^5)^2)^2)x)^2)x =$$
$$= ((((((x^4)x)^2)^2)x)^2)x = (((((((x^2)^2)x)^2)^2)x)^2)x$$

Therefore, to compute power x^{43}, we have to perform only 8 multiplications, instead of 42 (squaring a number needs one multiplication).

The discussion now leads to the recursive formulation of the above method:

$$x^n = \begin{cases} 1 & \text{for } n = 0 \\ (x^{n/2})^2 & \text{for } n - \text{even} \\ (x^{n-1})x & \text{for } n - \text{odd} \end{cases}$$

Students may wonder how many multiplications are used by this recursive algorithm. To make some suggestions we show another algorithm for the same task in which the binary representation of the exponent is expressed according to Horner’s rule:

$$x^{43} = x^{((((1\cdot 2+0)\cdot 2+1)\ 2+0)\cdot 2+1)\cdot 2+1} = x^{(((1\cdot 2+0)\cdot 2+1)\ 2+0)\cdot 2+1})^2x = x^{((1\cdot 2+0)\cdot 2+1)\ 2+0})^2x)^2x =$$
$$((((x^{(1\cdot 2+0)\cdot 2+1})^2)^2x)^2x = = ((((x^{1\cdot 2+0})^2x)^2)^2x)^2x = ((((x^2)^2x)^2)^2x)^2x$$

It is an interesting to notice that this procedure generates the same sequence of multiplications as the recursive algorithm. Based on the latter interpretation of the power n, one can observe that the number of multiplications depends on the length of the binary representation of n and is equal to the number of binary positions in the representation minus 1 plus the number of 1’s in the representation minus 1. Since the length of the binary representation of n is proportional to $\log_2 n$, the number of multiplication needed to calculate x^n is at most $2\log_2 n$.

Finally we can estimate how many multiplication performs the recursive exponentiation for n = 12345678912345678912345678912345. We have $2\log_2 n < 210$. It is tremendous achievement in complexity what makes RSA cryptographic systems really practical – exponents used in such systems have 200-300 digits.

5.3 Fast Computing of Fibonacci Numbers

The Fibonacci recurrence relation is a multiple recursion since it contains two (that is more than one) recursive calls at a level. It can be seen for instance in Fig. 4, that multiple calls cause a great inefficiency in realization since a large number of the same values is calculated many times. Therefore a multiple recurrence is also called exponential since it leads to non-polynomial computations with regard to the problem size.

Instead of recursion, a Fibonacci number can be easily computed using iteration which starts with two initial values 1 and 1 and adds two last values at each step. There are a number of such procedures based on the same recurrence relation for F_n, see [21] and [22], all of them are of linear complexity in n.

In the previous section however we demonstrate that a linear-time algorithm might be very impractical and we should concentrate on finding a method which is of logarithmic complexity (see [24] for discussion on the role of logarithm in algorithmics). There exists such an algorithm for computing Fibonacci numbers. It is based on a similar idea as used in the recursive exponentiation – if we want to change a linear process into a logarithmic one, we have to find a formula with recursive calls which reduce the parameters values by about at least half. Fortunately there are such relations for Fibonacci numbers:

$$\begin{aligned} &F_0 = 0, F_1 = 1 \\ &F_{2n-1} = F_{n-1}^2 + F_n^2 \qquad n \geq 2 \\ &F_{2n} = 2F_{n-1}F_n + F_n^2 \qquad n \geq 1 \end{aligned}$$

This is a system of two multiple and mutual recursive relations in which the indices of the Fibonacci numbers on the right-hand sides are about half of the indices of the numbers on the left-hand side. Since the relations are multiple, it is advised to implement them using iteration, that starts with the base values. We have perform some computational experiments with various recurrence relation for calculating the values of the Fibonacci numbers (see details in [22]) and the algorithm suggested in the previous sentence conquered all the other algorithms for $n > 30$.

5.4 Divide and Conquer Methods

A divide and conquer algorithm, such as binary search, merge sorting, quick sorting, is usually implemented in a form of a recursive procedure and this approach usually is responsible for a logarithmic factor in a formula for complexity of the algorithm. Two comments are in order.

In general, a computer program which involves recursion is usually shorter and more 'readable' than its iterative counterpart, although its computer execution may take much longer, especially when it involves multiple recursion, as illustrated by Fibonacci number computations.

Logarithm and the logarithmic functions are very important in computing and in computer science. It appears as a measure of data size in computer memory and more important, it describes the number of steps in divide and conquer (recursive) algorithms. Its importance follows from its very slow increase of values when the argument becomes large. The authors wrote another paper on the meaning and the use of logarithms in informatics, in particular in school informatics, see [24].

6 Conclusions

Recursion is one of the main concepts in computer science and remains a challenge for teachers and students. In this paper we discuss various facets of recursion and present some advices, how to approach and to teach recursion, coming from our didactical experience. Knowing the difficulty in introducing, explaining and using recursion, we differentiate our approach, tools, and methods. Our main focus is on recursion as a mental tool for computing – it naturally appears in formulation of some problems and then as a method for their solutions. As a tool for computations, we also focus on complexity of using and implementing recursive procedures.

We do not teach recursion as a separate topic or subject, it is a method and a tool, the way of thinking, used in various situations. The examples of recursion and the way we use them come from our textbooks based on the experience gathered while working with students in K-12 and at university level. Although we have not yet done any formal research on how successful is our approach, we are satisfied with students' grades and the results of final high school examinations in informatics (*matura*). In the near future we plan to run a project on testing various mental tools of computational thinking, such as: approximation, decomposition, recursion, and heuristics.

References

1. ACM K-12 Task Force Curriculum Committee, A Model Curriculum for K-12 Computer Science. ACM (2003)
2. Ben-Ari, M.: Recursion: From Drama to Programs. J. of Computer Science Education 11(3), 9–12 (1997)
3. CSTA: K-12 Computer Science Standards (2011), `http://csta.acm.org/Research/sub/CSTAResearch.html`
4. Diduszko, P.: Rekurencja. System do Nauki Posługiwania się Rekurencją, Master Degree Thesis, Institute of Computer Science, University of Wrocław (2013)
5. Edgington, J.: Teaching and Viewing Recursion as Delegation. J. Computing Sciences in Colleges 23(1), 241–246 (2007)
6. Educational software, `http://mmsyslo.pl/Materialy/Oprogramowanie`
7. Ershov, A.P.: Basic Concepts of Algorithms and Programming to be Taught in a School Course in Informatics. BIT 28, 397–405 (1988)
8. Ford, G.: A framework for Teaching Recursion. SIGCSE Bulletin 14(2), 32–39 (1982)

9. Ginat, D., Shifroni, E.: Teaching Recursion in a Procedural Environment – How much Should we Emphasize the Computing Model? SIGCSE Bulletin 31(1), 127–131 (1999)
10. Ginat, D.: Do Senior, CS Students Capitalize on Recursion? In: Proceedings ITiCSE 2004, Leeds, pp. 82–86. ACM (2004)
11. Gurbiel, E., Hard-Olejniczak, G., Kołczyk, E., Krupicka, H., Sysło, M.M.: Informatics (In Polish), vols. 1 and 2, Textbook for high school, WSiP, Warszawa (2002-2003)
12. Haberman, B., Averbuch, H.: The Case of Base Cases: Why are They so Difficult to Recognize? Student Difficulties with Recursion. In: ITiCSE 2002, Aarhus, Denmark, pp. 84–88 (2002)
13. Haynes, S.M.: Explaining Recursion to the Unsophisticated. SIGCSE Bulletin 27(3), 3–6, 14 (1995)
14. Hsin, W.: Teaching Recursion Using Recursion Graphs. J. of Computing Sciences in Colleges 23(4), 217–222 (2008)
15. Information and Communication Technology in Education. A Curriculum for Schools and Programme of Teacher Development, IFIP/UNESCO (2002)
16. ISTE: `http://www.iste.org/learn/computational-thinking`
17. Leron, U., Zaskis, R.: Computational Recursion and Mathematical Induction. For the Learning of Mathematics 6(2), 25–28 (1986)
18. Naps, T., Roessling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., MCNally, M., Rodger, S., Velazquez, J.: Exploring the Role of Visualization and Engagement in Computer Science Education. SIGCSE Bulletin 35(2), 131–152 (2003)
19. Rubio-Sanchez, M., Urquiza-Fuentes, J., Pareja-Flores, C.: A Gentle Introduction to Mutual Recursion. In: Proceedings ITiCSE 2008, Madrid, pp. 235–239. ACM (2008)
20. Sooriamurthi, R.: Problems in Comprehending Recursion and Suggested Solutions. SIGCSE Bulletin 33(3), 25–28 (2001)
21. Sysło, M.M.: Algorithms. WSiP, Warszawa (1997) (in Polish)
22. Sysło, M.M.: Pyramids, Cones and Other Algorithmic Constructions. WSiP, Warszawa (1998) (in Polish)
23. Sysło, M.M., Kwiatkowska, A.B.: Contribution of Informatics Education to Mathematics Education in Schools. In: Mittermeir, R.T. (ed.) ISSEP 2006. LNCS, vol. 4226, pp. 209–219. Springer, Heidelberg (2006)
24. Sysło, M.M., Kwiatkowska, A.B.: Think logarithmically! Accepted for KEYCIT, Potsdam (Germany) (July 2014)
25. Sysło, M.M.: Outreach to Prospective Informatics Students. In: Kalaš, I., Mittermeir, R.T. (eds.) ISSEP 2011. LNCS, vol. 7013, pp. 56–70. Springer, Heidelberg (2011)
26. Tessler, J., Beth, B., Lin, C.: Using Cargo-Bot to Provide Contextualized Learning of Recursion. In: Proceedings ICER 2013, San Diego, pp. 161–168. ACM (2013)
27. Tung, S.-H., Chang, C.-T., Wong, W.-K., Jehng, J.-C.: Visual Representations for Recursion. Int. J. Human-Computer Studies 54, 285–300 (2001)
28. Yang, F.-J.: Another Outlook on Linear Recursion. Inroads, SIGCSE Bulletin 40(4), 38–41 (2008)
29. Valazquez-Iturbide, J., Perez-Carrasco, A., Urquiza-Fuentes, J.: SRec: An Animation System of Recursion for Algorithm Courses. In: Proceedings ITiCSE 2008, Madrid, pp. 225–229 (2008)
30. Wirth, M.: Introducing Recursion by Parking Cars. SIGCSE Bulletin 40(4), 52–55 (2008)

A Present-Day "Glass Bead Game": A Framework for the Education of Prospective Informatics Teachers Inspired by a Reflection on the Nature of the Discipline

Claudio Mirolo

University of Udine, Dept. of Mathematics and Computer Science
via delle Scienze 206, 33100 Udine, Italy
`claudio.mirolo@uniud.it`
`http://www.dimi.uniud.it`

Abstract. Informatics has an intrinsically multifaceted nature. The usual approaches to plan classwork in the high school are organized around concepts, principles, tools. However, each concept, principle, or tool can be seen under diverse perspectives, in particular those held by mathematicians, scientists and engineers, with quite different implications from a conceptual, methodological and epistemological point of view. Accordingly, the learning issues and the pedagogical means to address them are also different. Hence, it may be worth considering a complementary approach in order to shift the focus from the notions to learn to the (cognitive, methodological, creative) processes required from the students, the latter being tightly connected to the aforementioned general perspectives. The framework outlined in this paper is the basis of a core module as part of the renewed program offered by the University of Udine for the education of prospective teachers of informatics. This module addresses in an integrated way the nature of our discipline and the teaching of programming as a key activity to appreciate the distinctive features of each perspective. It aims at deepening the teachers' awareness about the variety of coexisting views and their pedagogical implications.

Keywords: informatics education, secondary school, prospective teachers, nature of informatics.

Under the shifting hegemony of now this, now that science or art,
the Game of games had developed into a kind of universal language
through which the players could express values
and set these in relation to one another.

Hermann Hesse, "Das Glasperlenspiel" (1943)

1 Introduction

Informatics is an intricate discipline, not only due to the interplay with almost every aspect of human activity, but also because of its intrinsically multifaceted nature: as in the metaphor borrowed from Hesse's masterpiece, the "game" aims at a synthesis of different knowledge fields. The situation is particularly confused in the school [25], where most teachers are not used to reflecting on the actual nature of the activities that they propose under the hat of *informatics.* In fact, *doing informatics* tends to incorporate almost everything that has to do with ICTs, like the development of transferable operational skills (cognitive invariants, general interaction patterns [3, 27]), or even the opportunistic use of specific tools (where the focus is on the product, rather than on the way it is produced [10]). Here our interest is in teaching informatics as a *discipline.* A common approach to plan classwork in the high school starts from some content in terms of concepts, ideas, tools to be introduced and elucidated through lessons as well as practical labs [7, 24, 25]. However, each concept, idea or tool makes only sense in the context of specific kinds of activities it is relevant to, and may be seen differently according to the teacher's (explicit or, more often, implicit) perspective. In essence, the major such perspectives reflect some established practice of mathematicians, scientists and engineers, which have quite different implications from a conceptual, methodological and epistemological point of view [13, 26]. Consequently, and most importantly, the learning issues and the pedagogical means to address them are different. In this paper it is then suggested to integrate a complementary approach in order to shift the focus of attention from the pieces of content to be learnt to the (cognitive, methodological, creative...) processes required from the students, the latter being tightly related to the aforementioned general perspectives. Of course this doesn't mean that concepts and principles are not important, but a task where different perspectives are confused may be exceedingly demanding to students and, at the same time, elusive as to the actual goals of their work. Moreover, it is the experience of such processes that is more likely to produce long-lasting learning effects.

According to these premises, the framework outlined in the next section is the basis of a core module within the renewed program offered by the University of Udine for the education of prospective high-school teachers of informatics. The module addresses in an integrated way the nature of the discipline and the teaching of programming as a key activity to understand it, with the aim of deepening the teachers' awareness about the variety of coexisting perspectives and about the pedagogical implications of such perspectives. The rest of the paper is organized as follows. Section 2 addresses the three main perspectives of informatics as a discipline and some of their implications for computing education. Then, section 3 illustrates possible instantiations of these perspectives in the context of a programming project.

2 Three Disciplinary Perspectives

As mentioned in the introduction, informatics is a multifaceted discipline, part of whose *modi operandi* are drawn from mathematics, science, engineering [8, 9, 13, 16, 26], and it is precisely the *processes* carried out, both cognitively and methodologically, that better characterize these different perspectives. Each perspective, indeed, presupposes a particular view about the nature of the objects of study (*ontology*), the methods applied to acquire new knowledge about them (*methodology*), and the nature of such knowledge (*epistemology*). Among the works cited above, Eden (2007) has contributed an interesting discussion of these topics from a philosophical standpoint [13]. Whether we agree or not on his conclusions in favor of the primacy of a scientific perspective, his analysis lays a good basis to reflect on the cognitive demands and on the pedagogical implications inherent to the tasks assigned to students.

Although there may be "something unsatisfying about thinking of computing as a *blend of three sub-paradigms*" [9], and perhaps a new, richer paradigm results from such an irreducible mix, the originating perspectives still play an essential role in the development of appropriate mental scaffolding. Said otherwise, the acquirement of competencies of the type we find in mathematics, science and engineering is conceivably a precondition in order for a peculiar computational perspective to evolve: "Despite their inseparability, the three paradigms are distinct from one another because they represent separate areas of competence" [8]. In what follows much of the discussion will be concerned with the sphere of programming for a twofold reason: on the one hand, this is a core activity to reflect on the nature of informatics; on the other, it provides a wealth of tasks where each of the points addressed here can be experienced in its concreteness, as we will see in the example of sect. 3.

Our basic question is: What is the nature of the informatics objects? And, more specifically, what kind of "reality" can we ascribe to a *program*? Let us consider, for instance, the following interesting quotation from Abelson's et al. (1984) influential textbook: "A computer language is not just a way of getting a computer to perform operations but rather it is a novel formal medium for expressing ideas about methodology. Thus programs must be written for people to read, and only incidentally for machines to execute" [1]. In this excerpt two divergent views of *program* are contrasted:

– program as a text, "written for people to read" (*program-script* [13]): i.e., a cultural product, which may have an impact on our lives, but only *indirectly*, through some *mindful* and *knowledgeable* intervention of a human actor.
– program as a process, intended "for machines to execute" (*program-process* [13]): i.e., a phenomenal entity, which can affect the real world *directly*, independent of any informed human effort.

At one extreme, as a cultural product intended for humans — a medium to share knowledge under a "procedural epistemology" [1] —, a program (text) can be studied with the analytical methods drawn from the mathematical field. At the latter extreme, as a phenomenal entity, a program (process) can be subjected to scientific experimentation either to discover its properties or to test hypotheses. In both cases, we imagine to deal with a program which is already available in some way. A third view, revealed by the word *structure* in the title of Abelson's et al. book, focuses on the design strategies that can eventually produce as a result an appropriate program structure. In the ensuing subsections we will elaborate in a little more depth on each of these perspectives.

2.1 Mathematical Perspective

So, what is the nature of *programs*, *algorithms*, *information*? What kind of "reality" can we ascribe to these concepts from a mathematical standpoint? Essentially, these are abstract ideas that live in our mind: what mathematicians call *mathematical objects*. The only concrete things we deal with — program texts, flow-charts, trees… — are made up of notation, i.e., are *semiotic representations* [11] that denote mental (mathematical) objects. The main methodological tools to study similar objects are theoretical analysis and logical deduction, that allow us to gain *a priori* knowledge about them, independent of any actual instantiation in a physical computing device. This is the standard approach of theoretical computer science. In summary (table 1):

Table 1. Mathematical perspective (philosophical rationalism)

ontology	semiotic representations denote abstract mathematical objects
methodology	theoretical analysis, logical deduction
epistemology	*a priori* knowledge based on complete formal characterizations

Now a key pedagogical issue, under this perspective, is that students have to understand purely abstract objects. In this respect, it is interesting to elaborate a little on Duval's analysis of mathematics cognition [11, 12]. "From an epistemological point of view there is a basic difference between mathematics and the other domains of scientific knowledge. Mathematical objects [...] are never accessible by perception or by instruments [...]. The only way to have access to them and deal with them is using signs and semiotic representations" [11]. Then, according to him, two kinds of transformations play a central role: *treatment* and *conversion* of representations. Treatments are essentially algorithmic transformations within a given semiotic register. Conversions are based on mappings between different kinds of representations and are cognitively more complex since they presuppose the recognition of the same denoted object, which must be dissociated from the content of its representation. Thus, the peculiar thinking processes of mathematics require the cognitive coordination of different semiotic representations in order to compensate for the lack of direct (or instrumental) access to the denoted entity.

Conceivably, *treatments* and *conversions* play an analogous role in the comprehension of the foundations of computing. This shouldn't be surprising as far as "algorithmic" treatments are concerned. But also conversions are worth considering for two main reasons. Firstly, the basic ideas relating to information coding and algorithmic processing are *abstract* in nature and are to be approached by analogy with mathematical knowledge. Moreover, these ideas are of great relevance since they lay the grounds of any further learning progress in our discipline. Second, in informatics we have constantly to do with mappings and/or conversions between different types of representations of a same (either abstract or real) entity. To sum up, a useful insight of Duval's analysis for the teacher of informatics is that the learning of some abstract object (e.g., information model, algorithm) can be improved by exploring several heterogeneous representations of it — not necessarily restricted to paper-and-pencil "artifacts." Because of the cognitive complexity of the involved mental processes, the teacher should also pay attention to how confident students are while passing from a representation of some kind to one of a different kind that denote the same abstraction.

2.2 Scientific Perspective

We start again from the ontological question: what kind of "reality" can we ascribe to, say, a *program* from a scientific point of view? A program may be simply too complex in order to make reliable predictions only based on formal analysis of its text. Thus, as in the case of natural phenomena, the behavior resulting from its execution can often be partly, or even totally, unknown in advance. (Incidentally, notice that this is always the case when we are debugging a program!) In similar situations a more viable approach is to consider the program process on a par with related *information processing* phenomena that we can find in nature, e.g. DNA transcription or "mental processes", and study it accordingly by testing hypotheses on its behavior through a combination of deductive inference and scientific experimentation. As a consequence, what we learn on a program is a mix of *a priori* and *a posteriori* knowledge, the latter being usually predominant. This approach is typical of some branches of computer science like artificial intelligence. The above points are summarized in table 2:

Table 2. Scientific perspective (philosophy of science)

ontology	program process on a par with information processing in nature
methodology	inference from and experimental testing of hypotheses
epistemology	*a priori* (inferential) and *a posteriori* (experimental) knowledge

Following recent trends in education, central to a scientific perspective are the processes of *scientific inquiry* (or *investigation*) [19, 28, 29], which mainly raise methodological issues. At its heart, the scientific method requires to figure out hypotheses (models) and to test them against evidence derived from experiments (sometimes direct observations), where the hypotheses may be formulated either from previous/theoretical knowledge or from observations. "K-12 science instruction typically focuses on the more basic inquiry skills, including observing, inferring, predicting, measuring, and experimenting" [28]. It is then important that the students develop the abilities to ask testable questions, to design and carry out experiments, to analyze and make sense of the collected data. This is not, however, an easy task for the teachers, who have to be well aware of the relevant learning objectives, that go far beyond the operational skills necessary to achieve the implied practical tasks [2, 4, 6]. In particular, research in science education has pointed out that teachers often tend to confuse (structured) inquiry instruction with hands-on or laboratory activities [28].

In this last respect, the teachers of informatics should not overlook the risk of confusion between the cognitive and operational levels, since we often pay too much attention to "what students can do" rather than to "why they do it that way." As a further remark, the habitual *debugging* activity is worth more careful consideration from a methodological standpoint, and precisely by taking a scientific perspective. In the "diagnostic" phase, indeed, we have in mind a *model* explaining what the program was supposed to do, but the *experimental* evidence from program testing leads us to reject it. Then, in order to identify the problem we have to figure out a more accurate model which is able to account for the actual outcomes of the program. In light of this scientific view, a better structured approach to debugging can be envisaged, where the work planning, the goals, and the reasons for the choices are made explicit. More in general, it is also necessary to distinguish between *testing* in science vs. *testing* in a technological milieu (see next subsection). Although these kinds of practice may look similar in that they use similar equipment, they are in fact quite different both methodologically and epistemologically.

2.3 Technological Perspective

Once more, what kind of "reality" can we ascribe to a *program*? According to Eden's analysis [13], under a technological perspective we can only experience a program as a bunch of data, its script, on some concrete carrier, and it is regarded as useless to postulate the existence of some abstract, immaterial entity denoted by the program text. The engineers' attitude is also pragmatic as to the methodology. Their main concern is the production of reliable systems, which is attained by a regimented development and testing process where the quality is assessed statistically. The complexity of software systems as well as the evolution during their lifespan makes deductive reasoning impractical. As a consequence, we have to content ourselves with *a posteriori* knowledge in terms of reliability. This is the ordinary approach, for example, in the branch of software engineering. The technological perspective is then summarized in table 3:

Table 3. Technological perspective (philosophical empirism)

ontology	the only existing things are bunches of data on concrete carriers
methodology	regimented development and (reliability) testing processes
epistemology	*a posteriori* knowledge based on statistical reliability tests

A weakness of Eden's picture in this case is that the *design* process, almost hidden in a generic "regimented development" process, seems to lose the central role it plays for an engineer. From a disciplinary standpoint, technological knowledge is above all knowledge about the internal *structure* of artifacts and about how such structure realizes a given *function*. Following Kroes' philosophical analysis of this matter, the "dual nature" (structure vs. function) of artifacts "leads to a question that is of crucial importance for understanding the nature of design processes [...]: How can we account for the fact that designers are able to bridge the gap between a functional and a structural description of a technical artefact?" [17]. And the "gap" to bridge may actually be huge for the computing technologies: "computer science is [...] a *synthetic*, an engineering, discipline. [...] Especially important for us are system design problems characterized by arbitrary complexity" [5].

A pedagogical strategy suggested by Frederik et al. [15] in order to address the question raised by Kroes is to categorize the technological artifacts on the basis of their structural and functional properties, and then to try to relate structures to functions. A similar purpose can be achieved in informatics on the basis of *programming patterns* and *roles of variables* patterns, e.g. [18, 20–22]. For more general computing artifacts, it is also interesting to consider the path proposed by Schulte (2012) to explore the structure behind function by experimenting "different steps between use and design [...]: checking out or trial; use or apply; configure or modify; and create or produce" [23]. Besides the issues connected to design, an important component of a "regimented development" process is the implementation of testing sessions. In this respect, the tools available to the teacher are quite simple to use. Then, from an educational viewpoint, the crucial aspect is *planning* a useful set of tests, a task that is again related to the understanding of the structure of the software artifact.

3 An Example

We will now discuss one of the examples that have been proposed to the prospective teachers, in order to illustrate possible instantiations of the three disciplinary perspectives introduced so far through a set of related programming tasks. It will be presented as an integrated path where the students can explore a variety of aspects connected with each such perspective. However, in some contexts it may be unrealistic that the pupils work on all of these aspects to get a broad picture of computer science. Some of the results may then be presented in more or less depth, or some data may be provided directly by the teacher, depending on the specific learning objectives. Alternatively, the students may work in groups on different subtasks, in order to accommodate for different *learning styles* — even though we cannot rely on a clear-cut matching between learning styles and disciplinary perspectives, see for instance the concise survey in [14]. The important point is that the teacher has a clear idea of what sphere of competences (analysis, investigation, design...) her/his students are expected to develop.

The example deals with a well known topic: the performance of sorting algorithms. There are several reasons that can justify a similar choice. On the one hand, sorting problems are quite common in everyday life as well as in the students' experience (e.g., in the use of a spreadsheet); on the other, the related programming tasks are reasonably affordable at the upper secondary level. In addition, the observation of the behavior of basic sorting algorithms can lead to interesting and perhaps surprising discoveries. For the sake of simplicity, we consider only two basic algorithms: *insertion sort* and *quick sort*. Conceivably, the algorithms' behavior can be explained with the aid of animated visualizations and all or part of the code may be made available in advance.

3.1 Working Like a Mathematician

Let us consider the following questions:

MAT1. Are the algorithms able to sort *any* data sequence?
Even in the presence of repetitions?
MAT2. Does it matter what algorithm is used?
And if it does, what is the best-performing one?
MAT3. Is it possible to find a significantly better algorithm?
Or perhaps do there exist inescapable performance bounds?

We are approaching our problems from a mathematical perspective whenever we try to answer similar questions on the sole basis of the analysis of *program text*. In particular, the algorithms' performances are to be assessed in terms of number of comparisons, operations, assignments, iterations, recursions... At the upper secondary level the analysis of the average-case performance, that provides most interesting information to answer the questions labeled MAT2, can be simplified by roughly "adjusting" the (easier to understand) results for the worst-case (insertion sort) or for the best-case (quick sort). It may also be interesting to address the questions MAT3 by trying to estimate the depth of a binary tree with as many leaves as the number of permutations of the data to sort (see fig. 1b, where each node represents a data-pair comparison and the branches correspond to the possible outcomes of the comparison). It is worth remarking that, independent of the terminology, notation and formal precision introduced in the classroom, the computational-complexity classes we denote by $\theta(n\log n)$, $\theta(n^2)$, etc., are abstract mental objects, to which the pedagogical considerations of sect. 2.1 apply. For instance, the graphs drawn in fig. 1a are just *semiotic representations* of our idea of theoretical trend, not the theoretical trends themselves, even though they can reveal key properties related to this concept. Likewise, the tree evoked in fig. 1b is a semiotic representation of an algorithm, focusing on its potential computation paths.

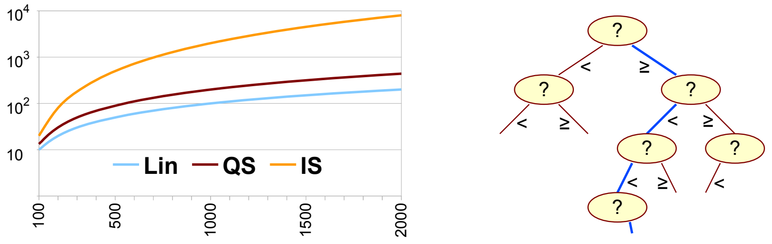


Fig. 1. (a) Juxtaposition of the graphs denoting the theoretical trends $\theta(n)$, $\theta(n\log n)$ and $\theta(n^2)$; (b) Pair-comparison nodes of the tree of all possible sorting paths

3.2 Working Like a Scientist

Under a scientific perspective, examples of pertinent questions are:

Sci1. How is it possible to measure computation times? How reliable are these measures? To what extent can the results be generalized?
Sci2. How accurate are the "trend" models of sect. 3.1? What aspects of the actual behavior can be predicted and what others cannot?
Sci3. Does anything interesting/unexpected emerge from observation? And if it does, how can the discovery be explained?

The questions of group Sci1 guide us to address — and find solutions to — several issues connected with devising an instrument of measure and an experimental equipment, both implemented by a *program* in our context, namely:

– object of the measures (e.g., time costs in the average case);
– orders of magnitude and units of measure (e.g., milliseconds);
– instrumental accuracy (usually coarser than a millisecond) and reliability;
– occurrence of systematic (e.g. generation loops in the program) as well as accidental (e.g., interferences of the operating system's activity) errors;
– size of the sample (e.g., meaningful for measuring average times).

The problems raised by Sci2 suggest a *structured inquiry* [28] to test the "predictions" based on the theoretical models (answers to Mat2) against the experimental evidence gathered to deal with questions Sci1. Moreover, they give the opportunity to discuss how the collected data can be represented in a meaningful way in light of the theoretical models — and, incidentally, this work can be helpful to better understand the theoretical stuff. An example is shown in fig. 2a, that charts the ratio between measured trend and theoretical trend for quick sort. (The more flattened graph is the result of an empirical adjustment of the model.) The last questions, on the other hand, give rise to an *open inquiry* [28] to discover new properties that are not accounted for by the theoretical models, and to try to explain what has been observed. A potential finding is the performance inversion for short data sequences illustrated in fig. 2b. In this respect, it is interesting to ask why the theoretical models weren't able to predict this kind of behavior. More broadly, it is worth noting that here the scientific method is applied to gain new knowledge about the objects of study of our discipline.

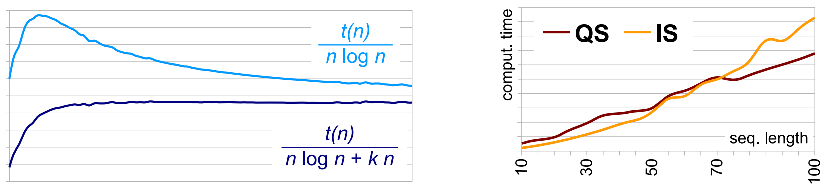


Fig. 2. (a) Observed vs. theoretical trend before/after empirical adjustment of the model (*quick sort*); (b) Observed performance inversion for short sequences

3.3 Working Like an Engineer

We complete this example with a few questions raised from a technological perspective:

Eng1. Is it possible to improve the performance of the "sorting artifacts"? To this aim, how can the available components be assembled?
Eng2. How to ensure the reliability of the devised system? How can the development process be organized?
Eng3. How can the "quality" of component integration be assessed? Which parameters affect the quality? And how?

The questions labeled Eng1 concern the *design* process. Two important aspects are implied. First, engineers apply scientific (and mathematical) knowledge to achieve their goals: the key knowledge in this example is the "performance inversion" observed in the last experience mentioned in the previous subsection. Second, how such knowledge can be exploited is a matter of creativity. The only constraints are established by the *functional* requirements, which include the improvement of performance. Here we can figure out a straightforward solution where the relationships between structure and function are clear: the almost-sorted output of a slightly modified version of the *quick sort* unit provides the input of an *insertion sort* unit that completes the sorting work (fig. 3). The design consists in changing just one line of the code of quick sort (the base-case condition) and then calling the two units in sequence. As to the questions of group Eng2, the reliability of the system can be ensured by testing the output of both units for consistency during the development of the new "system" (e.g., through the widespread *unit-testing* tools; lower ovals in fig. 3). The last questions are concretely addressed on an empirical basis by tuning a threshold, the larger size of a "short" interval in the base-case condition of quick sort, and by measuring the overall system performances (upper ovals in fig. 3). As a final remark, this example also illustrates neatly the different roles of measuring, e.g. times, in science (acquiring knowledge) and engineering (ensuring/improving quality).

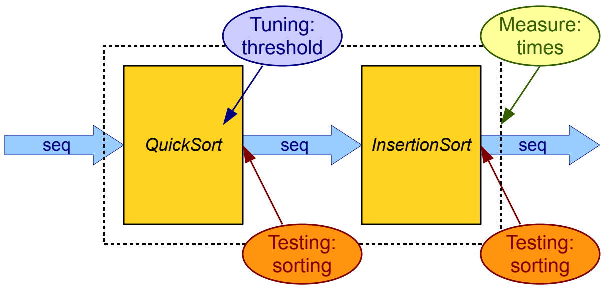


Fig. 3. System design and development process planning

4 Conclusions

The framework outlined in this paper is inspired by a reflection on some pedagogical implications of the manifold nature of informatics. It is intended for the education of prospective high-school teachers, and has been exploited to structure a core module in a renovated program offered to them. The underlying rationale is that the disciplinary content can be deeply understood only in connection with the cognitive and methodological processes that are required from students. A major aim is, therefore, to try to spread light on such processes in order to help the teachers reflect about their practice in general, the organization of hands-on and laboratory activities, the appropriateness of taking different perspectives in a given school context (e.g., technical vs. scientific education) and the cross-disciplinary connections with other school subjects.

All of this provides an unconventional angle from which to approach instructional design, that seems to have stimulated the prospective teachers' interest. In particular, 3 of the 8 students who attended the module in Spring 2013 decided to elaborate on it and conceived inventive paths to explore specific computing problems under different perspectives. Further future work is being planned in order to enrich the framework as to the cognitive and pedagogical dimensions. Moreover, a collection of related examples and materials, to be tested within the prospective teachers' fieldwork experience, is under development.

Epilogue. The central character of Hesse's novel is awarded the title of "Magister Ludi" (Master of the Game). By swapping the words, "Ludi Magister" means a teacher at a Roman school. Maybe the glass bead game is really relevant to school education.

References

1. Abelson, H., Sussman, G.J., Sussman, J.: Structure and Interpretation of Computer Programs. MIT Press, Cambridge (1984)
2. Anderson, R.D.: Reforming science teaching: What research says about inquiry. Journal of Science Teacher Education 13(1), 1–12 (2002)
3. Baudé, J.: Le développement de l'informatique et des technologies de l'information et de la communication dans l'enseignement – et si la voie suivie n'était pas la bonne? Revue Électronique de l'EPI, 95 (2007)
4. Bell, R.L., Blair, L.M., Crawford, B.A., Lederman, N.G.: Just do it? Impact of a science apprenticeship program on high school students' understandings of the nature of science and scientific inquiry. Journal of Research in Science Teaching 40(5), 487–509 (2003)
5. Brooks Jr., F.P.: The computer scientist as toolsmith II. Communications of the ACM 39(3), 61–68 (1996)
6. Crawford, B.A.: Learning to teach science as inquiry in the rough and tumble of practice. Journal of Research in Science Teaching 44(4), 613–642 (2007)
7. Denning, P.J.: Great principles of computing. Commun. ACM 46(11), 15–20 (2003)
8. Denning, P.J., Comer, D., Gries, D., Mulder, M., Tucker, A., Turner, J., Young, P.: Computing as a discipline. Communications of the ACM 32(1), 9–23 (1989)
9. Denning, P.J., Freeman, P.A.: The profession of IT: Computing's paradigm. Communications of the ACM 52(12), 28–30 (2009)

10. Duchâteau, C.: Mais qu'est la didactique de l'informatique devenue? In: Baron, G.L., Bruillard, E. (eds.) Le Technologies en Éducation: Perspectives de Recherches et Questions Vives – Actes du Symposium, pp. 33–42. INRP, Paris (2002)
11. Duval, R.: A cognitive analysis of problems of comprehension in a learning of mathematics. Educational Studies in Mathematics 61(1-2), 103–131 (2006)
12. Duval, R.: La conversion des représentations: un des deux processus fondamentaux de la pensée. In: Baillé, J. (ed.) Du mot au Concept: Conversion, Collection "Sciences de l'éducation", pp. 9–45. PUG, Grenoble (2007)
13. Eden, A.H.: Three paradigms of computer science. Minds and Machines 17(2), 135–167 (2007)
14. Felder, R.M., Brent, R.: Understanding student differences. Journal of Engineering Education 94(1), 57–72 (2005)
15. Frederik, I., Sonneveld, W., Vries, M.J.: Teaching and learning the nature of technical artifacts. International Journal of Technology and Design Education 21(3), 277–290 (2011)
16. Hromkovič, J.: Contributing to general education by teaching informatics. In: Mittermeir, R.T. (ed.) ISSEP 2006. LNCS, vol. 4226, pp. 25–37. Springer, Heidelberg (2006)
17. Kroes, P.: Design methodology and the nature of technical artefacts. Design Studies 23(3), 287–302 (2002)
18. Kuittinen, M., Sajaniemi, J.: Teaching roles of variables in elementary programming courses. In: Proc. of the 9th ITiCSE, pp. 57–61. ACM, New York (2004)
19. Moeed, A.: Science investigation that best supports student learning: Teachers understanding of science investigation. International Journal of Environmental and Science Education 8(4), 537–559 (2013)
20. Muller, O., Ginat, D., Haberman, B.: Pattern-oriented instruction and its influence on problem decomposition and solution construction. In: Proc. of the 12th ITiCSE, pp. 151–155. ACM, New York (2007)
21. Proulx, V.K.: Programming patterns and design patterns in the introductory computer science course. In: Proc. of the 31st SIGCSE, pp. 80–84. ACM, NY (2000)
22. Sajaniemi, J., Ben-Ari, M., Byckling, P., Gerdt, P., Kulikova, Y.: Roles of variables in three programming paradigms. Computer Science Education 16(4), 261–279 (2006)
23. Schulte, C.: Uncovering structure behind function: The experiment as teaching method in computer science education. In: Proc. of the 7th WiPSCE, pp. 40–47. ACM, New York (2012)
24. Schwill, A.: Computer science education based on fundamental ideas. In: Proc. of the IFIP TC3 WG3.1/3.5 Joint Working Conference on Information Technology, pp. 285–291. Chapman & Hall, London (1997)
25. Seehorn, D. (ed.): K-12 Computer Science Standards – Revised 2011: The CSTA Standards Task Force. CSTA & ACM (October 2011)
26. Tedre, M., Sutinen, E.: Three traditions of computing: what educators should know. Computer Science Education 18(3), 153–170 (2008)
27. Vandeput, E.: Milestones for teaching the spreadsheet program. In: Proceedings of EuSpRIG 2009 Conference, pp. 133–143 (2009)
28. VMSC Task Force: Teaching about scientific inquiry and the nature of science: Toward a more complete view of science. The Journal of Mathematics and Science: Collaborative Explorations 13, 5–25 (2013)
29. Windschitl, M.: What is inquiry? A framework for thinking about authentic scientific practice in the classroom. In: Luft, J., Bell, R.L., Gess-Newsome, J. (eds.) Science as Inquiry in the Secondary Setting, pp. 1–20. NSTA Press Book (2008)

Teacher Profiles for Planning Informatics Lessons

Ana-Maria Stoffers and Ira Diethelm

Carl von Ossietzky University, Computer Science Education,
26111 Oldenburg, Germany
{ana-maria.mesaros,ira.diethelm}@uni-oldenburg.de

Abstract. Changing conditions and different educational concepts are widely spread problems of secondary education in Informatics[1]. Due to this fact suitable teacher training can only be designed if the preconditions of Informatics teachers are being considered. Subjective theories play an important part in determining teachers' lesson planning. In this qualitative study subjective theories of Informatics teachers about planning their lessons were examined. After a process of analysis and grouping of results five profiles could be established. They describe the variants of didactic structuring of teaching Informatics determined by varying subjective theories of the teachers. These differ in their perception of the teachers' and students' roles and in the explanations given for chosen teaching methods, contents and learning objectives. With these categories we could distinguish between "the self-confident expert", "the student-oriented manager", "the creative pragmatist", "the inquisitive collector" and "the prudent newcomer".

Keywords: subjective theories, educational reconstruction, designing learning environments.

1 The Role of Teachers' Subjective Theories

For an emerging subject teacher education is crucial. Teacher education for Informatics does not have a long tradition, yet, and many challenges to cope with. In many countries CS or Informatics has been introduced or strengthened in the last few years like in New Zealand, the US, Great Britain and also parts of Germany. First results from New Zealand like [12] for our subject or for the natural sciences like [11] show the big role of teachers and teachers perceptions and personal attitudes in implementing (new) curricula. The framework of Educational Reconstruction for Computer Science Education [3] also lists the teachers' perspective as one central factor for designing Informatics lessons or investigating them.

Lister et al. analyzed in [8] the understanding of teaching of computing academics and found differing ways that CS academics understand teaching. They

[1] Here, we use Informatics synonymously to Computer Science.

suggest "that academics who are aware of the range of understandings will be better able to decide how to design a revision or a new offering."

Quite often teacher have difficulties even to identify themselves as Informatics teachers. Ni's and Guzdial's findings in [10] "indicate that current HS teachers teaching CS courses do not necessarily identify themselves as CS teachers. They have different perceptions related to CS teaching. Four kinds of factors can contribute to these perceptions: teachers educational background and certification, CS curriculum and department hierarchy, availability of CS teacher community, and teachers perceptions about the field of CS."

One phenomenon that confirms that is that teachers do not use the tools and material researchers provide for their teaching, see [7] and [11], not to the expected extend or in the intended way. In order to develop and provide teacher trainings and material for Informatics teachers and their lessons we have to know more about them. We already know that they differ; but not in which way. If we knew different types or profiles how Informatics teachers plan their lessons we could provide suitable trainings and material for different kinds or types of teachers.

In Informatics the planning of lessons on a specific topic differs considerably from teacher to teacher. Dann already in [2] pointed out how important subjective theories for the design of lessons are. Their differences in planning lessons are influenced by teachers' different subjective theories of the subject and by their concepts of how Informatics should be taught. According to Dann "subjective theories of teachers are partially acquired during the formal teacher education, but also during their years as students at school and especially during the years as practicing teachers." This and the fact that there are still many different ways to become an Informatics teacher it is plausible that the subjective theories of Informatics teachers would vary in a wide range, more than in other subjects.

This study deals with the question how these subjective theories can be arranged in groups to provide us with an instrument for describing Informatics teachers. This would be a big help for designing further trainings and material, because they characterize the participants of in-service training. For successful courses and long-term changes they must take the participants' prerequisites and requirements into account and start off from there.

Informatics as a subject in secondary schools in Germany consists of topics including programming, but also using and understanding systems that process information as computers and networks in general, it therefore also deals with modeling, interpretation and theoretical, social and legal aspects of computing and automation, see [1]. Thus, the topic "networks / Internet" served us as an important topic for teaching on the one hand. And on the other hand, even though it is not a central part of many national or international curricula, it left much space for different perceptions what content is connected with it, how lessons could be designed, what focus is set and how this decisions are made.

To give an insight into the study we begin by discussing the research question and then go on to describe briefly the design of the study that consists of the analysis of semi-structured interviews as a basis for the development of the profiles. The design and conduction of the interviews has already been described in detail in previous papers by us [9]. In this paper, we focus on the steps for creating profiles which we describe in section 4 and the detailed description of the profiles as a result of this process in section 5. We end with conclusions and an outlook.

2 The Research Question

To determine the subjective theories of Informatics teachers we put our emphasis on their structuring of Informatics lessons and on the importance they attach to students' conceptions. Based on Kelly [5] and Groeben [4] we define subjective theories as individual cognitive structures of self- and world views that have the function of explanation and prediction.

Our central question therefore is: What are teachers' subjective theories about designing Informatics lessons?

Using semistructured interviews, we asked Informatics teachers how they would structure and carry out a a number of lessons for the topic "networks and the Internet". The chosen focus is an important topic which offers a great and creative variety of teaching approaches and at the same time has seldom been on the agenda of courses for teacher training.

The selection of interview partners was determined by various characteristics which will be explained in the following chapter. The objective was to find different subjective theories of teachers. And therefore it was more important to find a wide spectrum of opinions rather than to fine one particular opinion more often than another one.

3 The Sample

For the purpose of finding different subjective theories on designing CS lessons it is not important how often one particular subjective theory is represented but how much these theories differ from each other. Our criterion for selection therefore was the probability of considerably differing theories.

We interviewed fifteen teachers altogether, three of them were female and twelve male. One of them worked at a junior high school, four at comprehensive schools and ten at senior high schools. The youngest was 27 and the oldest 57 years old. Five teachers were teaching in junior high at grade 6 as the lowest level. Another five teachers had been teaching Informatics in both junior and secondary high schools. The remaining five teachers were teaching in senior high schools only.

The teachers' experience was distributed quite evenly: Six had a teaching experience of less than five years, seven had been teaching Informatics for a period of five to twenty years. Two of the teachers answered that they had been teaching Informatics for a period of twenty to thirty years.

The question about their additional teaching subjects revealed that nearly all of them were teaching mathematics as well. Of the fourteen teachers teaching mathematics along with Informatics seven were also teaching physics. And only two of the teachers had been trained (at university or in-service) to teach Informatics.

The ways towards teaching Informatics or achieving the qualification to teach Informatics appeared to be very different. Five teachers do not have a formal qualification to teach Informatics and are either in charge of an optional study group in Informatics or teach Informatics in a compulsory optional course. They learned the content and teaching of informatics on their own. Five teachers had either studied Informatics as an additional subject during their time at university or had acquired a basic degree in Informatics. One teacher had participated in an intensive two-year in-service training for teachers of Informatics and had obtained the formal qualification for teaching Informatics. Four teachers said they had studied Informatics at university.

4 The Process of Building the Profiles

The individual steps of the evaluation are represented in the flow chart below, see fig. 1. The first step is the coding of the data. Here categories are being assigned to passages in the text. A category is a term which by being assigned to certain text passages helps to understand empirical phenomena of the data.

The transcribed interviews and the coding guidelines are the necessary input for the evaluation. In the first loop of the flow chart the whole data is being searched. In the coding process either an existing category is being assigned to a passage or a new category is being generated for coding this passage. When the whole text has been coded the individual categories are dimensionalized. In a second loop all categories are being searched and dimensionalized. We will explain this process of dimensionalization with an example later on in this section.

After coding the interviews there are four more steps to be performed to get the types aimed at. Fig. 2 shows the steps of creating types, see [6].

Coding the data however is not sufficient for understanding the phenomenon under investigation. The coding process scans the empirical field, but without subsequent evaluation it remains a mere description. For realizing the objective of our research, which is to present the whole spectrum of subjective theories of Informatics teachers, it becomes necessary to develop a typology.

According to [6, p:85] we understand typology as "the result of a grouping process where a range of objects is classified on the basis of one or several features into groups or types, aiming at the greatest possible similarity within one particular type and the greatest possible dissimilarity between the individual types". Since a type does not describe a person but a particular kind of teaching design, it is necessary to state clearly that the evaluation of this study does not refer to types of teachers as persons. The generated types describe nothing more than *a way of designing lessons* in Informatics, so that one single teacher can represent several types.

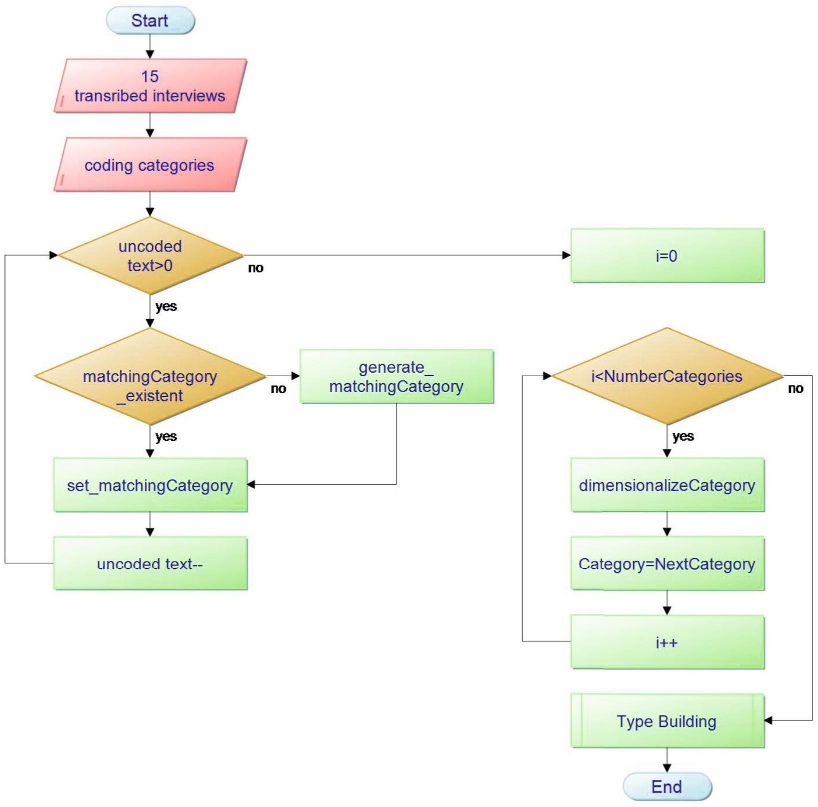


Fig. 1. Evaluation steps

As fig. 2 shows the first step is the development of relevant analyzing dimensions. This is achieved in two steps. After a deductive coding of the transcribed interviews a second coding of the interviews took place, this time inductively. Thus the given categories of the interviews were extended by new categories, based on the data. These extended categories we call dimensionalized, see [6]. The similarities and dissimilarities of the teachers can be described on the basis of these dimensions. An example for these two steps in developing relevant analyzing dimensions will be described as follows:

In the first step a category *learning objectives* may have been found. This only shows that the teacher in that interview said something about the learning objectives he or she meant to achieve in his or her teaching. This category on its own does not show if there are any other teachers who considered the same learning objectives important. Therefore the second step is necessary. In the second step the category of *learning objectives* can be split up into several categories like learning objectives in the fields of content knowledge or comprehension or

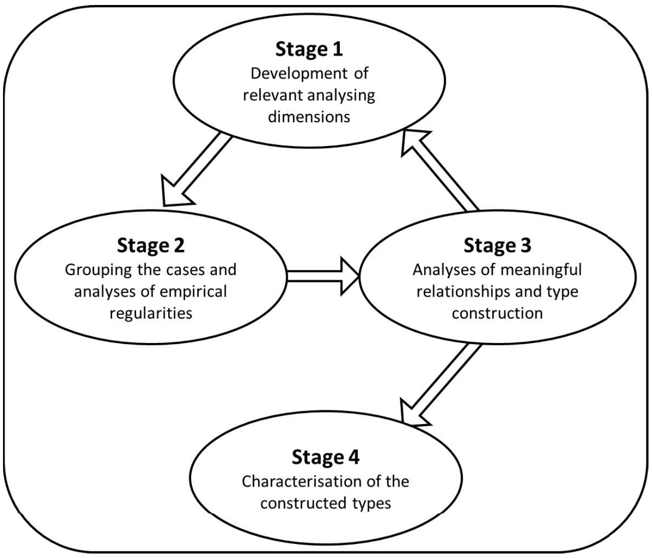

Stage 1
Development of relevant analysing dimensions
Stage 2
Grouping the cases and analyses of empirical regularities
Stage 3
Analyses of meaningful relationships and type construction
Stage 4
Characterisation of the constructed types

Fig. 2. The Construction of Types [6]

the use of the Internet. These new categories are used for analyzing the data and comparing the different interviews.Therefore we choose among others the dimensions: teaching methods, learning objectives, student's role, learning objectives and teaching content.

The second step in type building consists of grouping the cases and analyzing the empirical regularities. The analysis of the empirical regularities of the cases is based on the dimensions mentioned above. They can describe empirical differences or similarities between the cases. A similarity might be that in some interviews teacher describe the same teaching objectives or the same role. On this basis the grouping of the cases can be carried out. You then have to check whether internal homogeneity and external heterogeneity have been achieved. Internal homogeneity is given if, as an example, the interviews of one group describe the same teacher's role and interviews of another group show a different role of the teacher in class.

In a third step an analysis of meaningful relationships is the basis for generating types. After the second step, when cases were grouped on the basis of empirical regularities, an analysis of meaningful internal and external relationships within the groups is the next step. Grouping the cases is not sufficient when the aim of the study is to understand these relationships. This analysis may lead to new perspectives considering the grouping of cases or the dimensions of comparison. Therefore these first three steps can be carried out several times. For example you may associate during this step a described teachers' role of one group with the defined students' role in the same group of cases.

After having established the types, the last step is to characterize them. In spite of using the method of type-building for the evaluation, the sample is too small to really qualify as complete type-building. The field of Informatics teachers could not be covered representatively. Therefore we will use the term profiles instead. They serve as a first grouping and analysis of Informatics teachers and will be taken as a basis for further research.

5 Profiles of Informatics Teachers

The profiles are described on the basis of five resulting dimensions:

– The teacher's role in the planning and the teaching process
– The student's role in the teaching process
– The methods of Informatics teaching
– The choice of content in Informatics teaching
– The intended teaching objectives in Informatics teaching

During the evaluation process these dimensions proved to be adequate for describing the performance of a teacher and for emphasizing the differences between the generated profiles. But also there are similarities between individual profiles, for example the learning objectives between the profile *The Self-Confident Expert* and the profile *The Creative Pragmatist* are quite similar but the methods of teaching used to achieve the learning objectives differ in many aspects. As a result these five profiles are as follows:

5.1 The Self-Confident Expert

This profile is characterized mainly by self-confidence, confidence in one's own knowledge and expertise. Teachers with this profile show a high degree of professional and didactic competence. Their didactic decisions are well-founded and their choices can be justified at any point. These teachers deliberately chose to teach Informatics and are passionate about Informatics in any situation.

They understand their **teaching-role** as guiding and controlling the learning process. They feel responsible not only for presenting the subject matter but also for motivating the students. The whole learning process is controlled by the teacher, the learning objectives are clear and the students are skillfully guided towards achieving them.

Under the guidance of the teacher **the students** work independently, but they carry out relatively short and set assignments. They rely on the special knowledge which they learned from their teacher.

The **teaching methods** are varied and always adjusted to the stage of the learning process and above all well-founded and justified.

The **learning content** is being chosen to demonstrate the nature of Informatics to the students. The objective is to give every student an idea of what Informatics is about.

The **learning objective** is to achieve a basic understanding of the content of Informatics. The need for this basic understanding is motivated by the conviction that the fundamental ideas of Informatics are a part of general education.

5.2 The Student-Oriented Manager

This profile is characterized by strong confidence in students. Teachers consider their students to be independent, responsible young people who are capable of deciding what they want to learn. Some students' competence is rated so highly that they are allowed to take over the teacher's role. In these cases the teacher leaves not only the professional but also the didactic decisions to the students.

The **teacher's role** is to attend his students' learning process without restricting their independence. The teacher permits students to design and guide the teaching and intervenes only when necessary. They believe their students capable of choosing the learning content sensibly and teaching themselves.

The **students** are totally independent. They have the chance of planning their learning process completely on their own. They take over the teacher's role and plan their lessons.

The **teaching methods** result from the students' independence. Often projects are carried out which are a product of individual work. Teachers with this profile do not prescribe teaching methods.

The **content** is determined by the students' wishes. Their interests decide about the choice of topics.

The **learning objectives** result from the independent project work of the students. This teaching concept enables students to acquire many competences like the ability to work independently and to present the results. Problem solving also plays a part in project work.

5.3 The Creative Pragmatist

Teachers with this profile are characterized by their endeavor to offer lively and hands-on teaching. Their teaching focuses on problem-orientated learning. The students are to find out for themselves what the new contents mean to them and why it makes sense to learn them. In order to achieve this, the teacher tries to make his lessons exciting, motivating and student-activating.

The **teacher's role** is to find fascinating methods and tasks which will motivate the students. The teacher has a very clear idea of the kind of problem that will motivate the students to understand certain contents and consider them important.

The **students** discover the contents through the tasks they are given. They think about how to understand and solve the given problems. They are purposefully motivated to learn and think about problems.

The **methods** depend on the contents, for this type is very conscious of the context between method and contents. Therefore methodical considerations depend on content considerations and are chosen accordingly.

The **contents** are to be the basis of Informatics. Teachers with this profile select contents which are part of a general education, topics which especially those students should know about who are not going to choose Informatics as their priority.

The one **learning objective** above all teaching contents and didactic decisions is to educate students to become responsible adult citizens.

5.4 The Inquisitive Collector

Characteristic for this profile is great confidence in researchers and specialists. Teachers with this profile are very interested in new research results about teaching and learning processes and like integrating them into their teaching.

It is the **teacher's role** to use good teaching materials and be informed about new didactic ideas. They attend in-service training quite often where they learn how to present certain content and teach accordingly. Their teaching materials do not need to be adapted or changed at all since the scientists designed them well enough.

The **students** learn a lot from these good teaching materials. They take part in a well-structured teaching process which is systematized by precise instructions and tasks.

The **teaching methods** result from the material provided at in-service trainings.

The **contents** are also determined by in-service training courses. The courses reflect the contents which play a part in didactic discussions at university and therefore are of great importance for the subject at that point in time.

The **learning objective** is for the students to achieve basic knowledge in the respective fields of Informatics.

5.5 The Prudent Newcomer

This profile is distinguished by a prudent selection of contents and methods. Teachers with this profile are highly motivated and attempt to use all their knowledge and creativity to teach well. They do not have much teaching experience on the one hand but on the other hand the didactic principles they learned are still fresh in their minds and are used in their teaching design.

They have the necessary professional and didactic knowledge to design **their teaching** according to the students' ideas. They take the students' ideas on the topics into account and their teaching is directed at changing these, if necessary.

Under the teachers' strict guidance the **students** work to master the contents. They are being motivated to think carefully about their own conceptions of the topic and revise them.

The adopted **methods** match the contents and the students' ideas. Great importance is set on methodological diversity.

In their **selection of contents** teachers with this profile follow the curriculum which has been agreed on by the Informatics faculty at their school. If that curriculum offers alternative contents, these teachers choose contents which they feel confident about and can teach well.

The main **learning objective** is to change existing students' conceptions of a particular topic, for these are not considered to be suitable.

6 Conclusions and Future Work

The determined profiles give a first impression of the characteristics that influence how Informatics teachers structure their teaching. No teacher corresponds completely with one profile, which means that *pure* profiles are not to be found. They are exaggerated, so to speak, in order to emphasize the respectively relevant features of teaching design.

Comparing the encountered profiles with the professional identity of CS teachers' described by Ni and Guzdial [10] we can state that *the prudent newcomer* and *the self-confident expert* both understand themselves as CS teachers'. In our results the self-awareness of being a CS teacher can lead to self-confidence on the one hand or to prudence on the other.

Although these profiles only allow a first insight, they are nevertheless helpful for designing teacher trainings. They can be used for better adjusting the training to the teachers. Not all profiles will be represented at each training course, but it can be assumed that these profiles will apply to the participating teachers. A good training course must be addressed to all these profiles.

As all of our interview partners were attendees of some of our in-service trainings it must be pointed out that one or more profiles could be missing because they might belong to teachers that are not attending in-service trainings.

Having constructed the profiles now further steps are necessary. One objective of our study is to compare the profiles with existing didactic concepts to design teacher trainings for Informatics teachers. The resulting guidelines will enable each didactic specialist to check his further education course to ensure that it takes all profiles into account.

For example, the analysis of the profiles showed that teacher of different profiles need to learn from each other. As an exaggeration the *self-confidence* teacher could learn from the *inquisitive collector* the benefit of using ready-made materials. On the other hand the *inquisitive collector* could learn how to expand the materials or he could get the self-confidence to create them on his own. Therefore a guideline for in-service CS teachers' trainig might be to give teachers' the opportunity to learn from each other.

In addition to these guidelines another objective of this study is to design an exemplary teacher training on the topic of networks and the Internet that addresses all these profiles. Then, this might be used as a guideline for planning trainings for Informatics teachers in general.

References

1. Brinda, T., Puhlmann, H., Schulte, C.: Bridging ICT and CS: Educational standards for computer science in lower secondary education. In: Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2009, pp. 288–292. ACM, New York (2009), `http://doi.acm.org/10.1145/1562877.1562965`
2. Dann, H.D.: Subjective theories and their social foundation in education. In: von Cranach, M., Doise, W., Mugny, G. (eds.) Social Representations and the Social Bases of Knowledge, pp. 161–168. Hogrefe & Huber, Lewiston (1992)
3. Diethelm, I., Hubwieser, P., Klaus, R.: Students, teachers and phenomena: Educational reconstruction for computer science education. In: Proceedings of the 12th Koli Calling International Conference on Computing Education Research, Koli, Finland (2012)

4. Groeben, N., Scheele, B.: Dialogue-hermeneutic method and the "research program subjective theories". Forum Qualitative Sozialforschung/Forum: Qualitative Social Research 1(2), 9 (2001), http://nbn-resolving.de/urn:nbn:de:0114-fqs0002105
5. Kelly, G.A.: The psychology of personal constructs. Routledge, London (1991)
6. Kluge, S.: Empirically grounded construction of types and typologies in qualitative social research. Forum Qualitative Sozialforschung/Forum: Qualitative Social Research 1(1) (2000), http://www.qualitative-research.net/index.php/fqs/article/view/1124
7. Levy, R.B.B., Ben-Ari, M.: 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 2007, pp. 246–250. ACM, New York (2007), http://doi.acm.org/10.1145/1268784.1268856
8. Lister, R., Berglund, A., Box, I., Cope, C., Pears, A., Avram, C., Bower, M., Carbone, A., Davey, B., de Raadt, M., Doyle, B., Fitzgerald, S., Mannila, L., Kutay, C., Peltomäki, M., Sheard, J., Simon, K., S., Traynor, D., Tutty, J., Venables, A.: Differing ways that computing academics understand teaching. In: ACE 2007: Proceedings of the Ninth Australasian Conference on Computing Education, pp. 97–106. Australian Computer Society, Inc., Darlinghurst (2007)
9. Mesaroş, A.M., Diethelm, I.: Exploring computer science teacher's subjective theories on designing their lessons. In: Bezakova, D., Kalas, I. (eds.) Proceedings of the 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP 2011, Selected Papers. Library and Publication Centre, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava (2011)
10. Ni, L., Guzdial, M.: Who am I?: Understanding high school computer science teachers' professional identity. In: Proceedings of the 43rd ACM Technical Symposium on Computer Science Education, SIGCSE 2012, pp. 499–504. ACM, New York (2012), http://doi.acm.org/10.1145/2157136.2157283
11. Parchmann, I., Gräsel, C., Baer, A., Nentwig, P., Demuth, R., Ralle, B., The ChiK Project Group: "Chemie im Kontext": a symbiotic implementation of a context-based teaching and learning approach. International Jounal of Science Education 28(9), 1041–1062 (2006)
12. Thompson, D., Bell, T., Andreae, P., Robins, A.: The role of teachers in implementing curriculum changes. In: Proceedings of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE 2013, pp. 245–250. ACM, New York (2013), http://doi.acm.org/10.1145/2445196.2445272

Extracurricular Activities for Improving the Perception of Informatics in Secondary Schools

Carlo Bellettini[1], Violetta Lonati[1], Dario Malchiodi[1], Mattia Monga[1], Anna Morpurgo[1], Mauro Torelli[1], and Luisa Zecca[2]

[1] Dip. di Informatica, Università degli Studi di Milano
[2] Dip. di Scienze Umane per la Formazione, Università degli Studi di Milano-Bicocca

Abstract. In order to introduce informatic concepts to students of Italian secondary schools, we devised a number of interactive workshops conceived for pupils aged 10–17. Each workshop is intended to give pupils the opportunity to explore a computer science topic: investigate it firsthand, make hypotheses that can then be tested in a guided context during the activity, and construct viable mental models. This paper reports about how we designed and conducted these workshops.

1 Introduction

Several voices have been recently raised to urge a new understanding in schools of the nature and scope of informatics. A report issued by the UK Royal Society on UK schools [14] has documented clearly how a focus on the instrumental value of Information and Communication Technologies (ICTs) failed to lead pupils and teachers to develop any real knowledge of computing sciences. While informatics and computational thinking have the potential to be very formative for pupils, and several core aspects of computing are sufficiently basic to be taught as a fundamental subject in all the secondary schools, among the general public—but sometimes also among the teachers who are responsible of conducting courses of informatics[1]—the scientific discipline is often blurred by the use of office automation tools, or the Internet communication facilities and their social impact. Although these activities might indeed need special skills, they can be presented and mastered without referring to computing at all. Therefore, to be able to show to pupils the appeal of a rigorous scientific discipline we believe it is important to focus again on the basics of informatics. Thus, we started devising some enrichment program activities aimed at presenting and discussing the core of informatics as the *"automatic processing of information"*.

We wanted to expose to informatics a variety of pupils of different ages, not necessarily involved in a computing curriculum or with a previous knowledge of

[1] We recently asked to a group of prospective teachers of informatics to define the discipline with a statement: 5 out of 17 identified informatics with communication technologies or other devices and a recurring theme was the emphasis on the improvement "of life quality". Common misconceptions among teachers are reported also by [13].

information or programming, therefore we developed an approach based on playful activities which imply a mix of tangible and abstract object manipulations: a strategy which we call *algomotricity* [6,4,5].

Table 1 summarizes the grades of the pupils to which we proposed the workshops. Each activity was designed to focus on one fundamental concept:

information What is information? How can symbols/numbers be used to represent it?
processing How can information be manipulated/changed in order to produce new knowledge?
automation Which manipulations can be performed by a *mechanical* interpreter? How this can be done?

Table 1. Workshops proposed

Grade	4th–7th	8th–11th	12th–13th
information	Wikipasta	Human Pixels	Human Pixels (advanced)
automation	Mazes	Mazes (advanced)	
processing		Greedy Money (simplified)	Greedy Money

The workshops are described further in Section 3, here we just recall their main focus:

Wikipasta. In this workshop pupils are posed the problem of describing the typographic aspect of a text. By playing with pieces of pasta and other small objects, they are led through a game to the discovery of mark-up languages and then introduced to a lightweight "wiki" syntax. The final activity on the computer is about editing Wikipedia-like pages.

Human Pixels. After being shown a video of animations made in stadiums by coordinated soccer teams supporters (so called "human LCD"), pupils are asked to discuss how to set up a very simplified version of such animations. They eventually discover grids, sampling, resolution, compression and complete the activity by using a multi-view editor showing a picture along with different representations as a matrix of numbers.

Mazes. In this workshop pupils are faced with the problem of guiding someone through a simple maze. Pupils first focus on the task of verbally guiding a *human robot* (a blindfolded mate) through a simple path. Initially they are allowed to freely interact with the robot, then they are requested to propose a very limited set of primitives and to compose them into a program to be executed by the robot, with the possibility of exploiting three basic control structures (if, repeat-until, repeat-n-times). After this, pupils are provided with a visual programming language (a simplified version of MIT Scratch) and are asked to write programs guiding a sprite through mazes of increasing complexity.

Greedy Money. In this workshop pupils are requested to think about greedy strategies: they start by proposing an algorithm for the change-making problem with a set of coins that admits a greedy solution. Then they try to apply the same strategy to a scheduling problem and evaluate its suitability in finding the optimal solution.

Initially we tested such workshops in a lower-level and in an upper-level secondary school [4,5]. As a result, we could refine the design of these activities, and we prepared a set of two-hours workshops which we proposed to classes of 20–25 pupils. In 2013 a total of 26 classes attended our workshops, the activities are ongoing and we have currently planned 26 workshops also for this year.

The article is organized as follows: Section 2 discusses our methodological approach, Section 3 details the activities carried out in the workshops, and Section 4 draws some conclusions.

2 Methodological Approach

All the proposed workshops share a common strategy, which we call *algomotricity*. As the name suggests (this neologism is a portmanteau combining *algorithm* and *motoric*), this approach exploits kinesthetic learning activities [1], having the aim of informally exposing pupils to a specific informatic topic, followed by an abstract learning phase devoted to let students build their mental models of the topic under investigation and a final computer-based phase to close the loop with their previous acquaintance with applications.

Our approach took inspiration from several papers in computer science education (for example [3,7,12,2]), and it is clearly rooted on the *Experiential Learning Theory* (ELT), specifically on *Problem based learning* (PBL). ELT defines learning as "the process whereby knowledge is created through the transformation of experience. Knowledge results from the combination of grasping and transforming experience. ... Immediate or concrete experiences are the basis for observations and reflections, these reflections are assimilated and distilled into abstract concepts from which new implications for action can be drawn. These implications can be actively tested and serve as guides in creating new experiences" [11]. PBL designs an educational environment based on experiential learning organized around the investigation, explanation, and resolution of meaningful problems. In PBL, students work in small collaborative groups and learn what they need to know in order to solve a problem [10].

In our workshops we designed *allosteric environments* [9] advocating a non-direct transmission of knowledge in favour of active work on part of the learner who constantly reworks her/his mental models in order to learn something new. Following this methodology, the teacher has the very delicate task of avoiding any predefined, generic way of presenting concepts. Rather, the instructor has to adapt to each pupil in order to give her/him suggestions interfering with her/his (mis)conceptions, setting up a so-called "didactic environment" [9]. This is typically done by putting pupils in new situations and assign them goals to reach whithout (or with just a few) hints about how to reach them.

Thus the teacher is a sort of mediator [9] who, having clearly in mind the goal of an activity, helps pupils toward this goal without forcing them to follow a specific path[2]. This requires a trade-off between free exploring and external constraints: the didactic environment should suitably limit the available degrees of freedom so that pupils can effectively and proficiently explore the solutions' space without either getting lost or having the feeling that there is only one *right* answer. Moreover pupils should have a real possibility to make mistakes.

This kind of set-up fits particularly well our learning goals, where pupils are facing a somehow new subject, as we are focused on *internal* aspects of algorithmics and information representation and/or processing, and not on learning how to use specific computer applications.

All activities ended with a computer-based phase, where pupils were confronted with specially conceived software in order to exploit what they had learned during previous phases. This is also necessary in order to match at least in part the expectations of pupils, who often identify informatics with the use of a computer.

Pupils were asked to work in groups, to encourage confrontation and peer-knowledge exchange, in all steps of the activities. The steps should be carefully designed according to Vygotsky's concept of *zone of proximal development* [15]. This requires a suitable choice of the difficulty of each step, both avoiding too gradual a path, resulting in bored pupils losing interest in the overall activity, and steep steps causing pupils to get lost. Moreover, the importance of the teacher's flexibility emerges also in this respect, because a same sequence could be adequate for some pupils and not for other ones. It is also very important to plan each step so that the activity makes sense *per se* in order to keep the pupils engaged [8].

3 Description of Workshops

This section describes more in depth the workshops activities we proposed (except for "Wikipasta", which was extensively presented in [5]). Each of the workshops, lasting two hours and conceived for groups of approximately 20–25 students, was carried out in a room with enough space to ensure that pupils could move around in order to perform the tasks assigned to them.

The workshops start with a preliminary discussion devoted to let pupils express how they perceive informatics: at the beginning the conductor introduces the ethymology of the word "Informatics" and explains it as the *"automatic processing of information"* and pupils are asked to write on a sticky note their definition of the term on which the workshop is going to focus (see Table 1). All definitions are gathered together and clustered on a whiteboard, highliting common keywords. At the end of this process the conductor summarizes these keywords in order to let a group definition emerge from the notes written by pupils.

[2] Rather, the instructor should be able to exploit unexpected events to point out relevant issues not necessarily foreseen in the original design.

The core of each workshop consists of a phase where pupils engage in one or more game-like activities, followed by a teamwork session based on the interaction with expressly developed software tools.

A discussion phase ends each workshop. Its aims are to let pupils both reconsider their initial point of view about informatics and recognize a link between the game-like and the computer-based tasks.

3.1 Human Pixels

At the beginning of this workshop, pupils are shown a video of animations made by a group of soccer supporters in Korea[3]. In this video, each supporter wears a special shirt with different colors in its front and back part, where a third color can be shown by pulling two lateral flaps. When looked at from sufficiently afar, the group is perceived as an image and each supporter acts as a pixel. As supporters change the color of their shirts in sync and with sufficient velocity, the final result is an animation.

Pupils are then asked to form small groups (two or three persons) and to propose a method for showing a simplified version of such an animation, where each pupil of the class uses a black and a white paper sheet instead of the above mentioned shirt. The goal is to reproduce a simple message (*e.g.,* "Hello") through subsequent visualizations of the letters occurring in it. After each group has presented its idea, all pupils take a collective decision about how to organize the animation.

Although occasionally one of the proposed methods suggests to work with only black (or only white) sheets and changing each time the position of the students to let them shape the various characters (essentially discovering a representation in which every "pixel" is positioned relatively to another, as in Fig. 1(a)), pupils most of the times opt for:

1. building the equivalent of a black-and-white *bitmap* representation for each character occurring in the message (see Fig. 1(b));
2. positioning chairs in a rectangular grid;
3. assigning each student/pixel to a chair;
4. letting each student take note or memorize if the pixel s/he represents has to be set or reset in correspondence of the various characters, that is when s/he will have to use the black and when the white sheets.

Typically pupils also understand that some form of synchronization is necessary, and achieve it through a person who beats the time. A camera films the students so that they can test if the animation renders correctly (see Fig. 1(c)). Indeed, all pupils got a working animation, often also experimenting with variations such as animations with sliding characters.

As a final activity, pupils are introduced to a software especially conceived for editing bitmap images in a multiview interface. This interface shows a simple

[3] See `http://www.youtube.com/watch?v=tipHJlLUzNk`; we actually use a shortened version.

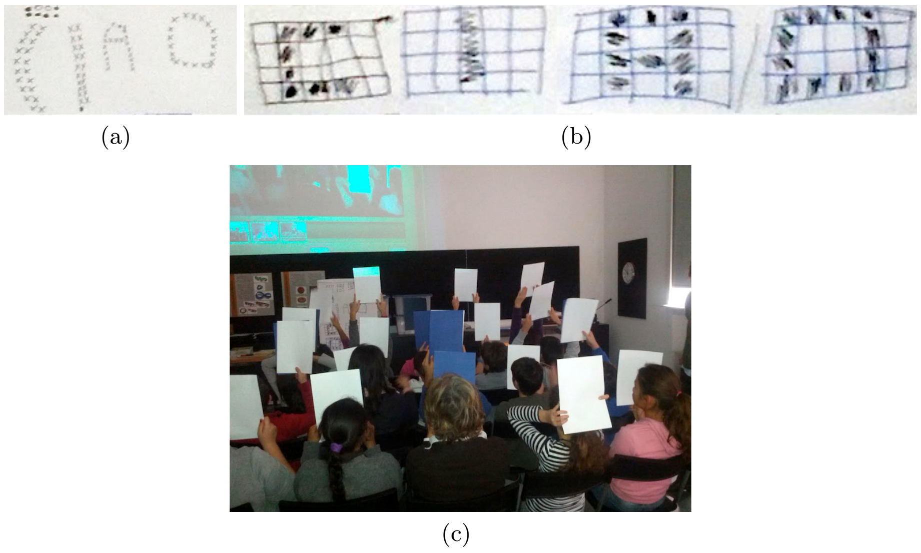


Fig. 1. A sample of proposed "relative" (a) and bitmap (b) representation of the characters occurring in the animation. In (c) some pupils realizes a frame in the animation.

image along with different rasterized representations, such as a bitmap- and a RGB-based, respectively for black and white and for color images, and a compressed view based on a run-length encoding. Pupils are asked to play with the tool and discover how to modify an image by working on each of the different representations.

3.2 Mazes

This workshop focuses on programming. The kinestethic activity consists in letting pupils drive a mate through a given path. Pupils are divided into two or three groups of approximately the same size, and within each group the following actors are chosen:

- a *robot*, who, blindfolded, will have to follow the given path;
- a *driver*, speaking aloud instructions to the robot;
- a *care taker*, ensuring that the pupil playing the robot does not get hurt or fall.

The path is chosen according to the age of pupils, ranging from a simple L-shape to S- or 8-shape (see Fig. 2(a–c)). Moreover, the task includes the additional requirements that the *robot* grab one object placed in a fixed position and sit on a chair at the end of the path.

Pupils perform two rounds of such activity: in the first one they interact freely with the *robot*; in the second one they are asked to write a *program* to be read by the *driver* and executed by the *robot*, with the following constraints:

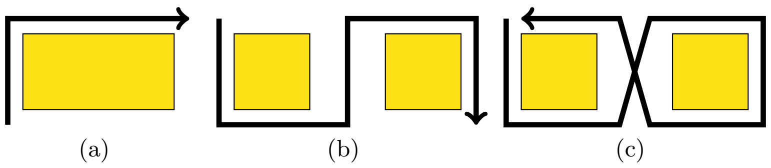


Fig. 2. Different paths for the algomotorial activity in the *Mazes* workshop: (a) L-shaped. (b) S-shaped. (c) 8-shaped. Paths are shown in increrasing order of difficulty.

- each group has to agree beforehand on the sequence of instructions that the *driver* will give the *robot*;
- they must work using at most four different instructions: students are given sticky notes of four different colors, and each time one of a certain color is used, it must always carry the same instruction;
- basic control structures (if, repeat-until, repeat-n-times) are available to simplify the program: the elementary semantics of these are explained;
- from a syntactic point of view, each instruction has to be written on a sticky note, stacking notes on a paper sheet in order to build a sequence of instructions, in case writing control structures around the sticky notes they refer to (see Fig. 3).

As a final step, after pupils have checked that their program allows the *robot* to correctly carry out the task, each of the programs set up by a group is executed by the *robot* of another group. Although groups have worked on the same maze, swapping programs has the effect of highlighting the following facts:

- programs are often specially tailored on some characteristics of the *robot* (*e.g.,* its step or shoe size),
- instructions are sometimes ambiguously or not precisely expressed (*e.g.,* turn left, without specifying the angle).

In the second part of the workshop, pupils are introduced to a simplified version of the Scratch programming language[4] with the task of moving an Aladdin lamp shaped sprite in virtual mazes of increasing complexity (shown in Fig. 4(a)–(e)). Pupils in pairs are challenged to write programs letting the sprite reach the exit of the mazes with the aim of minimizing the number of used instructions. Such minimality constraint lets pupils naturally apply specific control structures such as repeat-n-times and repeat-until, respectively in the mazes shown in Fig. 4(c),(d), and (e).

3.3 Greedy Money

Groups of 2/3 pupils are asked to write down the algorithm they automatically execute when giving a change with the aim of using the smallest number of

[4] `http://scratch.mit.edu/`

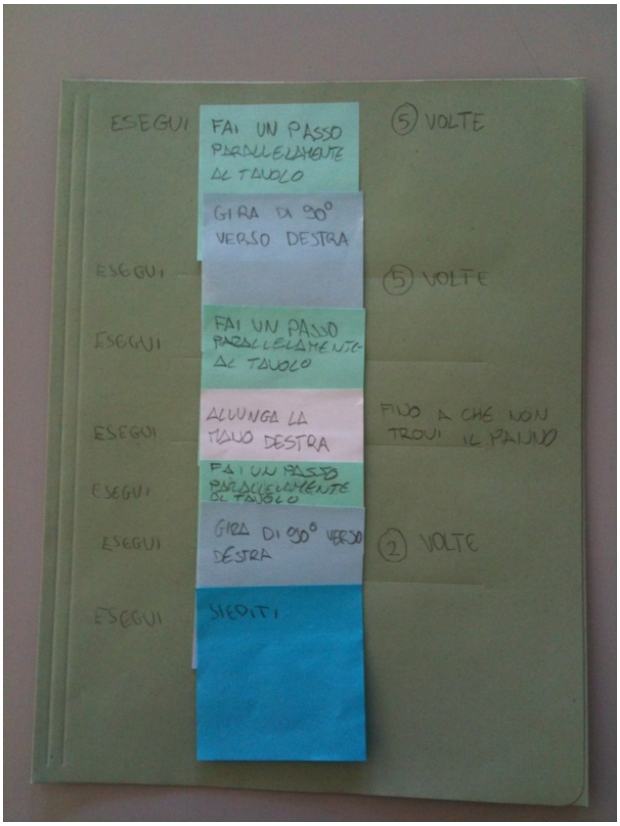

ESEGUI
FAI UN PASSO PARALLELAMENTE AL TAVOLO
5 VOLTE
GIRA DI 90° VERSO DESTRA
ESEGUI
5 VOLTE
ESEGUI
FAI UN PASSO PARALLELAMENTE AL TAVOLO
ESEGUI
ALLUNGA LA MANO DESTRA
FINO A CHE NON TROVI IL PANINO
ESEGUI
FAI UN PASSO PARALLELAMENTE AL TAVOLO
ESEGUI
GIRA DI 90° VERSO DESTRA
2 VOLTE
ESEGUI
SIEDITI

Fig. 3. A "program" containing sequences of instructions and control structures

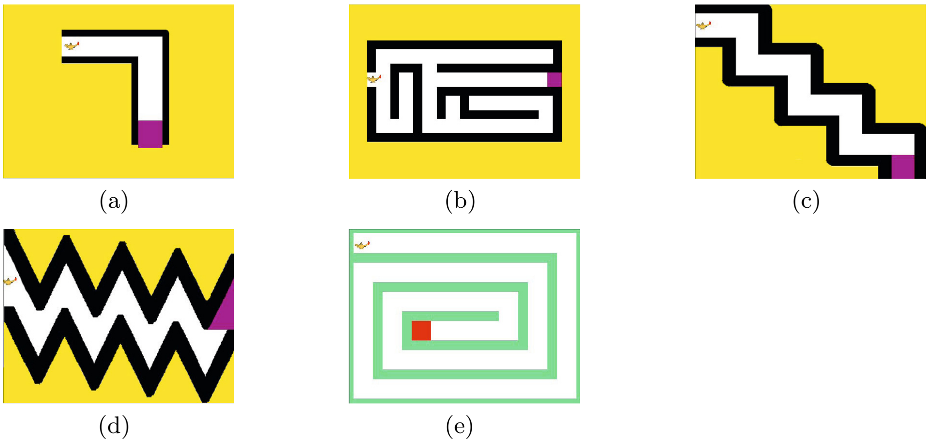


Fig. 4. Mazes used in Scratch. (a) L-shaped. (b) Standard maze. (c) Staircase. (d) Saw. (e) Spiral.

coins/bills. A set of play money can be used in order to test the algorithm through a step-by-step execution. Each group, in turn, describes its algorithm while the remaining ones test it. Most groups propose a procedure that manages to accumulate the change through subsequent additions of one coin or bill having the highest value yet not exceeding the residual change. Some use a more efficient approach exploiting the remainder of the division between the residual change and a coin/bill nominal value.

Thus the conductor can highlight commonalities, specifically:

1. the initial solution was the empty set;
2. coins and bills have been initally sorted in decreasing order of their nominal value;
3. coins and bills have been considered in such order, adding them to the solution until their value was higher than the residual change.

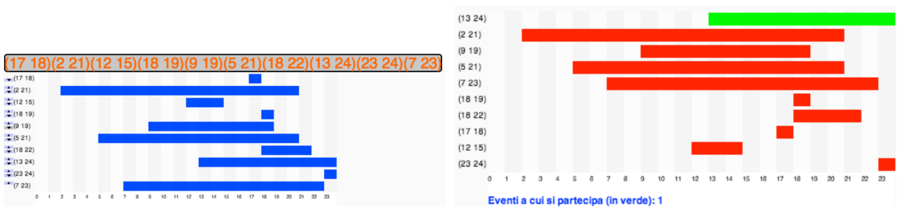
At this point, the conductor can generalize such approach to a more abstract procedure which builds the solution to a generic optimization problem by considering a set of objects in a given order, and for each of them deciding whether or not it has to be added to the solution according to a feasibility criterion. It should be emphasized that this constitutes an example of a *greedy procedure*, in that each object is considered only once for (possibly multiple) addition to the solution, without any possibility to remove objects previously added to the solution.

The second part of the activity consists in asking groups to apply such approach to a scheduling problem, namely that of maximizing the number of movies to be seen in a film festival whose program contains several, partially overlapping movies. Pupils are guided to find the analogies between this problem, the one concerning money change and the abstract description of a greedy procedure. Table 2 highlights such analogies: for instance, pupils tend to easily spot that movies, as well as coins/bills, play the role of objects. Analogously, the feasibility criteria are that of checking whether: i) a movie overlaps with other ones already in the solution, and ii) the considered coin/bill has a nominal value smaller than the residual change. Things change when considering the sorting order: the one based on nominal value naturally emerges in the first problem. On the second one there are several alternatives: starting or ending time , number of intersections with the remaining movies, or movies' length (either ascending or descending). Once all alternatives are clear, groups are asked to verify which criteria ensure the greedy solution to maximize the number of seen movies. More precisely, pupils are asked to find counter-examples to drop non-optimal criteria. A software supports the execution of this activity, generating at random a set of movies (cfr. Fig. 5(a)), rearranging them according to a chosen sorting criterion, and applying the greedy procedure (cfr. Fig. 5(b)).

Consider for instance the case shown in Fig. 5(b). With movies sorted according to decreasing overlapping numbers, finding a counter-example is easy. The greedy procedure would suggest to view only one movie (the highlighted one on the top), discarding all the remaining ones. Anyway, it would be possible to see four different movies: the fifth, the seventh, the eighth and the ninth from the

Table 2. Comparison between the scheduling and money change problem and the more general greedy approach

	Change problem	*Scheduling problem*
Objects	coins/bills	movies
Sorting order	nominal value, decreasing	several alternatives
Feasibility criterion	value not greater than the residual change	no overlapping between one movie and the ones already added to the solution


(17 18)(2 21)(12 15)(18 19)(9 19)(5 21)(18 22)(13 24)(23 24)(7 23)

Eventi a cui si partecipa (in verde): 1

Fig. 5. The software showing a randomly generated set of movies (a) and applying them the greedy procedure according to a selected sorting criterion (b)

top. Students in different groups tend to find counter-examples for all criteria, except the one based on increasing ending time. The conductor can thus show an informal proof of the optimality of such criterion. In a final recap, students are warned about the fact that a greedy procedure does not always lead to the optimal solution. This happens for instance if using the non-optimal criteria for the film festival problem. It could also happen for the money change problem when considering a different set of coins/bills. Examples of such money system are the imperial British one or the one described in Harry Potter's books.

4 Conclusions

The power of informatics lives in its interplay between abstraction and concreteness: abstract ideas are implemented by concrete programs and real word machines. When presented in secondary schools, however, this power is often hindered by a predominant focus on using computer applications, thus informatics risks to be perceived as a bag of ready-to-use recipes, with almost no space for creativity, understanding and cleverness. *Algomotricity* is our attempt to present abstract symbolic manipulations in a very concrete way, but *without starting from computers.* Computers come at the end, just to close the conceptual loop with the previous acquaintance of the pupils with the ubiquitous ICT tools. We designed our workshops with a twofold goal:

- propose a methodological approach to informatics teachers, and
- present some core aspects of informatics to pupils of different grades and their teachers.

From a broader perspective, we aim at conveying a view of informatics as a scientific discipline, as opposed to the current perception of this field. We also paid attention in designing workshops requiring resources that are commonly available or easy to prepare (pasta, colored sheets, ...), and the software used is downloadable for free and runs on standard PCs. All classes participated with engagement in the proposed activities, collaborating, discussing different points of view, and showing satisfaction for their achievements. Many pupils admitted that their idea of informatics and of the job of ICT professionals was different before they attended the workshop. For example, the idea of informatics as a challenging discipline where creativity and collaborative work are often needed was new to many. A teacher in a lower secondary school, in an interview we made after a workshop, told us:

> "I did some experiments in teaching mathematics by concrete tasks, and I found them very important at this age, because pupils have great difficulties with abstraction, especially in the first year. The language of informatics is hard for my pupils because it is symbolic, abstract, it requires precision... They use computer applications by trial and errors, without reflection. From the workshop, I believe pupils learned that it is difficult to avoid ambiguities: for them it is difficult to put themselves in others' shoes..."

In fact, while informatics in Italy has been recognized as an independent academic discipline since the '70s, teachers with an informatic background are rather rare in non-vocational schools. Thus, while other abstract disciplines, like mathematics, have a certain tradition and established practices in secondary education, no such culture exists for informatics. Thus, we believe our workshops could represent a valuable experience to support teachers, and we want to use them as the starting point for a further action research.

Acknowledgments. We would like to thank the interuniversity research center MATEMATITA (`http://www.matematita.it/`) who hosted the workshops, and all the schools who partecipated to the activities.

References

1. Begel, A., Garcia, D.D., Wolfman, S.A.: Kinesthetic learning in the classroom. In: Proc. of the 35th SIGCSE TSCSE, pp. 183–184. ACM, New York (2004), http://doi.acm.org/10.1145/971300.971367
2. Bell, T., Rosamond, F., Casey, N.: Computer science unplugged and related projects in math and computer science popularization. In: Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D. (eds.) Fellows Festschrift 2012. LNCS, vol. 7370, pp. 398–456. Springer, Heidelberg (2012), http://dl.acm.org/citation.cfm?id=2344236.2344256
3. Ben-Ari, M.: Constructivism in computer science education. Journal of Computers in Mathematics and Science Teaching 20(1), 45–73 (2001)

4. Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M.: Exploring the processing of formatted texts by a kynesthetic approach. In: Proc. of the 7th Workshop in Primary and Secondary Computing Education, WiPSCE 2012, pp. 143–144. ACM, New York (2012)
5. Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Morpurgo, A., Torelli, M.: What you see is what you have in mind: constructing mental models for formatted text processing. In: Diethelm, I., Arndt, J., Dünnebier, M., Syrbe, J. (eds.) Informatics in Schools ISSEP 2013 — Selected Papers, Commentarii Informaticae Didacticae, vol. 6, pp. 139–147. Universitätsverlag Postdam, Germany (2013)
6. Lonati, V., Monga, M., Morpurgo, A., Torelli, M.: What's the fun in informatics? Working to capture children and teachers into the pleasure of computing. In: Kalaš, I., Mittermeir, R.T. (eds.) ISSEP 2011. LNCS, vol. 7013, pp. 213–224. Springer, Heidelberg (2011)
7. Curzon, P., McOwan, P.W., Cutts, Q.I., Bell, T.: Enthusing & inspiring with reusable kinaesthetic activities. In: Proc. of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE 2009, pp. 94–98. ACM, New York (2009), http://doi.acm.org/10.1145/1562877.1562911
8. De Vecchi, G., Carmona-Magnaldi, N.: Faire construire des savoirs (2ème ed.). Hachette Education, France (2003)
9. Giordan, A.: From constructivism to allosteric learning model. In: UNESCO Conference on Science Education 2000+ (1996), http://www.ldes.unige.ch/ang/publi/articles/unesco_AG_96/unesco96.htm
10. Hmelo-Silver, C.E.: Problem-based learning: What and how do students learn? Educational Psychology Review 16(3), 235–266 (2004)
11. Kolb, D.A., Boyatzis, R.E., Mainemelis, C., et al.: Experiential learning theory: Previous research and new directions. Perspectives on Thinking, Learning, and Cognitive Styles 1, 227–247 (2001)
12. Pattis, R.E.: Karel the Robot: A Gentle Introduction to the Art of Programming, 1st edn. John Wiley & Sons, Inc., New York (1981)
13. The CSTA Curriculum Improvement Task Force: The new educational imperative: Improving high school computer science education. Tech. rep., Computer Science Teachers Association (February 2005), http://csta.acm.org/Communications/sub/DocsPresentationFiles/White_Paper07_06.pdf
14. The Royal Society: Shut down or restart? The way forward for computing in UK schools (January 2012), http://royalsociety.org/education/policy/computing-in-schools/report/
15. Vygotsky, L.: Mind in Society: Development of Higher Psychological Processes. Harvard University Press, Cambridge (1978)

Author Index