Kendra M. L. Cooper
Antonio Bucchiarone *Editors*

Software Engineering for Games in Serious Contexts

Theories, Methods, Tools, and Experiences

Springer

Software Engineering for Games in Serious Contexts

Kendra M. L. Cooper • Antonio Bucchiarone
Editors

Software Engineering for Games in Serious Contexts

Theories, Methods, Tools, and Experiences

Springer

Editors
Kendra M. L. Cooper
Kelowna, BC, Canada

Antonio Bucchiarone
Fondazione Bruno Kessler
Trento, Italy

ISBN 978-3-031-33337-8 ISBN 978-3-031-33338-5 (eBook)
https://doi.org/10.1007/978-3-031-33338-5

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

Paper in this product is recyclable.

Preface

Serious games and gamification are two distinct but related areas of game development that aim to harness the power of games to achieve goals. Serious games are designed to educate, train, or inform players about real-world topics, such as healthcare, safety, or environmental issues. *Gamification*, on the other hand, involves applying game elements and design principles to non-game applications for employee training, marketing, or education to increase engagement and motivation. Both serious games and gamification have gained increasing attention in recent years due to their potential to create engaging and immersive experiences that can lead to better learning outcomes, increased productivity, and positive behavioral change. As such, they have become a popular tool in a range of fields, including education, finance, healthcare, and transportation; they are expected to continue to grow in importance in the years to come.

Serious games and gamified applications are inherently complex due to their interdisciplinary nature. They draw upon contributions from arts, behavioral sciences, business, education and training, engineering, humanities, physical sciences and mathematics, and specific targeted domains such as healthcare, transportation, and so on. To help address this complexity, a new research community has emerged that spans software engineering and games in serious contexts. This community investigates software engineering approaches that are tailored for serious games and gamification, with respect to achieving beneficial user experiences.

This book is designed as an accessible means to introduce readers to and/or deepen existing knowledge about software engineering for games in serious contexts. The book is intended for software engineers, game developers, educators, and anyone interested in how games in serious contexts can be effectively created. It covers a wide range of topics, from game design principles to software architecture, testing, and deployment, and is structured into two parts. These topics are spread on 11 essential chapters contributed by 45 authors who are from 9 different countries—Brazil, Canada, Germany, Greece, Italy, Mauritius, Mexico, the United Kingdom, and the United States of America.

The book starts with **Chap.** 1, written by Kendra M. L. Cooper, which introduces the interdisciplinary research community that has emerged at the intersection of software engineering and games, with a focus on serious games and gamified systems. At the same time, it summarizes examples of recent research (2016–2022) on these topics and organizes it around established software engineering research topics. The chapter also provides an overview of the book's content, which includes 11 core chapters on various aspects of gameful systems.

Immediately after the introductory chapter, the volume is organized in two parts: **Part I** delves into various aspects of designing, maintaining, adapting, and evaluating games in serious contexts, while **Part II** focuses on the experiences of realizing and using games in serious contexts.

Part I: Topics on the Design, Maintenance, Adaptation, and Evaluation of Gameful Systems

Chapter 2 by Sotiris Kirginas presents various methods for evaluating users' experiences in serious games during and after development. The methods covered include qualitative, quantitative, subjective, objective, short term, long term, summative, and formative. The chapter also offers guidance on when to use these different user experience assessment methods during the development cycle.

Chapter 3 by Vipin Verma et al. introduces a new content-agnostic framework for engineering serious educational games. The framework allows for the creation of games by reusing existing educational game mechanics and separates the game into three independent components: mechanics, content, and student modeling.

Chapter 4 by Leticia Davila-Nicanor et al. presents a method for designing and analyzing the architecture of serious mobile games. The technique presented exploits established design patterns like Wrapper, Singleton, and MVC to optimize the distribution of memory and execution time resources, and architecture analysis is done using graph theory and software metrics. By evaluating the architecture early, costs can be reduced and software performance can be improved. For this purpose, a dispersion diagram is used to visualize the architecture with acceptable quality levels.

Chapter 5 by Edward Melcer et al. proposes a serious game-based online platform called ENTRUST, designed to assess trainees' decision-making competencies in various entrustable professional activity domains in the healthcare education field. The chapter covers the platform's design, insights, lessons learned throughout the development process, and results from a pilot study of the platform. The pilot study shows that ENTRUST can discriminate between levels of surgical expertise and can serve as an objective assessment approach for clinical decision-making.

Chapter 6 by Michael Miljanovic et al. proposes a generalized model for adapting existing serious games to utilize machine learning approaches without the need to rebuild the game from scratch. The chapter presents an approach to engineer machine learning-based serious games and discusses five common challenges that arise in the process, along with possible solutions. The challenges include selecting data for the machine learning input, choosing game elements to adapt, addressing the cold start problem, determining the frequency of adaptation, and testing that an adaptive game benefits from machine learning.

Part II: Topics on Experiences with Gameful Systems

Chapter 7 by Vanissa Wanick et al. explores the potential to transfer strategies commonly utilized in entertainment games to serious games. The chapter presents three complementary perspectives regarding emerging aspects of player agency, serious game modification, and transferability across different contexts, including emergent possibilities, modding, and the importance of AI emotion modeling to inform better game design.

Chapter 8 by Leckraj Nagowah presents a serious game called Code-Venture, designed to help players learn the basics of coding and improve their programming skills. A teacher's application is included to monitor, assess, evaluate, and store the player's performance data. Pre- and post-game surveys show that the game is useful and engaging, with an increase in the number of students interested in pursuing a programming career and finding programming easy.

Chapter 9 by Bruce Maxim et al. presents a method for revising traditional lecture-based game design courses to utilize flipped classroom models. The revised courses use agile software engineering practices and gamification to design, implement, and test game prototypes through active learning and role-play. The effectiveness of the revised courses is measured using surveys, which show that students attending flipped classes are slightly more engaged with the course materials than those in lecture-only classes. Additionally, students who interact with the active learning course materials feel more able to apply their knowledge than those in traditional lecture courses.

Chapter 10 by Riccardo Coppola et al. presents a gamified framework for manually exploring GUI testing of Web-based applications, which is implemented and evaluated. The framework improves test coverage, reduces the number of bugs identified, and provides a positive user experience, as indicated by the participants in the evaluation.

Chapter 11 by Mathias Eggert et al. presents a field experiment that involved gamified quality improvement of five student projects in an information systems course using leaderboards. The project members were interviewed to capture the impact of using a leaderboard on their programming behavior, and the results indicate a motivational effect with respect to improving code quality and a reduction of code smells.

Chapter 12 by Gabriele Costa et al. presents a serious game for cybersecurity education called "A NERD DOGMA." It is an escape room adventure game based on real security scenarios, which requires the player to exfiltrate data, break ciphers, and intrude remote systems to gain experience in security operations. The game also includes the introduction of security tools, integration of third-party technologies, and mimicking external environments.

Perspectives

The book highlights several challenges and opportunities for the field of software engineering for games in serious context. One of the primary challenges is to develop effective methods for evaluating serious games and measuring their impact and outcomes. Another challenge is to design serious games that are both engaging and effective, which requires a deep understanding of game design principles and instructional design. The book also emphasizes the need to develop effective software engineering practices for serious game development and the importance of gamification in improving user engagement and motivation. Additionally, the potential of serious games for addressing societal challenges such as cybersecurity and healthcare is highlighted.

Despite these challenges, the book also identifies several opportunities for the field, including the potential of serious games to provide new and innovative approaches to learning and the potential of serious games to address real-world problems in new and effective ways. Overall, the chapters in the book provide a valuable snapshot of the current state of the field and offer insights into where it may be headed in the future. While there are still challenges to overcome, the opportunities for researchers and practitioners in the field of software engineering for games in serious context are exciting and numerous as presented in **Chap.** 13 by Antonio Bucchiarone.

We hope that this book will be a valuable resource for anyone interested in designing, developing, or using games in serious contexts. We believe that serious games have the potential to transform education, healthcare, social awareness, and environmental conservation, and we hope that this book will contribute to their continued development and adoption.

Acknowledgments

The editors would like to express their deepest gratitude to all of the authors who have submitted their valuable contributions and provided their highly qualified expertise in reviewing the chapters included in this book.

Kelowna, BC, Canada — Kendra M. L. Cooper
Trento, Italy — Antonio Bucchiarone

Contents

Editors and Contributors

About the Editors

Kendra M. L. Cooper is an Independent Scholar whose research interests span software and systems engineering (requirements engineering and architecture) and engineering education; these topics are explored within the context of game engineering. Dr. Cooper's current topics include graph-based approaches such as biologically inspired adaptable software architecture models and ontological foundations for serious educational games. She has co-edited the book *Computer Games and Software Engineering* (2015) and edited the book *Software Engineering Perspectives in Computer Game Development* (2021). Dr. Cooper has co-organized/organized ICSE Workshops on Games and Software Engineering (GAS 2012, GAS 2013, GAS 2015, GAS 2016, GAS 2022, GAS 2023) and the CSEE&T Workshop on Essence in Education and Training (2020). She has served as an Editor for the journal *Software: Practice and Experience*. Dr. Cooper received a Ph.D. in Electrical and Computer Engineering from The University of British Columbia, Canada.

Antonio Bucchiarone is a Senior Researcher at the Motivational Digital Systems (MoDiS) research unit of the Bruno Kessler Foundation (FBK) in Trento, Italy. His research activity is focused principally on many aspects of software engineering for adaptive socio-technical systems. Dr. Bucchiarone's current topics include advanced methodologies and techniques supporting the definition, development, and management of gameful systems in different domains (i.e., education, sustainable mobility, software modeling, etc.). He was the General Chair of the 12th IEEE International Conference on Self-Adaptive and Self Organizing Systems (SASO 2018) and has co-organized/organized multiple workshops including Models@run.time at MODELS 2021; Ensemble-based Software Engineering (EnSEmble) at ESEC/FSE 2018 and 2019; Microservices: Science and Engineering (MSE) at STAF2018 and SEFM2017; and the ICSE GAS 2022, GAS 2023 workshops. Dr. Bucchiarone is an Associate Editor of the IEEE *Transactions on Intelligent Transportation Systems*

(T-ITS) journal of the *IEEE Software Magazine* and of the *IEEE Technology and Society Magazine*.

Contributors

Ashish Amresh Arizona State University, Phoenix, AZ, USA

Luca Ardito Politecnico di Torino, Turin, Italy

Ajay Bansal Arizona State University, Phoenix, AZ, USA

Tyler Baron Arizona State University, Phoenix, AZ, USA

Jeremy Bradbury Ontario Tech University, Oshawa, ON, Canada

Fatyma Camacho University of California, Santa Cruz, CA, USA

Riccardo Coppola Politecnico di Torino, Turin, Italy

Gabriele Costa IMT Lucca, Lucca, Italy

Diksha Cuniah University of Mauritius, Réduit, Mauritius

Leticia Davila-Nicanor Universidad Autónoma del Estado de México, Atizapán de Zaragoza, Mexico

Hyrum Eddington Stanford University School of Medicine, Stanford, CA, USA

Mathias Eggert Aachen University of Applied Sciences, Aachen, Germany

Tommaso Fulcini Politecnico di Torino, Turin, Italy

Giacomo Garaccione Politecnico di Torino, Turin, Italy

Martin Haase Aachen University of Applied Sciences, Aachen, Germany

Irene Aguilar Juarez Universidad Autónoma del Estado de México, Texcoco, Mexico

Oleksandra G. Keehl University of California, Santa Cruz, CA, USA

Sotiris Kirginas University of Athens, Athens, Greece

James R. Korndorffer Stanford University School of Medicine, Stanford, CA, USA

Melissa Lee Stanford University School of Medicine, Stanford, CA, USA

Cara A. Liebert Stanford University School of Medicine, Stanford, CA, USA

Dana T. Lin Stanford University School of Medicine, Stanford, CA, USA

Sochitl Cruz López Universidad Autónoma del Estado de México, Texcoco, Mexico

Abraham Banda Madrid Universidad Autónoma del Estado de México, Atizapán de Zaragoza, Mexico

Bruce Maxim University of Michigan-Dearborn, Dearborn, MI, USA

Edward F. Melcer University of California, Santa Cruz, CA, USA

Sylvia Bereknyei Merrell Stanford University School of Medicine, Stanford, CA, USA

Michael Miljanovic Ontario Tech University, Oshawa, ON, Canada

Maurizio Morisio Politecnico di Torino, Turin, Italy

Leckraj Nagowah University of Mauritius, Réduit, Mauritius

Marina Ribaudo University of Genoa, Genoa, Italy

Samuel Shields University of California, Santa Cruz, CA, USA

James Stallwood University of Southampton, Southampton, UK

Marco Torchiano Politecnico di Torino, Turin, Italy

Amber Trickey Stanford University School of Medicine, Stanford, CA, USA

Jason Tsai University of California, Santa Cruz, CA, USA

Joel Ayala de la Vega Universidad Autónoma del Estado de México, Texcoco, Mexico

Vipin Verma Arizona State University, Phoenix, AZ, USA

Vanissa Wanick University of Southampton, Southampton, UK

Martin Wolf Aachen University of Applied Sciences, Aachen, Germany

Guilherme Xavier PUC-Rio, Rio de Janeiro, Brazil

Jeffrey Yackley University of Michigan-Flint, Flint, MI, USA

Philipp Zähl Aachen University of Applied Sciences, Aachen, Germany

Chapter 1
Introduction to Software Engineering for Games in Serious Contexts

Kendra M. L. Cooper

Abstract Software engineering researchers have been actively investigating novel approaches that focus on the effective development, evolution, and maintenance of high-quality, complex systems for over 50 years. Recently, an interdisciplinary research community has emerged that spans software engineering and games. This community addresses a broad range of issues that prevail in developing games for entertainment, serious games, and gamified applications. In this book, the focus is on the latter two. Serious games are also known as games with a purpose. Beyond their entertainment value, they also fulfill a purpose such as educating or training users on specific learning objectives. Gamified systems are non-entertainment applications that are enhanced with game elements to help motivate and engage users to improve their productivity, satisfaction, time on tasks, and so on. Although distinct research topics, serious games and gamification share a core quality of service attribute: user experience. These applications possess the inherent, interdisciplinary complexity of creating user experiences that engage and motivate users to accomplish specific goals.

This introductory chapter begins with a brief presentation of background material covering serious games, gamified systems, and a description of their inherent interdisciplinary development nature. This is followed by a summary of examples for recent advances that are reported in peer-reviewed publications (2016–2022) at the intersection of software engineering and gameful systems. The results are organized around established software engineering research topics. In addition, this chapter provides an overview of the book structure and content; brief summaries of the 11 core chapters are included.

Keywords Software engineering · Serious game · Gamified systems · User experience · Interdisciplinary research · Background material

K. M. L. Cooper (✉)
Independent Scholar, Vancouver, BC, Canada

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_1

1.1 Introduction

Since the late 1960s, software engineering researchers have been actively investigating novel approaches that focus on the effective development, evolution, and maintenance of high-quality, complex systems. At a high level, topics include engineering activities (e.g., requirements engineering, architecture and design, construction, testing), umbrella activities (e.g., configuration management, lifecycle processes, project management, software quality assurance, traceability), frameworks and platforms, metrics, models, reuse, and so on. The topics under investigation continue to evolve to meet new challenges; communities emerge to address these challenges, often in interdisciplinary directions.

One of these interdisciplinary research communities that has emerged spans software engineering and games. This community addresses a broad range of issues that prevail in developing games for entertainment, serious games, and gamified applications. In this book, the focus is on the latter two. Serious games are also known as games with a purpose. Beyond their entertainment value, they also fulfill a purpose such as educating or training users on specific learning objectives. In contrast, gamified systems are non-entertainment applications that are enhanced with game elements to help motivate and engage users to improve their productivity, satisfaction, time on tasks, and so on. Research that is focused on serious games and gamified applications are grouped in this chapter under the term *gameful engineering* to reflect their serious contexts. Although distinct research topics, serious games and gamification share a core quality of service attribute: user experience. These applications possess the inherent, interdisciplinary complexity of creating user experiences that engage and motivate users to accomplish specific goals.

In this book, chapters on the design, maintenance, adaptation, evaluation, and experiences with gameful systems are presented. The structure of this chapter is as follows. Section 1.2 briefly presents background material on gameful systems. The section begins by distinguishing between serious games and gamification. This is followed by a description of their inherent development complexity due to their interdisciplinary nature and the use of software engineering perspectives to help address the complexity. Section 1.3 presents examples of recent advances (2016–2022) in the literature at the intersection of software engineering and gameful systems; the results are organized around established software engineering research topics. The overall structure and content of the book is described in Sect. 1.4; this section includes brief summaries of the core chapters. A summary is presented in Sect. 1.5.

1.2 Background: Gameful Engineering

1.2.1 Distinguishing Serious Games and Gamified Systems

The field of **Serious games** seeks to integrate the entertainment value of games with a purpose such as educating, training, exploring, or enhancing skills and competencies. This genre of games is well established. From a research perspective, an extensive body of literature is available that spans over five decades; the first results are reported by Abt [2]. The issues around creating desirable user experiences permeate the literature. The results include proposals on a wide variety of frameworks, methods, models, platforms, theoretical foundations, and so on; in addition, reports on the design, implementation, and evaluation of specific serious games are available. These games are often focused on educational or training environments. For example, in K–20, young players can learn fundamental skills in reading and mathematics with simpler games. Business students can play games to learn about finance and risk management topics. Computer science and engineering students can learn programming and software engineering skills (e.g., architecture, design, requirements engineering, project management, software quality assurance, testing). Medical students can learn anatomy with augmented or virtual reality applications. In training environments, serious games are available in numerous domains such as fire safety, healthcare, infrastructure inspection, manufacturing, software engineering, and so on.

Gamification has recently emerged as a field to guide the effective integration of game elements with non-entertainment applications to improve engagement, outcomes, productivity, and satisfaction within organizations (e.g., business enterprises, education, government). The first use of the term is accredited in the grey literature to Pelling in 2002 [74]. However, foundational work that proposes definitions for the gamification of software applications appears in the literature almost a decade later (e.g., [32]). From a research perspective, a robust body of literature is available that spans just over a decade. The issues around creating desirable user experiences permeate the literature; it includes proposals on a wide variety of frameworks, methods, models, platforms, theoretical foundations, and so on. In addition, reports on the design, implementation, and evaluation of specific gamified system are available. In business environments, gamification efforts often center on improving the engagement, productivity, and satisfaction of employees by enhancing core business applications or training activities. In educational environments, learning management platforms and course content are gamified to improve the students' engagement and motivation in order to improve learning outcomes and satisfaction.

1.2.2 Inherent Complexity

Gameful engineering relies on knowledge from multiple disciplines, which incurs a high level of complexity: arts, behavioral sciences, business, education, engineering, humanities, physical sciences and mathematics, and specific targeted domains such as healthcare (Fig. 1.1). The more creative aspects of gameful development rely heavily on the arts, humanities, and behavioral sciences in order to emotionally affect the users. In the realm of arts, expertise in digital media including asset modeling in 2D/3D, as well as proficiency in performing arts, music, and visual arts particularly in scene composition, is explored. Additionally, in the field of humanities, philosophical inquiries delve into ethical concerns such as those related to game addiction. In addition, knowledge of literature (e.g., narrative, plot, setting) may be used. From the behavioral sciences, contributions from the disciplines of anthropology, communication, sociology, and psychology may be adopted. Anthropology supports game development by considering the social and cultural norms of target audiences. Communication informs how to effectively share information across diverse forms of media. Sociology supports interactive multiplayer game development for communities of players. Psychology addresses the users' gameplay experience: engagement, motivation, retention, and reward systems.

The more technical development aspects of gameful applications rely on contributions from the physical sciences (e.g., computer science, physics), mathematics, and engineering (e.g., software and systems) disciplines. Within computer science, for example, knowledge in artificial intelligence, data analytics, graphics, human-computer interactions, kinds of systems (e.g., cloud, embedded, mobile, real-time, Web based), programming languages, and visualizations may be needed. From software engineering, knowledge in established engineering activities (e.g.,

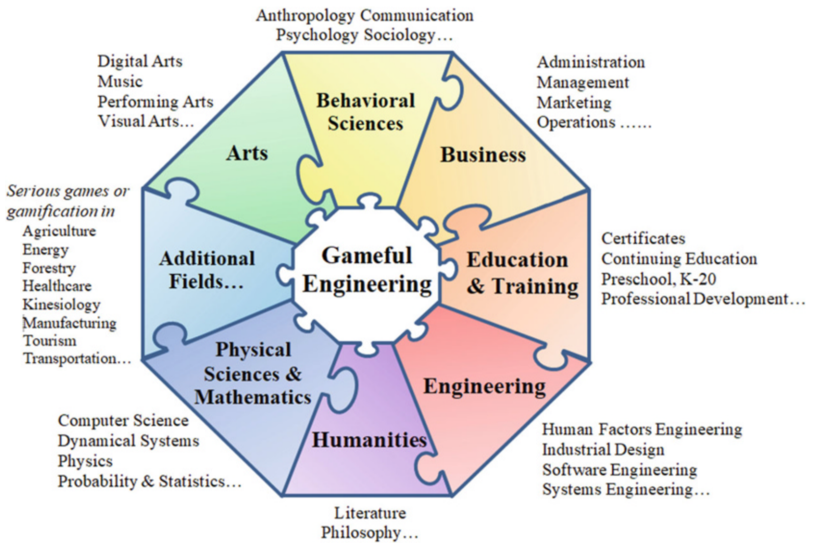


Fig. 1.1 The interdisciplinary nature of gameful engineering

requirements engineering, architecture and design, construction, testing), umbrella activities (configuration management, lifecycle processes, project management, traceability), re-use (components, patterns, product lines), metrics, and model-based transformations (code, test cases) may be valuable. However, knowledge in the software engineering community needs to be tailored for games in serious contexts, in particular with respect to achieving the necessary user experience.

1.2.3 Addressing the Complexity

To help address the complexity of creating gameful systems, an active interdisciplinary community spanning software and gameful engineering researchers has emerged. Numerous topics receive attention such as:

- Engineering activities tailored for serious games or gamification
  - requirements engineering
  - architecture
  - design
  - testing
- Umbrella activities tailored for serious games or gamification
  - lifecycle process models
  - project management
  - software quality assurance
  - traceability
- Established topics tailored for serious games or gamification
  - frameworks
  - metrics
  - model-based or model-driven engineering
  - platforms and tools
  - reuse

These topics are not always considered independently in the community. For example, research questions may investigate a model-based design framework for gamification or a metrics-based testing platform for serious games. In addition, the research may draw upon other disciplines (e.g., computer science topics artificial intelligence, computer-human interfaces, data analytics, visualization) to propose innovative solutions.

The community has established dedicated workshops (e.g., Games and Software Engineering [14, 15, 17, 24, 26, 92]) and journals (e.g., IEEE Transactions on Games [49]). Edited books on computer games and software engineering that provide snapshots of the state-of-the research are available [23, 25]; these span software engineering for games and serious educational games for software engineering education. Edited books on gamification topics are also available [83, 91].

1.3 Recent Advances at the Intersection of Software and Gameful Engineering

Examples of recently published peer-reviewed results (2016–2022) that span software and gameful engineering topics are presented in this section. A set of high-level topics is explored that includes traditional engineering activities (requirements engineering, architecture and design, testing), established topics (frameworks and platforms, metrics, models, reuse), and umbrella activities (lifecycle processes, project management, software quality assurance, traceability). The sources for the article searches are ACM Digital Library, IEEE Xplore, SCOPUS, and Google Scholar. The searches for articles focused on serious games are structured as follows:

- “requirements engineering” “serious games” “software engineering”
- architecture “serious games” “software engineering”
- design “serious games” “software engineering”
- testing “serious games” “software engineering”
- “lifecycle process” “serious games” “software engineering”
- “project management” “serious games” “software engineering”
- “software quality” “serious games” “software engineering”
- traceability “serious games” “software engineering”
- frameworks “serious games” “software engineering”
- platforms “serious games” “software engineering”
- metrics “serious games” “software engineering”
- model “model based” “model driven” “serious games” “software engineering”
- reuse “serious games” “software engineering”

Searches with a parallel structure for gamified systems are also run, substituting “serious games” with the terms gamification and gamified.

The results are presented in Table 1.1. The table is organized into four main parts: software engineering for serious games, serious games for software engineering, software engineering for gamification, and gamification for software engineering. The results are at various levels of maturity; they appear in workshops, conferences, and journals.

Overall, the summary reveals that serious-games-related topics have received more attention in recent years than the gamification topics (53 vs. 30). This may be due to the more established nature of the serious game community in comparison to the gamification community. In addition, there are gaps in the recent literature that the community may be interested in exploring.

Table 1.1 Recent results in software and gameful engineering

Software engineering for serious games	
Requirements engineering	[28, 61]
Architecture and design	[43, 62]
Testing	[4, 64]
Frameworks and platforms	[1, 44, 47, 80, 93]
Metrics	[3, 52, 57, 85]
Models	[12, 31, 45, 58, 86, 94]
Reuse	[7, 60, 75, 82, 88]
Lifecycle processes	[16, 55]
Project management	[21]
Software quality assurance	[89]
Traceability	–
Serious games for software engineering	
Requirements engineering	[30, 36, 40, 53]
Architecture and design	[13, 67]
Testing	[38, 78, 79, 87]
Frameworks and platforms	–
Metrics	–
Models	[76]
Reuse	–
Lifecycle processes	[5, 9, 59]
Project management	[19, 54, 65]
Software quality assurance	[6, 11, 22, 34, 46, 63]
Traceability	–
Software engineering for gamification	
Requirements engineering	[51]
Architecture and design	[42, 68, 73]
Testing	[37]
Frameworks and platforms	[50]
Metrics	–
Models	[8, 20, 27, 71, 90]
Reuse	–
Lifecycle processes	–
Project management	–
Software quality assurance	–
Traceability	[35, 56, 72]

(continued)

Table 1.1 (continue)

Gamification for software engineering	
Requirements engineering	[29]
Architecture and design	[73]
Testing	[38, 87]
Frameworks and platforms	[18, 33, 39, 66, 84]
Metrics	–
Models	–
Reuse	–
Lifecycle processes	[10, 70, 77, 81]
Project management	[69]
Software quality assurance	[41, 48]
Traceability	–

The search results are intended to provide a preliminary summary of recent results; however, additional peer-reviewed results for the topics selected may be available in alternative sources or identified with a snowballing approach to explore recently cited work. The search may also be expanded to include interesting topics covering software engineering, serious games and gamification, and computer science specializations such as artificial intelligence, data analytics, visualization, and human-computer interfaces.

1.4 Content of the Book

This edited book consists of 13 chapters that have been prepared by 45 authors from 9 countries (Brazil, Canada, Germany, Greece, Italy, Mauritius, Mexico, the United Kingdom, the United States). There are 11 core chapters that are organized into 2 main parts. Part I considers topics on the design, maintenance, adaptation, and evaluation of gameful systems. Part II considers topics on experiences with gameful systems.

1.4.1 Part I Topics on the Design, Maintenance, Adaptation, and Evolution of Gameful Systems

A range of methods that can be used to evaluate users' experiences in serious games during and after the development process are presented by Sotiris Kirginas in **Chap.** 2, "User Experience Evaluation Methods for Games in Serious Contexts." The methods include qualitative, quantitative, subjective, objective, short term, long term, summative, and formative. The chapter also provides insights into when it is most beneficial to apply the different user experience assessment methods in the development cycle.

A novel content-agnostic serious educational game engineering framework is proposed by Vipin Verma et al. in **Chap.** 3, "Software Engineering for Dynamic Game Adaptation." The framework can be used to create games by reusing existing educational game mechanics. The framework is used to separate the game into three independent components: mechanics, content, and student modeling. In addition, stealth assessment can be integrated into the student model.

A technique to design and analyze the architecture of serious mobile games is proposed by Leticia Davila-Nicanor et al. in **Chap.** 4, "Performance on Software Architecture Design to Serious Games for Mobile Devices." The advantages of an early architecture evaluation include reducing the costs to remove defects and improve the software performance. The technique utilizes established design patterns (Wrapper, Singleton, MVC) to improve the distribution of memory and execution time resources. The architecture analysis is achieved with graph theory and software metrics. A dispersion diagram visualizes an architecture with acceptable quality levels.

A serious game-based online platform in the healthcare education domain is proposed by Edward Melcer et al. in **Chap.** 5, "ENTRUST: Co-design and Validation of a Serious Game for Assessing Clinical Decision-Making and Readiness for Entrustment." The purpose of the platform is to assess trainees' decision-making competencies across various entrustable professional activity domains. The design, insights identified and lessons learned throughout the development process, and results from a pilot study of the platform are presented. The pilot study demonstrates the tool's capability to discriminate between levels of surgical expertise and provides initial evidence for its use as an objective assessment approach for clinical decision-making.

A generalized model for evolving existing serious games to utilize machine learning approaches without the need to rebuild the game from scratch is proposed by Michael Miljanovic et al. in **Chap.** 6, "Engineering Adaptive Serious Games Using Machine Learning." An approach to engineer machine learning-based serious games is presented; in addition, five common challenges and possible solutions are discussed. The challenges include selecting data for the machine learning input, choosing game elements to adapt, addressing the cold start problem, determining the frequency of adaptation, and testing that an adaptive game benefits from machine learning.

1.4.2 Part II Topics on Experiences with Gameful Systems

The potential to transfer strategies commonly utilized in entertainment games to serious games are presented by Vanissa Wanick et al. in **Chap.** 7, "Future Directions in Games for Serious Contexts: A Conversation About Transferability." Three complementary perspectives regarding emerging aspects of player agency, serious game modification, and transferability across different contexts are discussed. More specifically, these perspectives include emergent possibilities, modding, and the importance of AI emotion modeling to inform better game design.

A serious game for programming is presented by Leckraj Nagowah et al. in **Chap.** 8, "Code-Venture: A Mobile Serious Game for Introductory Programming." This serious game helps players understand the basics of coding and improve their programming skills. The learning objectives of the game are based on the programming principles in the ACM/IEEE Computer Science Curricula 2013. A teacher's application can monitor, assess, evaluate, and store the player's performance data. To evaluate the game, pre- and post-game surveys indicate the game is useful and engaging. The results reveal an increase in the number of students that (1) are interested in pursuing a programming career and (2) find programming easy.

Approaches to revising two traditional, lecture-based game design classes to make use of flipped classroom models are presented by Bruce Maxim et al. in **Chap.** 9, "Using Active Learning to Teach Software Engineering in Game Design Courses." The revised courses use agile software engineering practices to design, implement, and test game prototypes; they rely on active learning, role-play, and gamification. Surveys are used to measure the perceived levels of engagement with course activities. The results indicate that students attending flipped classes are slightly more engaged with the course materials than those taking the class offered using lecture only. In addition, students interacting with the active learning course materials felt better able to apply their knowledge than students in a traditional lecture course.

A gamified framework for manual exploratory GUI testing of Web-based applications is presented by Riccardo Coppola et al. in **Chap.** 10, "A Framework for the Gamification of GUI Testing." The framework is implemented, and a preliminary evaluation is conducted to assess efficiency, effectiveness, and user experience. The results indicate that the gamified framework helps to obtain test suites with higher coverage; in addition, the number of bugs identified while traversing the applications under test is slightly reduced. The participants indicate the gamified framework provides a positive user experience, and the majority of participants expressed their willingness to use such instruments again.

A field experiment involving the gamified quality improvement of five student projects in an information systems course is presented by Mathias Eggert et al. in **Chap.** 11, "Applying Leaderboards for Quality Improvement in Software Development Projects." The project members are interviewed to capture the impact of using a leaderboard in terms of changing their programming behavior. The interviews are based on 11 main questions that are open. The results reveal a motivational effect with respect to improving code quality is achieved; in addition, a reduction of code smells is reported.

The design and implementation of a serious game for cybersecurity education, A NERD DOGMA, is presented by Gabriele Costa et al. in **Chap.** 12, "Designing a Serious Game for Cybersecurity Education." It is an escape room adventure game, in which the challenges are based on real security scenarios. The player needs to exfiltrate data, break ciphers, and intrude in remote systems in order to acquire first-hand experience in the planning and execution of security operations. Key design decisions for the A NERD DOGMA game include approaches to introduce security tools, integrate third-party technologies, and how to mimic external environments.

1.5 Summary

This chapter introduces a collection of research results that provide software engineering perspectives on games for serious contexts. It begins by presenting background material on gameful systems (serious games and gamified applications), the complexity of gameful systems due to their interdisciplinary nature, and the potential to address their complexity with tailored software engineering approaches. This is followed by a snapshot of recently published peer-reviewed articles that are focused on interdisciplinary work spanning gameful and software engineering. Next, the overall structure of the book and brief summaries of the core chapters are presented.

References

1. Abdellatif, A.J., McCollum, B., McMullan, P.: Serious games: quality characteristics evaluation framework and case study. In: 2018 IEEE Integrated STEM Education Conference (ISEC), pp. 112–119. IEEE, Piscataway (2018)
2. Abt, C.C.: Serious games. Am. Behav. Sci. **14**(1), 129–129 (1970)
3. Alonso-Fernandez, C., Calvo, A., Freire, M., Martinez-Ortiz, I., Fernandez-Manjon, B.: Systematizing game learning analytics for serious games. In: 2017 IEEE Global Engineering Education Conference (EDUCON), pp. 1111–1118. IEEE, Piscataway (2017)
4. Alonso-Fernández, C., Perez-Colado, I.J., Calvo-Morata, A., Freire, M., Ortiz, I.M., Manjon, B.F.: Applications of simva to simplify serious games validation and deployment. IEEE Rev. Iberoam. de Tecnol. del Aprendiz. **15**(3), 161–170 (2020)
5. Ammons, B., Bansal, S.K.: Scrumify: A software game to introduce agile software development methods. J. Eng. Educ. Transform. **30**(Special Issue) (2017)
6. Ardiç, B., Yurdakul, I., Tüzün, E.: Creation of a serious game for teaching code review: an experience report. In: 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T), pp. 1–5. IEEE, Piscataway (2020)
7. Argasiski, J.K., Wegrzyn, P.: Affective patterns in serious games. Fut. Gen. Comput. Syst. **92**, 526–538 (2019)
8. Ašeriškis, D., Blažauskas, T., Damaševičius, R.: UAREI: a model for formal description and visual representation/software gamification. DYNA **84**(200), 326–334 (2017)
9. Aydan, U., Yilmaz, M., Clarke, P.M., O'Connor, R.V.: Teaching ISO/IEC 12207 software lifecycle processes: a serious game approach. Comput. Stand. Inter. **54**, 129–138 (2017)
10. Ayoup, P., Costa, D.E., Shihab, E.: Achievement Unlocked: A Case Study on Gamifying Devops Practices in Industry. Association for Computing Machinery, New York (2022)
11. Baars, S., Meester, S.: Codearena: inspecting and improving code quality metrics using minecraft. In: 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pp. 68–70. IEEE, Piscataway (2019)

12. Barajas, A., Álvarez, F., Muñoz, J., Santaolaya, R., Collazos, C., Hurtado, J.: Verification and validation model for short serious game production. IEEE Lat. Am. Trans. **14**(4), 2007–2012 (2016)
13. Bartel, A., Hagel, G.: Gamifying the learning of design patterns in software engineering education. In: 2016 IEEE Global Engineering Education Conference (EDUCON), pp. 74–79. IEEE, Piscataway (2016)
14. Bell, J., Cooper, K.M.L., Kaiser, G.E., Swapneel, S.: Welcome to the 2nd international games and software engineering workshop (gas 2012). In: 2012 Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS), pp. iii–iv (2012)
15. Bishop, J., Cooper, K.M.L., Scacchi, W., Whitehead, J.: Introduction to the 4th international workshop on games and software engineering (gas 2015). In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, pp. 979–980 (2015)
16. Braad, E., Zavcer, G., Sandovar, A.: Processes and models for serious game design and development. In: Entertainment Computing and Serious Games: International GI-Dagstuhl Seminar 15283, Dagstuhl Castle, July 5–10, 2015, Revised Selected Papers, pp. 92–118. Springer, Berlin (2016)
17. Bucchiarone, A., Cooper, K.M., Lin, D., Melcer, E.F., Sung, K.: Games and software engineering: engineering fun, inspiration, and motivation. ACM SIGSOFT Softw. Eng. Notes **48**(1), 85–89 (2023)
18. Cacciotto, F., Fulcini, T., Coppola, R., Ardito, L.: A metric framework for the gamification of web and mobile GUI testing. In: 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 126–129 (2021)
19. Calderon, A., Ruiz, M., O'Connor, R.V.: Prodecadmin: a game scenario design tool for software project management training. In: Proceedings of the Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, September 6–8, 2017, pp. 241–248. Springer, Berlin (2017)
20. Calderón, A., Boubeta-Puig, J., Ruiz, M.: MEdit4CEP-Gam: a model-driven approach for user-friendly gamification design, monitoring and code generation in CEP-based systems. Inform. Softw. Tech. **95**, 238–264 (2018)
21. Calderon, A., Trinidad, M., Ruiz, M., O'Connor, R.V.: Towards a standard to describe and classify serious games as learning resources for software project management. In: Proceedings of the Systems, Software and Services Process Improvement: 25th European Conference, EuroSPI 2018, Bilbao, September 5–7, 2018, pp. 229–239. Springer, Berlin (2018)
22. Clegg, B.S., Rojas, J.M., Fraser, G.: Teaching software testing concepts using a mutation testing game. In: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET), pp. 33–36. IEEE, Piscataway (2017)
23. Cooper, K.: Software Engineering Perspectives on Computer Game Development. CRC Press, Taylor & Francis, Boca Raton (2021)
24. Cooper, K.M.L.: Message from the chair of the 5th international workshop on games and software engineering. In: GAS '16: Proceedings of the 5th International Workshop on Games and Software Engineering. Association for Computing Machinery, New York, NY (2016)
25. Cooper, K., Scacchi, W.: Computer Games and Software Engineering. CRC Press, Taylor & Francis, Boca Raton (2015)
26. Cooper, K.M.L., Scacchi, W., Wang, A.I.: Welcome to the 3rd international workshop on games and software engineering: engineering computer games to enable positive, progressive change (gas 2013). In: 2013 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change (GAS), pp. iii–iii (2013)
27. Cosentino, V., Gérard, S., Cabot, J.: A model-based approach to gamify the learning of modeling. In: Proceedings of the 5th Symposium on Conceptual Modeling Education and the 2nd International iStar Teaching Workshop Co-located with the 36th International Conference on Conceptual Modeling (ER 2017), Valencia, November 6–9, 2017, pp. 15–24 (2017)

28. Dalpiaz, F., Cooper, K.M.: Games for requirements engineers: analysis and directions. IEEE Softw. **37**(1), 50–59 (2020)
29. Dar, H.S.: Reducing ambiguity in requirements elicitation via gamification. In: 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 440–444 (2020)
30. Delen, M., Dalpiaz, F., Cooper, K.: Bakere: a serious educational game on the specification and analysis of user stories. In: 2019 IEEE 27th International Requirements Engineering Conference (RE), pp. 369–374. IEEE, Piscataway (2019)
31. De Lope, R.P., Medina-Medina, N., Urbieta, M., Lliteras, A.B., García, A.M.: A novel UML-based methodology for modeling adventure-based educational games. Entertain. Comput. **38**, 100429 (2021)
32. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: defining "gamification". Association for Computing Machinery, New York (2011)
33. Dichev, C., Dicheva, D., Irwin, K.: Gamification driven learning analytics. In: Proceedings of the 13th International Conference on e-Learning, pp. 70–76 (2018)
34. dos Santos, H.M., Durelli, V.H., Souza, M., Figueiredo, E., da Silva, L.T., Durelli, R.S.: Cleangame: gamifying the identification of code smells. In: Proceedings of the XXXIII Brazilian Symposium on Software Engineering, pp. 437–446 (2019)
35. Ebert, C., Vizcaino, A., Grande, R.: Unlock the business value of gamification. IEEE Softw. **39**(6), 15–22 (2022)
36. Espinha Gasiba, T., Beckers, K., Suppan, S., Rezabek, F.: On the requirements for serious games geared towards software developers in the industry. In: 2019 IEEE 27th International Requirements Engineering Conference (RE), pp. 286–296 (2019)
37. Fraser, G.: Gamification of software testing. In: 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST), pp. 2–7. IEEE, Piscataway (2017)
38. Fulcini, T., Ardito, L.: Gamified exploratory GUI testing of web applications: a preliminary evaluation. In: 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 215–222 (2022)
39. Garcia, F., Pedreira, O., Piattini, M., Cerdeira-Pena, A., Penabad, M.: A framework for gamification in software engineering. J. Syst. Softw. **132**, 21–40 (2017)
40. Garcia, I., Pacheco, C., Leon, A., Calvo-Manzano, J.A.: A serious game for teaching the fundamentals of ISO/IEC/IEEE 29148 systems and software engineering–lifecycle processes–requirements engineering at undergraduate level. Comput. Stand. Interf. **67**, 103377 (2020)
41. Gasca-Hurtado, G.P., Gómez-Alvarez, M.C., Muñoz, M., Mejía, J.: Gamification proposal for defect tracking in software development process. In: Proceedings of the Systems, Software and Services Process Improvement: 23rd European Conference, EuroSPI 2016, Graz, September 14–16, 2016, pp. 212–224. Springer, Berlin (2016)
42. Gasca-Hurtado, G.P., Gómez-Álvarez, M.C., Machuca-Villegas, L., Muñoz, M.: Design of a gamification strategy to intervene in social and human factors associated with software process improvement change resistance. IET Softw. **15**(6), 428–442 (2021)
43. Goli, A., Teymournia, F., Naemabadi, M., Garmaroodi, A.A.: Architectural design game: a serious game approach to promote teaching and learning using multimodal interfaces. Educ. Inform. Technol. **27**(8), 11467–11498 (2022)
44. Haendler, T., Neumann, G.: A framework for the assessment and training of software refactoring competences. In: Proceedings of the 11th International Conference on Knowledge Management and Information Systems, pp. 307–316 (2019)
45. Haendler, T., Neumann, G.: Ontology-based analysis of game designs for software refactoring. In: Proceedings of the 11th International Conference on Computer Supported Education (CSEDU 2019) (1), pp. 24–35 (2019)
46. Haendler, T., Neumann, G.: Serious refactoring games. In: Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS-52), pp. 7691–7700. IEEE, Piscataway (2019)
47. Hamiye, F., Said, B., Serhan, B.: A framework for the development of serious games for assessment. In: Proceedings of the Games and Learning Alliance: 8th International Conference, GALA 2019, Athens, November 27–29, 2019, pp. 407–416. Springer, Berlin (2019)

48. Herranz, E., Guzmán, J.G., de Amescua-Seco, A., Larrucea, X.: Gamification for software process improvement: a practical approach. IET Softw. **13**(2), 112–121 (2019)
49. IEEE transactions on games. https://transactions.games/. Accessed 16 Feb 2023
50. Klock, A.C.T., Gasparini, I., Pimenta, M.S.: 5w2h framework: a guide to design, develop and evaluate the user-centered gamification. In: Proceedings of the 15th Brazilian Symposium on Human Factors in Computing Systems, pp. 1–10 (2016)
51. Kumar, B.S., Krishnamurthi, I.: Improving user participation in requirement elicitation and analysis by applying gamification using architect's use case diagram. In: Proceedings of the 3rd International Symposium on Big Data and Cloud Computing Challenges (ISBCC–16'), pp. 471–482. Springer, Berlin (2016)
52. Loh, C.S., Li, I.H., Sheng, Y.: Comparison of similarity measures to differentiate players' actions and decision-making profiles in serious games analytics. Comput. Hum. Behav. **64**(C), 562–574 (2016)
53. Marcelino-Jesus, E., Sarraipa, J., Agostinho, C., Jardim-Goncalves, R.: The use of serious games in requirements engineering. In: Enterprise Interoperability VII: Enterprise Interoperability in the Digitized and Networked Factory of the Future, pp. 263–274. Springer, Berlin (2016)
54. Marín, B., Vera, M., Giachetti, G.: An adventure serious game for teaching effort estimation in software engineering. In: IWSM-Mensura, pp. 71–86 (2019)
55. Marín-Vega, H., Alor-Hernández, G., Colombo-Mendoza, L.O., Bustos-López, M., Zataraín-Cabada, R.: Zeusar: a process and an architecture to automate the development of augmented reality serious games. Multimed. Tools Appl. **81**(2), 2901–2935 (2022)
56. Maro, S., Sundklev, E., Persson, C.O., Liebel, G., Steghöfer, J.P.: Impact of gamification on trace link vetting: a controlled experiment. In: Proceedings of the Requirements Engineering: Foundation for Software Quality: 25th International Working Conference, REFSQ 2019, Essen, March 18–21, 2019, pp. 90–105. Springer, Berlin (2019)
57. Mäses, S., Hallaq, B., Maennel, O.: Obtaining better metrics for complex serious games within virtualised simulation environments. In: European Conference on Games Based Learning, pp. 428–434. Academic Conferences International Limited, Reading (2017)
58. Matallaoui, A., Herzig, P., Zarnekow, R.: Model-driven serious game development integration of the gamification modeling language GaML with unity. In: 2015 48th Hawaii International Conference on System Sciences, pp. 643–651 (2015)
59. Maxim, B.R., Kaur, R., Apzynski, C., Edwards, D., Evans, E.: An agile software engineering process improvement game. In: 2016 IEEE Frontiers in Education Conference (FIE), pp. 1–4. IEEE, Piscataway (2016)
60. Meftah, C., Retbi, A., Bennani, S., Idrissi, M.K.: Mobile serious game design using user experience: modeling of software product line variability. Int. J. Emerg. Technol. Learn. (Online) **14**(23), 55 (2019)
61. Mejbri, Y., Khemaja, M., Raies, K.: Requirements engineering for pervasive games based smart learning systems. In: Innovations in Smart Learning, pp. 129–138. Springer, Berlin (2017)
62. Mestadi, W., Nafil, K., Touahni, R., Messoussi, R.: An assessment of serious games technology: toward an architecture for serious games design. Int. J. Comput. Games Technol. **2018** (2018)
63. Miljanovic, M.A., Bradbury, J.S.: Robobug: a serious game for learning debugging techniques. In: Proceedings of the 2017 ACM Conference on International Computing Education Research, pp. 93–100 (2017)
64. Moizer, J., Lean, J., Dell'Aquila, E., Walsh, P., Keary, A.A., O'Byrne, D., Di Ferdinando, A., Miglino, O., Friedrich, R., Asperges, R., Sica, L.S.: An approach to evaluating the user experience of serious games. Comput. Educ. **136**, 141–151 (2019)
65. Molléri, J.S., Gonzalez-Huerta, J., Henningsson, K.: A legacy game for project management in software engineering courses. In: Proceedings of the 3rd European Conference of Software Engineering Education, pp. 72–76 (2018)

66. Monteiro, R.H.B., Oliveira, S.R.B., Souza, M.R.D.A.: A standard framework for gamification evaluation in education and training of software engineering: an evaluation from a proof of concept. In: 2021 IEEE Frontiers in Education Conference (FIE), pp. 1–7. IEEE, Piscataway (2021)
67. Montenegro, C.H., Astudillo, H., Álvarez, M.C.G.: ATAM-RPG: a role-playing game to teach architecture trade-off analysis method (ATAM). In: 2017 XLIII Latin American Computer Conference (CLEI), pp. 1–9. IEEE, Piscataway (2017)
68. Morschheuser, B., Hassan, L., Werder, K., Hamari, J.: How to design gamification? A method for engineering gamified software. Inform. Softw. Technol. **95**, 219–237 (2018)
69. Muñoz, M., Pérez Negrón, A.P., Mejia, J., Gasca-Hurtado, G.P., Gómez-Alvarez, M.C., Hernández, L.: Applying gamification elements to build teams for software development. IET Softw. **13**(2), 99–105 (2019)
70. Neto, P.S., Medeiros, D.B., Ibiapina, I., da Costa Castro, O.C.: Case study of the introduction of game design techniques in software development. IET Softw. **13**(2), 129–143 (2019)
71. Oberhauser, R.: An ontological perspective on the digital gamification of software engineering concepts. Int. J. Adv. Softw. **9**(3 and 4), 207–221 (2016)
72. Parizi, R.M.: On the gamification of human-centric traceability tasks in software testing and coding. In: 2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA), pp. 193–200. IEEE, Piscataway (2016)
73. Pedreira, O., García, F., Piattini, M., Cortiñas, A., Cerdeira-Pena, A.: An architecture for software engineering gamification. Tsinghua Sci. Technol. **25**(6), 776–797 (2020)
74. Pelling, N.: (2015). https://nanodome.wordpress.com/2011/08/09/the-short-prehistory-of-gamification/
75. Perez-Medina, J.L., Jimenes-Vargas, K.B., Leconte, L., Villarreal, S., Rybarczyk, Y., Vanderdonckt, J.: ePHoRt: towards a reference architecture for tele-rehabilitation systems. IEEE Access **7**, 97159–97176 (2019)
76. Prasetya, W., Leek, C., Melkonian, O., ten Tusscher, J., van Bergen, J., Everink, J., van der Klis, T., Meijerink, R., Oosenbrug, R., Oostveen, J., et al.: Having fun in learning formal specifications. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), pp. 192–196. IEEE, Piscataway (2019)
77. Ren, W., Barrett, S., Das, S.: Toward gamification to software engineering and contribution of software engineer. In: Proceedings of the 2020 4th International Conference on Management Engineering, Software Engineering and Service Sciences. Association for Computing Machinery, New York (2020)
78. Rojas, J.M., Fraser, G.: Teaching mutation testing using gamification. In: European Conference on Software Engineering Education (ECSEE) (2016)
79. Sherif, E., Liu, A., Nguyen, B., Lerner, S., Griswold, W.G.: Gamification to aid the learning of test coverage concepts. In: 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T), pp. 1–5. IEEE, Piscataway (2020)
80. Shi, W., Kaneko, K., Ma, C., Okada, Y.: A framework for automatically generating quiz-type serious games based on linked data. Int. J. Inform. Educ. Technol. **9**(4), 250–256 (2019)
81. Sisomboon, W., Phakdee, N., Denwattana, N.: Engaging and motivating developers by adopting scrum utilizing gamification. In: 2019 4th International Conference on Information Technology (InCIT), pp. 223–227 (2019)
82. Söbke, H., Streicher, A.: Serious games architectures and engines. In: Entertainment Computing and Serious Games: International GI-Dagstuhl Seminar 15283, Dagstuhl Castle, July 5–10, 2015, Revised Selected Papers, pp. 148–173. Springer, Berlin (2016)
83. Stieglitz, S., Lattemann, C., Robra-Bissantz, S., Zarnekow, R., Brockmann, T. (eds.): Gamification: using Game Elements in Serious Contexts. Springer, Berlin (2016)
84. Stol, K.J., Schaarschmidt, M., Goldblit, S.: Gamification in software engineering: the mediating role of developer engagement and job satisfaction. Emp. Softw. Eng. **27**(2), 35 (2022)

85. Suryapranata, L.K.P., Soewito, B., Kusuma, G.P., Gaol, F.L., Warnars, H.L.H.S.: Quality measurement for serious games. In: 2017 International Conference on Applied Computer and Communication Technologies (ComCom), pp. 1–4. IEEE, Piscataway (2017)
86. Toda, A.M., Oliveira, W., Klock, A.C., Palomino, P.T., Pimenta, M., Gasparini, I., Shi, L., Bittencourt, I., Isotani, S., Cristea, A.I.: A taxonomy of game elements for gamification in educational contexts: proposal and evaluation. In: 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT), vol. 2161, pp. 84–88. IEEE, Piscataway (2019)
87. Valle, P.H.D., Vilela, R.F., Hernandes, E.C.M.: Does gamification improve the training of software testers? A preliminary study from the industry perspective. Association for Computing Machinery, New York (2021)
88. van der Vegt, W., Nyamsuren, E., Westera, W.: Rage reusable game software components and their integration into serious game engines. In: Proceedings of the Software Reuse: Bridging with Social-Awareness: 15th International Conference, ICSR 2016, Limassol, June 5–7, 2016, pp. 165–180. Springer, Berlin (2016)
89. van der Vegt, W., Westera, W.: Quality of reusable game software: empowering developers with automated quality checks. In: 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), pp. 446–452. IEEE, Piscataway (2019)
90. Vapiwala, F., Pandita, D.: A decision model for using gamification technology in employee training. In: 2022 International Conference on Decision Aid Sciences and Applications (DASA), pp. 942–946. IEEE, Piscataway (2022)
91. Vesa, M. (ed.): Organizational Gamification: Theories and Practices of Ludified Work in Late Modernity (1st ed.). Routledge, Abingdon (2021)
92. Whitehead, J., Lewis, C.: Abstract for the proceedings of the 1st international workshop on games and software engineering. In: GAS '11: Proceedings of the 1st International Workshop on Games and Software Engineering, pp. 1194–1195. Association for Computing Machinery, New York (2011)
93. Wilson, D.W., Jenkins, J., Twyman, N., Jensen, M., Valacich, J., Dunbar, N., Wilson, S., Miller, C., Adame, B., Lee, Y.H., et al.: Serious games: an evaluation framework and case study. In: 2016 49th Hawaii International Conference on System Sciences (HICSS), pp. 638–647. IEEE, Piscataway (2016)
94. Zahari, A.S., Ab Rahim, L., Nurhadi, N.A., Aslam, M.: A domain-specific modelling language for adventure educational games and flow theory. Int. J. Adv. Sci. Eng. Inform. Technol. **10**(06) (2020)

Part I
Topics on the Design, Maintenance, Adaptation, and Evaluation of Gameful Systems

Chapter 2
User Experience Evaluation Methods for Games in Serious Contexts

Sotiris Kirginas

Abstract User experience in digital games can be influenced by many factors such as flow [Csikszentmihalyi (Flow: the psychology of optimal experience. Harper Collins, 1990), Sweetser and Wyeth (Computers in Entertainment 3(3):1–24, 2005)], immersion [Brown and Cairns (ACM Conference on Human Factors in Computing Systems, CHI 2004, ACM Press, 2004), Ermi and Mayra (Proceedings of Chancing Views – Worlds in Play. Digital Games Research Association's Second International Conference, 2005)], frustration or tension [Gilleade and Dix (Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology – ACE '04, 2004)], psychological absorption [Funk et al. (Proceedings of the Second International Conference on Entertainment Computing Pittsburgh, Carnegie Mellon University, 2003)], and social game context [Bracken et al. (Online video games and gamers' sensations of spatial, social, and copresence. FuturePlay 2005, 2005)]. Most of these factors should be present in a digital game in order to provide the optimal gaming experience [Kirginas (Contemporary Educational Technology 14(2):ep351, 2022), Kirginas et al. (International Journal of Child-Computer Interaction 28, 2021), Kirginas and Gouscos (The International Journal of Serious Games 4:53–69, 2017; International Journal of Serious Games 3:29–45, 2016)]. As there are many different game genres, sub-genres, and game types, user experience needs to be explored in more detail in research studies. This need is even greater when we talk about serious games. User experience is a multifactorial concept that is difficult to measure. This chapter aims to present a range of quantitative and qualitative/objective and subjective/short-term and long-term/formative and summative methods that can be used to evaluate users' experience in serious games during and after the development process. It is also intended to provide insight into when the different user experience assessment methodologies should be employed in the development cycle.

S. Kirginas (✉)
National and Kapodistrian University of Athens, Athens, Greece
e-mail: skirginas@media.uoa.gr

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_2

Keywords Digital games · Serious games · Assessment · Experience · Quantitative · Qualitative

2.1 Introduction

Nowadays, digital games compete with traditional activities like reading books, watching movies, listening to music, surfing the Internet, or playing sports [1]. Digital games regularly attract billions of players online and offline, generating huge revenue. However, digital games also present new research challenges for many traditional and new scientific areas [1]. With recent advances in the field of human-computer interaction [2], new methods are available to precisely measure how people interact with entertainment technologies [3, 4]. With new evaluation methods of player interaction, we aim to support the traditional digital serious games development and improve game design process [5].

Game developers increasingly employ user testing with playability evaluation in the development of digital games [6–9]. Unlike other software, digital games often offer a unique experience that contains elements that are difficult to be evaluated. User experience in digital games can be influenced by many factors, such as flow [10, 11], immersion [12, 13], frustration or tension [14], psychological absorption [15], and social game context [16].

A component of a game design process is observation of players in response to mechanics. Since it is very time-consuming to gain such individual knowledge of game design, it is necessary to gain a more rapid understanding of the complex behavior of players in response to game mechanics. To gain a more complete view of user experience, several recent solutions have combined event logging with objective and subjective player feedback [3, 17]. Similarly, player behavior is modeled to find "optimal spots in the game and level design" [4].

This chapter aims (a) to present a range of quantitative and qualitative/objective and subjective/short-term and long-term/formative and summative methods that can be used to evaluate users' experience in digital serious games during and after the development process and (b) to provide insight into when the different user experience evaluation methodologies should be employed in the development cycle.

The structure of this chapter is as follows: First, we outline methods for evaluating the user experience of digital serious games based on the body of the literature. In the next section, we explain how users' experiences are measured in digital serious games. Last but not least, we discuss when, how, and why to use all main methodologies proposed to measure the effectiveness of games in serious contexts.

2.2 Defining User Experience

According to Almeida et al. [18], experience is both the process and the outcome of a user's engagement with the environment at a given moment. It is both an interactive (the process of playing the game) and an emotional (the consequence of playing) experience—a feeling (or combination of emotions) that occurs as a result of playing [19]. The interaction process is the way players interact when playing; it is how the player interacts with other playable and non-playable characters and objects in the game environment [20] and how they make decisions. The game limits this process, which is influenced by the players' background, motivations, expectations, and current emotional experiences, which can change during the game [19]. Almeida [20] argues that in many cases, the emotional state of the players also influences the interaction processes: If they are anxious, they may be less attentive, which could affect their ability to play and win, while if they are relaxed, they could be in a flow state according to Csikszentmihalyi [10]. This is still a fairly open field in the game industry, as horror games, a prominent video game genre, is dedicated to keeping players in flow through anxiety or fear.

This approach has an impact on the outcome of the game. If the emotional experience is positive, games can trigger positive emotions (e.g., satisfaction, happiness, and excitement); if the emotional experience is unpleasant, games can trigger negative emotions (anger, sadness, boredom). Positive or negative effects can influence the interaction process by changing players' motivations and engagement [19, 21, 22]. This bidirectional interaction can explain why players can sometimes experience both pleasure and frustration during the course of a game [20].

According to Roto [23], there are three phases of the game experience: (a) the expected game experience (before a player interacts with a game), (b) the game experience during interaction (experience that occurs while interacting with the game), and (c) the overall player experience (experience that occurs while interacting with the game) (experience after the game ends). The player experience during interaction is the most important of the three phases of player experience mentioned above. Examining the player experience during interaction is critical to improving a game, as this phase can identify features and components that provide a positive experience as well as those that do not. According to Lallemand [24], three factors should be considered in order to understand the game experience during the interaction phase: the human aspect (dispositions, expectations, needs, motivation, mood, etc.), the system aspect (e.g., complexity, purpose, usability, functionality, etc.), and the contextual aspect (or environment) in which the interaction takes place (e.g., organizational/social environment, meaningfulness of the activity, voluntariness of use, etc.).

2.3 Methods to Evaluate UX in Serious Games

While game developers should construct games that are rewarding, entertaining, and appealing to consumers in order to enhance game reviews and sales, designing and developing digital games is a demanding and difficult process [25]. Therefore, it is important to understand how different players behave and interact with games. Understanding target players and their game experiences during game development is critical to create a better user experience and perhaps improve game ratings and financial success.

A survey by the Entertainment Software Association (ESA) found that digital games have become an important part of the games industry in recent decades. Due to a number of variables such as rapidly growing market, broader player demographics, and unique controller interfaces and platforms, digital games are an important area of research [25].

Consequently, the opportunity is broader; however, a deeper understanding of player demographics and platforms is required to address this market. According to Mirza-Babaei [25], stereotypes of the single player (e.g., the image of a teenager addicted to digital games) are generally disappearing in the industry in favor of a new image of multiple players playing simultaneously on multiple devices. In modern digital games, there are different types of interaction that offer more opportunities for player interaction.

Through the growing field of games user research (GUR), developers are evaluating their games for usability and user experience to improve the gaming experience. Games user research borrows user research techniques from human-computer interaction (HCI) and psychology, such as behavioral observation, interviews, questionnaires, and heuristic evaluation. Despite advances in applying user research methods to understand the usability of productivity applications, researchers and practitioners still face challenges in applying these methods to digital games. Digital games have unique characteristics that prevent the application of most conventional user research methods to the evaluation of the game experience [25].

As a result, user research methodological approaches have been modified and improved to better meet the goals of game development. These methods aim to provide players with a combination of qualitative and quantitative/objective and subjective/formative and summative/short-term and long-term methods to choose from depending on their research context and the needs of their participants. One of the main issues facing user experience and game usability evaluation is determining the optimal combination of different methods and combining the data from each method into a relevant report for game developers.

2.4 Analysis of Methodologies

Users' experiences in digital games can be measured and evaluated using different methods. These methods are classified in various ways in the following sections.

2.4.1 Quantitative vs Qualitative Assessment

Qualitative methods are used to explore and understand players' perceptions and interactions. Users' experiences are usually recorded in non-numerical data. In contrast, quantitative methods use numerical data [26]. Quantitative approaches show levels of engagement and interest by providing statistics, while qualitative approaches capture players' experiences during play. There are times when players lack emotional expression and do not speak freely when evaluating verbally or nonverbally. It is difficult for players to concentrate and talk about their experiences at the same time while playing a game. When evaluating a project, both methods should be used to achieve objective and comprehensive results.

In any research, researchers have to make a primary but basic methodological choice between the quantitative and the qualitative approach (or their combination) to investigate their topic. With the quantitative approach, they can find out "what happens," while with the qualitative approach, they investigate "why it happens." The aim of qualitative research is to "discover the views of the research population by focusing on the perspectives from which individuals experience and feel about events" [27]. In summary, qualitative assessment involves categorizing and evaluating qualitative data to help researchers analyze and interpret game events, user behavior, and player experiences. Collecting qualitative data can lead us down such paths, whereas collecting quantitative data cannot, especially when it comes to user experience.

2.4.2 Subjective vs Objective Assessment

Instruments for measuring players experience fall into two categories depending on their reliability: objective and subjective.

Objective assessment instruments provide accurate data that are objective and free from any subjective judgment of the participants because they are accurately recorded by machines [28]. Objective data are recorded automatically and continuously without disturbing the participants or interfering them.

In contrast, subjective instruments, are not precisely because they are completed by the users themselves, contain subjectivity, so they have lower reliability compared to objective instruments. An objective assessment tool measures the expressive or psychophysiological aspect of the user's experience using facial expressions and collected psychophysiological data, while a subjective tool assesses the subjective feeling of the user's experience using self-reports, rating scales, and verbal protocols.

2.4.3 Short-Term vs Long-Term Assessment

In the early stages of game development, measuring users' initial and momentary experiences is important to obtain feedback [29]. It is also known that users' experiences change over time [30]. Therefore, it is necessary to use instruments that measure the experience over time to get more reliable information about game playability. In this way, a game developer can gain insight into how a player interacts with their game. Currently, user experience research mostly focuses on short-term evaluations. However, the relationship between a user and a game evolves over time, so long-term user evaluation is critical to a game's success.

These different categorizations are important because the reasons we want to measure user experience may vary from research to research. In some cases, we may want to measure qualitative attributes derived from the player's experience, while in other cases, we may want to measure quantitative attributes. Similarly, we may want to measure the player experience at a particular point in the game, such as when the player wins a significant player, or we may want to assess it over a longer period of time.

2.4.4 Formative vs Summative Evaluation

There are two types of evaluation in user experience, formative and summative. Which type of evaluation we should use depends on where we are in the process of developing a digital game.

Formative evaluations focus on identifying aspects of the design that work well or not well and why. These evaluations are conducted during the redesign of a game and provide information to gradually improve the game. Considering the case of designing a new digital game for mobile phones, as part of the design process, a prototype is created for this game and tested with (usually a few) users to see how easy it is to use and how players experience it. The research may reveal several weaknesses in the prototype, which are then addressed with a new design. This research is an example of a formative evaluation—it helps the designers determine what needs to be changed to improve the game. Formative evaluations involve testing and modifying the game, usually many times, and are therefore appropriate when developing a new game or redesigning an existing game. In both cases, the prototyping and testing steps are repeated until the game is ready for mass production [31].

Summative evaluation describes how well a game performs, often compared to a benchmark, such as a previous version of the game or a competitive game. Unlike formative evaluations, whose goals are to inform the design process, summative evaluations involve getting the big picture and evaluating the overall experience of a completed game. Summative evaluations are done less frequently than formative evaluations, usually immediately before or immediately after a redesign. Assume the redesign of the mobile phone game is complete, and now it is time to evaluate how well it performs compared to the previous version of the game. After the data from the survey is collected, it is then compared to the data obtained from the previous version of the game to see if there has been any improvement. This type of survey is a summative evaluation as it evaluates the product shipped with the goal of tracking performance over time and ultimately calculating our return on investment. However, during this study, we may uncover some usability issues. These issues should be noted and addressed during the next game design. Alternatively, another type of summative evaluations could compare results to those obtained from one or more competitive games or to known data across the gaming industry. All summary ratings give an overview of a game's usability. They are meant to serve as benchmarks so we can determine whether our own games have been improved over time. The final summative evaluation is the go/no-go decision on whether to release a product [31].

2.5 Overview of the Main Methodologies

There are a variety of tools and methods to uncover the quality of the experience generated by a game, either to improve it or to use the game for the purposes of education, training, awareness raising, and behavior change of subjects. Table 2.1 summarizes all UX assessment methods together with their assignment to one or more of the categories mentioned above.

2.5.1 Think-Aloud Protocol

The think-aloud protocol is a qualitative method of collecting data in which players describe their playing experiences to an expert facilitator. The facilitator pays attention to both verbal and nonverbal (e.g., behaviors, body language) players' responses to gain insights into the player experience [32]. Think-aloud protocol asks participants to spontaneously report any thoughts they have while they interact with a game without interpreting or analyzing what they have thought about [33]. The think-aloud protocol consists of two components: (a) the technique for collecting verbal data (think-aloud interview) and (b) the technique for predicting and analyzing verbal data (protocol analysis). The method is useful for researchers interested in observing, exploring, and understanding the thoughts and opinions of

Table 2.1 Overview of the main instruments and methods

Methods/instruments	Quantitative	Qualitative	Subjective	Objective	Short-term	Long-term	Formative	Summative
Think-aloud protocol		✓	✓		✓	✓	✓	✓
Surveys		✓	✓		✓	✓	✓	✓
Expert evaluation		✓	✓		✓	✓	✓	✓
Playtesting		✓	✓		✓	✓	✓	✓
Observation		✓	✓		✓	✓	✓	✓
Interviews		✓	✓		✓	✓	✓	✓
Focus groups		✓	✓		✓	✓	✓	✓
Psychophysiological measurements	✓	✓		✓	✓	✓	✓	✓
Self-assessment measurements	✓		✓		✓	✓	✓	✓

players, which can be a challenging endeavor [34, 35]. Depending on the interaction with the game, it can generate reports during the interaction or afterward [36].

In order to implement the think-aloud protocol, the following steps are taken: (a) users are assigned tasks, (b) users are asked to speak aloud their thoughts during the performance of the tasks, (c) users' thoughts are recorded as they are performing the tasks, and (d) the material is analyzed and commented on by the researcher(s). Based on Avouris [37], the think-aloud protocol can be divided into the following variations: (a) critical response protocols, in which the user is required to speak aloud only during a predetermined task, and (b) periodic report protocols, in which the user explains his/her thoughts only after completing a particular task so that the task is not disturbed.

The advantage of think-aloud protocol is that researchers are able to identify players' main misconceptions, since it allows them to understand how players view a game. Think aloud also enables them to obtain a rapid and high-quality response from a small number of participants [38]. A number of researchers have criticized the method for disrupting user concentration [39] and claiming that self-observation would interfere with thought process and, as a result, wouldn't show real thought processes [33].

2.5.2 Expert Evaluation

Expert evaluation refers to an overview of the game conducted by an expert or a team of experts. It is a formative or summative evaluation conducted by designers and user experience experts to identify potential problems and improve the design [40]. Expert evaluation can be conducted for an existing game to identify problems that can be fixed by redesigning the game. Expert evaluation of games under development can identify new problems before a prototype is created. Klas [41] describes two types of expert evaluation: In the first, the experts themselves act as evaluators, conduct the evaluation, and report on the results. In the second, the evaluators supervise the experts, lead the evaluation, and assess their performance. In comparison, expert evaluations provide quick and cost-effective results, in contrast to more expensive types of qualitative user studies, such as playtesting, which require more evaluators for a representative result [41].

In addition, expert evaluation can be used at different stages of the development process to identify usability issues early in the process [40]. Expert evaluation can be made more efficient through the use of heuristic analysis. A heuristic is a set of guidelines that help ensure design is consistent with best practices within an industry, and it is often used by researchers to support their evaluations [42]. The evaluators then come together to produce the results report.

Typical findings include:

(a) Which features of the game may cause usability problems and need to be improved

(b) Which features are likely to be successful and should be retained
(c) Which features should be tested with real players

2.5.3 Cognitive Walk-Through (CW)

Cognitive walk-through (CW) is a user interface design method that allows designers to model how a particular type of user will understand a user interface through exploration [43, 44] and to evaluate the learnability of a digital serious game [45]. It is an expert-based evaluation method that is therefore relatively inexpensive to implement and can be used to identify usability issues in a system effortlessly, quickly and economically [46]. As in expert evaluation, a team of reviewers walks through a task and evaluates the interface from a new user's perspective. As our main interest is in serious games, we propose that the cognitive walk-through is an appropriate and effective method to evaluate the learning potential of serious games, as both the design and evaluation practices of serious games can benefit from the cognitive walk-through method.

A cognitive walk-through cannot be conducted until the design of the game, the task scenario, the user assumptions, the scope of the game, and the sequence of actions that players must perform to successfully complete a given task are accurately described [43, 46]. Then, an evaluator or group of evaluators (2–6 expert evaluators) simulates a series of cognitive processes that users go through when completing a set of tasks. By understanding the behavior of the interface and its influence on players, evaluators are able to choose actions that are difficult for ordinary players. It would therefore be useful to use this evaluation method in the early stages of system development to ensure that users' needs are met.

2.5.4 Playtesting

The term playtesting refers to the use of traditional user testing methods for games [47]. The game design literature argues that playtesting is the most popular and most important method for game developers to evaluate their game designs. It is important for game developers to use playtesters to give feedback on unintended challenges in their games, to collect data on the way players prioritize tasks and goals, and to understand how players understand the mechanics of the game [31]. During playtesting, testers who have characteristics similar to those of the expected end users (e.g., age, education level, professional similarities, gaming experience) test the first and subsequent versions of a game and provide feedback to the game developers, which is then incorporated into the game design [48].

Playtesting can be formal (or open), informal (or closed), or beta. Formal playtesting can be conducted with non-design group members according to Korhonen [49]. Participants are usually required to fill out a questionnaire or provide contact information in order to be considered for participation. Several members of the design group can conduct informal playtesting. Finally, beta playtesting relates to the final phases of testing, before releasing a product to the public, and is sometimes conducted semi-formally with a limited version of the game to identify any last-minute issues.

2.5.5 Interviews

Interviews are an essential element of a qualitative evaluation session with users [50]. They provide one of the few ways of validating observations, discovering issues, gathering opinions, and determining the sources of challenges encountered by players [50]. Interviews can be used with other methodologies to enhance the gathered data and give a holistic perspective of the user's attitudes and emotions, and they are an essential element in identifying and understanding usability issues and obstacles in the player's experience [51]. Therefore, interviews seem to be the right choice for specific study aims and knowledge [52]. Nacke et al., for example, suggest using interviews to measure the PX and capture the context and social influences on the individual player's experience with serious games [5].

2.5.6 Focus Groups

Focus groups are a form of qualitative and subjective research. In a focus group, a group of people gather in a room to discuss a topic under guidance. It is a semi-structured interview process in which a small group of people, usually six or eight, discuss a specific study topic [19]. Krueger and Casey [53] describe the focus group method as a means of obtaining perceptions about a particular area of interest in a permissive, non-threatening environment (p. 5). To obtain qualitative data about the research topic, the moderator steers the discussion more or less according to its structure. Take a research project on user experience with a digital game. A more in-depth interview with the players might be necessary, but before we do that, we want to see what kinds of questions work and whether the players might raise issues we are not considering so that we can include them in our questions.

In a focus group, participants are selected based on their relevance and relationship to the topic. Therefore, they are not considered statistically representative of a significant population because they are not selected using strict probability sampling methods. Instead, participants are selected through random sampling, advertising, or snowballing, depending on the type of person and the characteristics the researcher wants to consider. There are several advantages of focus groups: It is a socially oriented research method that collects real-life data in a social setting, is flexible, has high validity, provides rapid results, and costs nothing to conduct. There are also some disadvantages of focus groups: the researchers have less control than with individual interviews, the data can sometimes be difficult to analyze, the moderators need certain skills, and the discussion needs to take place in a conducive environment.

2.5.7 Observation

Observation is a deeply qualitative research methodology that can be integrated into a variety of qualitative and quantitative research projects. Researchers can gain a great deal of data and information from their observations by watching users engage in a particular activity and then analyzing it. When observation is combined with other methods and techniques, it is possible to gather valuable data to interpret the topic the researcher is exploring.

The researcher must have specific skills, and the observation procedure involves some methodological risks, especially in terms of its validity and reliability, as the question of objectivity and impartiality is always present. Therefore, it is usually better for inexperienced researchers to combine this technique with another one, such as an interview, in order to collect all the data needed, to shed light on certain aspects of the study or to triangulate the information.

2.5.8 Surveys

Surveys may be used in research to examine player-game interactions and, depending on the results, improve the gaming experience [54]. The goal of surveys is to collect data on a subset of the population being studied by the researcher [55]. The survey results can then be extrapolated to the full population. Surveys are a quick, simple, and low-cost technique to collect a big amount of data that tells more about the subjective experience of playing a game [54, 56]. This may give the impression that creating a survey is simple, yet seemingly slight oversights can dramatically restrict the utility of your survey data.

Surveys can help researchers collect objective and subjective data. Objective data are directly observable and can be verified by others, such as demographic characteristics and the number of hours spent playing games. In contrast, subjective data are not objectively verifiable, such as attitudes and emotions. Overall, surveys can be used to assess player attitudes and experiences, motives, player characteristics, differences between groups of players, or different iterations of a design [54]. Their advantages include ease of use, use in many situations, minimal cost, access to large population, absence of interviewer bias, and fast transmission/response times [54, 56]. Surveys become even more effective when combined with other methods [54]. For example, while game analytics may indicate that players are more likely to succeed in a game, survey data may show that players were less challenged and bored [57]. In addition, survey data can be combined with physiological measures, such as facial recognition and electrodermal activity measurements [21]. Researchers can create their own questionnaires to measure outcomes or use existing, validated questionnaires to compare the results of their own studies with those of other studies.

Below are some of the most commonly used questionnaires:

2.5.8.1 The Player Experience of Need Satisfaction

According to Rigby and Ryan [58], people have three universal needs: competence (perception of a challenge), autonomy (voluntary aspects of an activity), and relatedness (connection to others). These are the main components of what we call Player Experience of Need Satisfaction (PENS) method. The PENS evaluation includes two additional factors, presence (the experience of being in the game world) and intuitive control, both of which are considered key features of games [59]. Using 7-point Likert scales, the PENS assesses these needs as well as the additional factors. When games meet these motivational criteria, the game experience and game success improve significantly. The PENS method is methodologically easy to apply as it successfully targets specific experiences related to need satisfaction and provides practically rapid feedback. These measurements can be easily applied to specific design or game concepts as well as to games that already have established.

2.5.8.2 Challenge Originating from Recent Gameplay Interaction Scale

The challenge originating from recent gameplay interaction scale (CORGIS) is a psychometric instrument developed by Denisova et al. [60]. This instrument is used to assess perceived challenge in digital games. The questionnaire assesses four types of perceived challenge in games:

Cognitive challenge: it stems from the need to plan ahead, memorize, exert effort, prepare, and multitask.

Performative challenge: it arises from the fact that the game requires the player to act quickly and accurately.

Emotional challenge: it arises from the emotions evoked in the player, which can also affect the things he thinks about outside the game.

Decision-making challenge: it arises from having to make decisions that are difficult or can lead to unfortunate outcomes.

2.5.9 Immersive Experience Questionnaire

Jennett et al. [61] developed the Immersion Experience Questionnaire (IEQ) to measure the level of immersion of players. It measures the user experience using a 5-

point Likert scale but focuses primarily on the concept of immersion. The IEQ uses positively and negatively worded questions. For every positively worded question, there is a negatively worded question, which adds accuracy to the questionnaire. The total score is the sum of the scores of the positively and negatively worded questions. When the IEQ was developed, it was assumed that immersion was based on five components. In practice, however, immersion is considered as a single dimension, with the components influencing the interpretation of the results.

2.5.9.1 Sensual Evaluation Instrument

The sensual evaluation instrument (SEI) was developed by Isbister et al. [62]. This is a nonverbal, body-based tool that can be used to capture shared responses more directly, saving designers time and energy and in turn increasing the likelihood that users will engage early in the design process. The SEI consists of eight sculptural objects that represent the range of emotions one would expect to experience when interacting with a digital game. The objects are not one-to-one with specific emotions. Rather, they are meant to serve as a starting point so that everyone can develop their own expressive taxonomy of the objects. People share their feelings as they engage in the experience. They arrange the objects as they wish or show in some way that they feel comfortable with the object or objects that correspond to their current feelings. In the end, the researcher watching the video in conjunction with SEI can better understand how the player felt during the game [63].

2.5.9.2 Game Experience Questionnaire

It is a tool designed specifically for young children (8–12 years old) to assess their gaming experiences. The game experience questionnaire (GEQ) [64] assesses seven different dimensions of gaming experience (immersion, flow, effectiveness, intensity, challenge, positive emotion, negative emotion) Each of the seven dimensions is distinguished into five sub-themes rated on a 5-point Likert scale. The game experience questionnaire is divided into three separate modules, each of which deals with a different experience: (1) core module, which evaluates the user's experience while playing the game; (2) social presence module, which evaluates the user experience while playing a game with others; and (3) post-game module, which evaluates the user's experience after completing the game. It has the advantage of measuring different aspects of the game experience (immersion, flow, effectiveness, intensity, challenge, positive emotions, and negative emotions), assessing the experience during and after the game, and assessing social presence as well. As it covers such a large area, it can be difficult to complete by all the researchers, so many researchers only use some of the modules.

2.5.10 Psychophysiological Measurements

Quantitative and qualitative researches both use psychophysiological measurements to assess users' experiences. As users' experiences during gameplay can have a significant impact on the playability of digital games, physiological data can be very useful to assess players' emotional state and performance, especially when correlated with subjective measurements [21]. So far, results have only been reported for first-person shooters games [65, 66]. The question arises whether physiological and subjective measurements might prove equally reliable for other types of digital games. The main methods for assessing user experience using physiological methods are as follows:

Electrodermal activity (EDA): perhaps the most commonly used physiological measurement. It is often referred to in the literature as galvanic skin response or skin conductance. Sweat gland secretions during play are indicators of positive arousal and mental activity [67, 68].

Cardiovascular activity measurement: an important physiological measure of human activity. Cardiovascular activity measures heart rate and heart rate variability [69, 70].

Electromyography (EMG): provides measurements of the electrical muscles. When a person is excessively anxious, skeletal movements are observed as a sign of involuntary muscle contractions during intense mental activity, intense emotions, and cognitive stress [46, 71, 72].

Facial expression: analyses human facial expressions during activity and measures basic human emotional states such as happiness, sadness, anger, surprise, disgust, etc. [73].

Electroencephalography (EEG) is performed with special electrodes that are attached to the participant's head during the test. Brain activity is then measured using frequency wave patterns that represent different mental activities [74, 75]. Since electrodes are used in electroencephalography, it is purely a laboratory measurement.

2.5.10.1 Biofeedback Measuring Device

The biofeedback measuring device is a device designed and built in the Laboratory of New technologies of the Department of Communication and Media Studies, University of Athens. This device consists of a sensor part housed on a typical computer mouse, an analogue electronic circuit that transmits the processed signal to a typical home computer, and finally a software component that converts the measurements into a suitable format. The STC is seamlessly detected by the contact of the thumb and ring finger with the Al-Si ring sensors, located on the left and right sides of the computer mouse, respectively (Fig. 2.1).

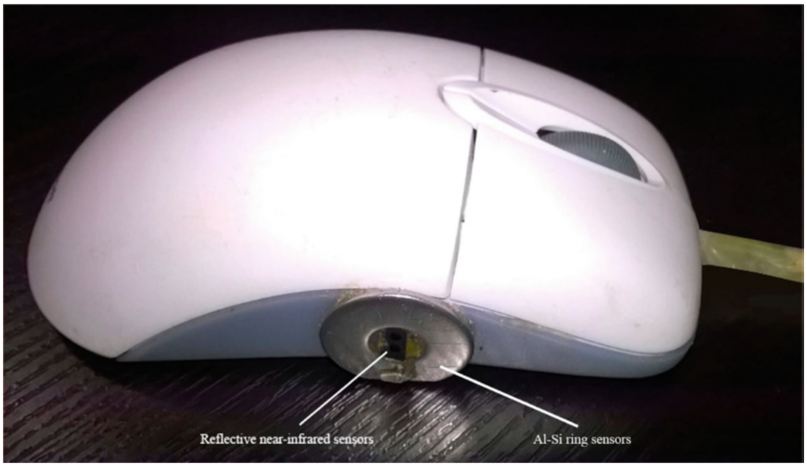


Fig. 2.1 The biofeedback measuring device

Heart rate is also detected by reflective near-infrared sensors in the center of the ring sensors (Fig. 2.1), based on the principle of reflective absorption that occurs during changes in skin coloration caused by the pulsation of blood in the tissue.

2.5.10.2 FaceReader

A software application called FaceReader was developed by Noldus Information Technology. The FaceReader software uses algorithms to rate facial images according to seven basic emotional states—happy, sad, angry, surprised, scared, disgusted, and "neutral emotional state." These seven emotions are rated from 0 (not at all) to 100 (perfect match). FaceReader "is an effective tool for measuring emotional experience during human-computer interaction, as it strongly suggests that more effective and well-designed systems elicit more positive emotions and fewer arousing falls than less effective applications" [21].

2.5.10.3 Self-Assessment Methods

Self-assessment methods are subjective, most often quantitative, and either short or long term. They provide players with the ability to self-evaluate or make judgments about their experience and the games they play based on specific self-assessment tools. Their great advantages are ease of use and the use in many situations. However, their disadvantage lies in the subjectivity of the judgments, which can be affected by a number of factors, including bias, differences in age and gender, economic and social status, and past experiences, among others.

2.5.10.4 Fun Toolkit

The Fun Toolkit was developed by Read and MacFarlane [76]. It consists of three separate questionnaires:

(a) Smileyometer: It is a measurement scale based on a 5-point Likert scale, with ratings from 1 “Poor” to 5 “Excellent.” The Smileyometer can be used both before and after the child’s experience with a digital application, be it an educational software or a website or a digital game. By using it before engaging with the application, we can gather information about the children’s expectations from the game. Using it latter, we can collect information about the fun of the game or the emotional experience of the players.

(b) Fun sorter table: A fun sorter table generally compares a set of products, whether they are educational software or digital games, as in our case. For a survey on children’s ratings of digital games, children compare and rank them from best to worst or from easiest to hardest or from what they intend to play again to what they intend to play less.

(c) Again and again table: The questionnaire consists of a table in which children mark whether they experienced each activity with a “Yes,” “Maybe,” or “No.” The idea for this tool comes from the field of psychology where it is argued that we are more likely to return to an activity we liked again and again if we like it. In the present study, children were asked, “Would you like to play with the toy again?”, and they had to answer accordingly.

2.5.10.5 Self-Assessment Manikin (SAM)

The Self-Assessment Manikin (SAM) is a system for evaluating three dimensions of gaming experience: valence, arousal, and dominance [77]. It uses three pictorial scales, illustrating cartoon creatures. All three scales are 9-point and take values from 1 to 9, with 5 representing the middle of the scale. Although it is stated that it is a weighted method, there are insufficient studies that support this claim. Its advantages include ease of completion and its ability to be used in different circumstances. The disadvantages are what all objective assessment tools suffer from: objectivity of judgment and difficulty in matching experience with graphic.

2.5.10.6 UX Curve

The UX Curve is a tool for retrospectively evaluating user experiences. There is a timeline and a horizontal area in which the user can graph his positive and negative experiences. The advantage of UX Curve is that it allows the user to design the most immersive game experience. Nevertheless, its disadvantage is that it relies on retrospective memory from the game rather than reality for its completion [78].

2.5.10.7 MemoLine

The MemoLine is actually a timeline that can be used for retrospective evaluations. There are as many frames as there are time periods in which the user plays a game. As the tool is intended for children, the experiences they have are represented by three different colors: green represents positive experiences, red represents negative experiences, and gray represents times when the game is not played, e.g., weekends. Users are given questionnaires for each of these game scenarios: usability, challenge, quantity, and general impression [79].

The above questionnaires are certainly not the only ones. There are a large number of other relative questionnaires such as Emo-watch, EGameFlow, Gameful Experience Questionnaire, Model for the Evaluation of Educational Games (MEEGA+), Game User Experience Satisfaction Scale (GUESS), iScale CORPUS (Change Oriented analysis of the Relationship between Product and USer), and many others.

2.6 Discussion

The aim of this chapter is to provide an overview of evaluation methods to game developers and researchers whose research interests are related to digital serious games. This process is extremely important, considering that serious games differ from games whose goal is to entertain players, rather than teach or train them. It is also very important not only to describe these methods but also to highlight the advantages and disadvantages of each method, as well as to explain when, how, and why it makes sense to use each of these evaluation instruments. As it has been discussed in this chapter, the tools for evaluating player experience can be divided into four groups: objective-subjective, quantitative-qualitative, formative-summative, and short term-long term.

Beginning with the objective and subjective instruments for evaluating players' experiences, things are plain. Objective evaluation instruments provide objective data, free from any subjective judgment. Data are accurately recorded by machines and software, without disturbing or interfering participants. In contrast, subjective instruments are not accurate, as they are completed by the users themselves and therefore have lower reliability than objective instruments. Each of these evaluation methods has its own advantages and disadvantages. On the one hand, objective evaluation provides reliable results but is difficult to be applied as it requires expensive equipment and is a purely laboratory procedure. In contrast, subjective evaluation is easier to be applied, since it only requires finding suitable subjects, whether they are players or experts, but the data collected is less reliable due to the subjectivity of the participants. An evaluation system that uses both objective (e.g., a skin conductance measurement) and subjective methods (e.g., a self-reported questionnaire) to evaluate players' experiences is proposed to overcome the disadvantages and benefit from both forms of evaluation. Therefore, researchers are able to collect data that is free of users' biases, while at the same time they can interpret it based on users' perceptions and opinions.

A lot of information can be gained from both formative and summative evaluation, which can be used by developers to improve their games. In formative evaluation, developers and experts or players have a dialogue about game play, which helps gather information for game design. An evaluation of this kind identifies what aspects of the design work well and what aspects don't. As a game is being redesigned, these evaluations provide information that can be used to improve the game gradually. As opposed to formative evaluation, summative evaluation discusses how well a game performs, usually in comparison with a benchmark, such as a previous version or a competitive game. A summative evaluation takes a step back from formative evaluations, which aim to inform the design process, and instead looks at the big picture and evaluates the overall experience. In most cases, summative evaluations are conducted just before or just after a redesign, and they are less frequent than formative evaluations. A developer can thus use formative or summative evaluation based on what they want to measure and the stage at which it is being developed. Formative and summative evaluations can be implemented with most of the tools described and suggested in this chapter, and the development team can decide which types to use. As a general rule, formative evaluations produce qualitative data, and summative evaluations produce quantitative data. To conduct a formative evaluation, developers should rely on instruments such as a think-aloud protocol, cognitive walk-throughs, observation, focus groups, interviews, etc. To conduct a summative evaluation, developers should rely on instruments such as psychophysiological and self-assessment measurements.

Serious game evaluation is essential for any developer, as it is an important function at every stage of game development. A comprehensive evaluation of players' experience is beneficial to a developer in many ways. It is a well-known method to assess the strengths and weaknesses of the game experience, which further serves as a basis for working and improving the overall game experience. Usually, the evaluation is done at the end when the game is ready for use. However, some developers also evaluate player experiences in the short term (during development). This has its own advantages, because if the game development is not going in the desired direction, the developer can correct it, instead of waiting for the end of the development and then making corrections.

Lastly, qualitative methods provide statistics about player engagement and interest, while quantitative approaches help developers and researchers study players' perceptions and interactions. The researcher must choose a methodological approach (or a combination of both) when researching any topic (either quantitative or qualitative). In a quantitative approach, developers discover "what happens," while in a qualitative approach, they discover "why it happens." In summary, qualitative assessment involves categorizing and evaluating qualitative data to help us analyze and interpret game events, user behavior, and player experiences. Collecting qualitative data can lead us down such paths, whereas collecting quantitative data cannot, especially when it comes to user experience.

2.7 Conclusions

This chapter is intended to serve as a guide for serious game developers and researchers who wish to evaluate existing games, to improve the players' experience and reach an optimal level. A player experience evaluation should record and interpret players' experiences of interacting with a digital game, and it is important that these records are accurate and reliable in order to produce meaningful and useful results. It is also important that an evaluation can identify the situations and factors that impact the player experience and make it more or less positive. In this case, we can make the necessary adjustments and changes to improve the player experience. According to what has been discussed in this chapter, the tools for evaluating the player experience can be divided into four groups: objective-subjective, quantitative-qualitative, formative-summative, and short term-long term.

Since the game experience is multidimensional and difficult to measure, it is important to use methods with different characteristics. Measurement and evaluation of player experience should be done using instruments derived from different methods, e.g., quantitative instruments and qualitative evaluation instruments or objective instruments and qualitative evaluation instruments. It is possible to negatively impact our evaluation efforts if we only use instruments from a single methodology.

Last but not least, the methodology we use to evaluate the user experience is crucial for understanding and interpreting the experience of playing a digital serious game. Future research should evaluate digital games using different evaluation methods and instruments. These studies should ultimately aim to find the most effective combination of tools and methods to measure the potential of a game.

References

1. Nacke, L.E.: Affective ludology: scientific measurement of user experience, interactive entertainment. Blekinge Institute of Technology (2009)
2. Mandryk, R.L., Atkins, M.S., Inkpen, K.M.A.: Continuous and objective evaluation of emotional experience with interactive play environments. In: Proceedings of CHI 2006, Montréal, Québec, Canada, April 2006, pp. 1027–1036. ACM (2006)
3. Drachen, A., Canossa, A.: Towards gameplay analysis via gameplay metrics. In: Proceedings of MindTrek, Tampere, Finland, October 1–2. ACM (2009)
4. Kim, J.H., Gunn, D.V., Schuh, E., Phillips, B., Pagulayan, R.J., Wixon, D.: Tracking real-time user experience (TRUE): a comprehensive instrumentation solution for complex systems. In: Proceedings of CHI 2008, pp. 443–452. ACM, Florence, Italy (2008)
5. Nacke, L., Niesenhaus, J., Engl, S., Canossa, A., Kuikkaniemi, K., Immich, T.: Bringing digital games to user research and user experience. In: Proceedings of the Entertainment Interfaces Track 2010 at Interaktive Kulturen 2010 ceur workshop proceedings, 12–15 September (2010)
6. Pagulayan, R., Keeker, K., Wixon, D., Romero, R.L., Fuller, T.: User-centered design in games. In: The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications. L, pp. 883–906. Erlbaum Associates, New York, NY (2003)

7. Pagulayan, R., Steury, K.R., Fulton, B., Romero, R.L.: Designing for fun: user-testing case studies. In: Funology: From Usability to Enjoyment, pp. 137–150. Kluwer Academic Publishers, Norwell, MA (2004)
8. Desurvire, H., Caplan, M., Toth, J.A.: Using heuristics to evaluate the playability of games. In: CHI '04 Extended Abstracts, pp. 1509–1512. ACM, Vienna (2004)
9. Korhonen, H., Koivisto, E.M.I.: Playability heuristics for mobile games. In: Proceedings of Conference on HCI with mobile devices and services, pp. 9–16. ACM, Espoo, Finland (2006)
10. Csikszentmihalyi, M.: Flow: the psychology of optimal experience, vol. 39, 1st edn. Harper Collins, New York (1990)
11. Sweetser, P., Wyeth, P.: GameFlow: a model for evaluating player enjoyment in games. Comput. Entertain. **3**(3), 1–24 (2005)
12. Brown, E., Cairns, P.: A grounded investigation of immersion in games. In: ACM Conference on Human Factors in Computing Systems, CHI 2004, pp. 1297–1300. ACM Press (2004)
13. Ermi, L., Mayra, F.: Fundamental components of the gameplay experience: analysing immersion. In: de Castell, S., Jenson, J. (eds.) Proceedings of Chancing Views – Worlds in Play. Digital Games Research Association's Second International Conference, Vancouver (2005)
14. Gilleade, K.M., Dix, A.: Using frustration in the design of adaptive videogames. In: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology – ACE '04, pp. 228–232. (2004)
15. Funk, J.B., Pasold, T., Baumgardner, J.: How children experience playing video games. In: Proceedings of the Second International Conference on Entertainment Computing Pittsburgh, Carnegie Mellon University, pp. 1–14 (2003)
16. Bracken, C., Lange, R.L., Denny, J.: Online video games and gamers' sensations of spatial, social, and copresence. FuturePlay 2005. East Lansing (2005)
17. Nacke, L., Lindley, C., Stellmach, S.: Log who's playing: psychophysiological game analysis made easy through event logging. In: Proceedings of Fun and Games, 2nd International Conference, Eindhoven, The Netherlands, October 20–21, pp. 150–157. Springer (2008)
18. Almeida, S.: The player and video game interplay in the gameplay experience construct. PhD, Universidade de Aveiro (2013)
19. Kirginas, S., Gouscos, D.: Development and validation of a questionnaire to measure perceptions of freedom of choice in digital games. Int. J. Serious Games (IJSG). **3**(2), 29–45 (2016)
20. Almeida, S., Veloso, A., Roque, L., Mealha, O., Moura, A.: The video game and player in a gameplay experience model proposal. In: Proceedings of Videojogos 2013 – 6th Annual Conference in the Science and Art of Video Games. University of Coimbra, Coimbra, Portugal (2013)
21. Kirginas, S., Psaltis, A., Gouscos, D., Mourlas, C.: Studying children's experience during free-form and formally structured gameplay. Int. J. Child-Comput. Interact. **28** (2021)
22. Kirginas, S., Gouscos, D.: Exploring the impact of free-form and structured digital games on the player experience of kindergarten and primary school students. In: Russell, D., Laffey, J. (eds.) Handbook of Research on Gaming Trends in P-12 Education, pp. 394–420. Hershey, PA, Information Science Reference (2016)
23. Roto, V.: User Experience from Product Creation Perspective. Towards a UX Manifesto, pp. 31–34 (2007)
24. Lallemand, C.: Towards consolidated methods for the design and evaluation of user experience. Doctoral dissertation, University of Luxembourg (2015)
25. Mirza-Babaei, P.: Getting ahead of the game: challenges and methods in games user research. User Exp. Magaz. **15**(2) (2015) https://uxpamagazine.org/getting-ahead-of-the-game/
26. Neill, J.: Qualitative & Quantitative Research. http://wilderdom.com/research/QualitativeVersusQuantitativeResearch.html (2009)
27. Bird, M., Hammersley, M., Gomm, R., Woods, P.: Educational Research in Action/ Fragkou E: Translate in Greek. Educational Research in Practice-Study Manual, Patras (1999)
28. Cacioppo, J., Tassinary, L., Berntson, G.: Handbook of Psychophysiology, 3rd edn. Cambridge University Press, New York (2007)

29. Vermeeren, A., Lai-Chong Law, E., Roto, V., Obrist, M., Hoonhaut, J., Väänänen-Vainio-Mattila, K.: User experience evaluation methods: current state and development needs. In: Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries: NordiCHI '10 (2010)
30. Fenko, A., Schifferstein, H.N.J., Hekkert, P.: Shifts in sensory dominance between various stages of user-product interactions. Appl. Ergon. **41**, 34–40 (2010)
31. Joyce, A.: Formative vs. Summative Evaluations [blog]. https://www.nngroup.com/articles/formative-vs-summative-evaluations/ (2019, July 28)
32. Pellicone, A., Weintrop, D., Ketelhut, D.J., Shokeen, E., Cukier, M., Plane, J.D., Rahimian, F.: Playing aloud: leveraging game commentary culture for playtesting. Int. J. Gaming Comput.-Mediat. Simulat. (IJGCMS). **14**(1), 1–16 (2022)
33. Guss, C.D.: What is going through your mind? Thinking Aloud as a method in cross-cultural psychology. Front. Psychol. **9**, 1292 (2018)
34. Johnson, W.R., Artino, A.R., Jr., Durning, S.J.: Using the think aloud protocol in health professions education: an interview method for exploring thought processes: AMEE Guide No. 151. Medical teacher, 1–12. Advance online publication (2022)
35. Lundgren-Laine, H., Salantera, S.: Think-aloud technique and protocol analysis in clinical decision-making research. Qual. Health Res. **20**(4), 565–575 (2010)
36. Zhang, X., Simeone, A.L.: Using the think aloud protocol in an immersive virtual reality evaluation of a virtual twin. In: Proceedings of the 2022 ACM Symposium on Spatial User Interaction (SUI '22), pp. 1–8. Association for Computing Machinery, New York, NY (2022)
37. Avouris, N., Katsanos, C., Tselios, N., Moustakas, K.: Introduction to Human-Computer Interaction [Undergraduate textbook]. Chapter 11. Kallipos, Open Academic Editions (2015)
38. Jordan, P.W.: An introduction to usability. Taylor & Francis, London (1998)
39. Hashemi Farzaneh, H., Neuner, L.: Usability evaluation of software tools for engineering design. In: Proceedings of the 22nd International Conference on Engineering Design (ICED19), Delft, The Netherlands, 5–8 August (2019)
40. Nova, A., Sansalone, S., Robinson, R., Mirza-Babaei, P.: Charting the uncharted with GUR: how AI playtesting can supplement expert evaluation. In: Proceedings of the 17th International Conference on the Foundations of Digital Games (FDG '22). Association for Computing Machinery, New York, NY, Article 28, pp. 1–12 (2022)
41. Klas, C.: Expert evaluation methods. In: Dobreva, M., O'Dwyer, A., Feliciati, P. (eds.) User Studies for Digital Library Development, pp. 75–84. Facet (2012)
42. Rajanen, M., Rajanen, D.: Heuristic evaluation in game and gamification development. In: Proceedings of GamiFin 2018 Conference, Pori (2018)
43. Allendoerfer, K., Aluker, S., Panjwani, G., Proctor, J., Sturtz, D., Vukovic, M., Chen, C.: Adapting the cognitive walkthrough method to assess the usability of a knowledge domain visualization. In: IEEE Symposium on Information Visualization, 2005. INFOVIS 2005, pp. 195–202. IEEE (2005)
44. Farrell, D., Moffat, D.C.: Adapting cognitive walkthrough to support game based learning design. Int. J. Game-Based Learn. (IJGBL). **4**(3), 23–34 (2014)
45. Salazar, K.: Evaluate Interface Learnability with Cognitive Walkthroughs. [online]. https://www.nngroup.com/articles/cognitive-walkthroughs/ (2022)
46. Farzandipour, M., Nabovati, E., Sadeqi Jabali, M.: Comparison of usability evaluation methods for a health information system: heuristic evaluation versus cognitive walkthrough method. BMC Med. Inf. Decis. Making. **22**(1), 1–1 (2022)
47. Yanez-Gomez, R., Cascado-Caballero, D., Sevillano, J.L.: Academic methods for usability evaluation of serious games: a systematic review. Multimed. Tools Appl. **76**, 5755–5784 (2017)
48. Koutsabasis, P., Gardeli, A., Partheniadis, K., Vogiatzidakis, P., Nikolakopoulou, V., Chatzigrigoriou, P., Vosinakis, S.: Field playtesting with experts' constructive interaction: an evaluation method for mobile games for cultural heritage. In: 2021 IEEE Conference on Games (CoG), pp. 1–9 (2021)

49. Korhonen, H.: Comparison of playtesting and expert review methods in mobile game evaluation. In: Fun and Games '10: Proceedings of the 3rd International Conference on Fun and Games, pp. 18–27 (2010)
50. Drachen, A., Mirza-Babaei, P., Nacke, L.E.: Games User Research. Oxford University Press (2018)
51. Carneiro, N., Darin, T., Pinheiro, M., Viana, W.: Using interviews to evaluate location-based games: lessons and challenges. J. Interact. Syst. **11**(1), 125–138 (2020)
52. Isbister, K., Schaffer, N.: Game usability: advice from the experts for advancing the player experience. Morgan Kaufmann Publishers, Burlington, MA (2008)
53. Krueger, R.A., Casey, M.A.: Focus groups A practical guide for applied research. Sage Publications, Thousand Oaks (2000)
54. Bruhlmann, F., Mekler, E.D.: Surveys in games user research. In: Drachen, A., Mirza-Babaei, P., Nacke, L. (eds.) Games User Research, pp. 141–162. Oxford University Press, Oxford (2018)
55. Rahman, M.M., Tabash, M.I., Salamzadeh, A., Abduli, S., Rahaman, M.S.: Sampling techniques (probability) for quantitative social science researchers: a conceptual guidelines with examples. Seeu Rev. **17**(1), 42–51 (2022)
56. Story, D.A., Tait, A.R.: Survey research. Anesthesiology. **130**(2), 192–202 (2019)
57. Hazan, E.: Contextualizing data. In: El-Nasr, M.S., et al. (eds.) Game analytics, pp. 477–496. Springer, London (2013)
58. Rigby, S., Ryan, R.: The Player Experience of Need Satisfaction (PENS) Model, pp. 1–22. Immersyve Inc (2007)
59. Ryan, R., Rigby, S., Przybylski, A.: The motivational pull of video games: a self-determination theory approach. Motiv. Emot. (2006)
60. Denisova, A., Cairns, P., Guckelsberger, C., Zendle, D.: Measuring perceived challenge in digital games: development & validation of the challenge originating from recent gameplay interaction scale (CORGIS). Int. J. Hum.-Comput. Stud., 137 (2020)
61. Jennett, C., Cox, A.L., Cairns, P., Dhoparee, S., Epps, A., Tijs, T., Walton, A.: Measuring and defining the experience of immersion in games. Int. J. Hum.-Comput. Stud. **66**(9), 641–661 (2008)
62. Isbister, K., Höök, K., Sharp, M., Laaksolahti, J.: The sensual evaluation instrument: developing an affective evaluation tool. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1163–1172 (2006)
63. Laaksolahti, J., Isbister, K., Höök, K.: Using the sensual evaluation instrument. Digit. Creativ. **20**(3), 165–175 (2009)
64. Ijsselsteijn, W.A., Poels, K., de Kort, Y.A.W.: The game experience questionnaire: development of a self-report measure to assess player experiences of digital games, deliverable 3.3. FUGA technical report, TU, Eindhoven, The Netherlands (2008)
65. Nacke, L.E., Grimshaw, M.N., Lindley, C.A.: More than a feeling: measurement of sonic user experience and psychophysiology in a first-person shooter game. Interact. Comput. **22**(5), 336–343 (2010)
66. Drachen, A., Nacke, L.E., Yannakakis, G., Lee Pedersen, A.: Correlation between heart rate, electrodermal activity and player experience in First-Person Shooter games. In: Spencer, S.N. (ed.) Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games, pp. 54–2010. ACM, Los Angeles, CA (49)
67. Ortega, M.C., Bruno, E., Richardson, M.P.: Electrodermal activity response during seizures: a systematic review and meta-analysis. Epilepsy Behav. **134**, 108864 (2022)
68. Paloniemi, S., Penttonen, M., Eteläpelto, A., Hökkä, P., Vähäsantanen, K.: Integrating self-reports and electrodermal activity (EDA) measurement in studying emotions in professional learning. In: Methods for Researching Professional Learning and Development, pp. 87–109. Springer, Cham (2022)
69. Richter, H., Krukewitt, L., Müller-Graf, F., Zitzmann, A., Merz, J., Böhm, S., Kühn, V.: High resolution EIT based heart rate detection using Synchrosqueezing (2022)

70. Thissen, B.A., Schlotz, W., Abel, C., Scharinger, M., Frieler, K., Merrill, J., Haider, T., Menninghaus, W.: At the heart of optimal reading experiences: cardiovascular activity and flow experiences in fiction reading. Read. Res. Q. **57**(3), 831–845 (2022)
71. Fuentes del Toro, S., Wei, Y., Olmeda, E., Ren, L., Guowu, W., Díaz, V.: Validation of a low-cost electromyography (EMG) system via a commercial and accurate EMG device: pilot study. Sensors. **19**(23), 5214 (2019)
72. Tortora, G., Derrickson, B.: Principles of Anatomy and Physiology, 12th edn. Biological Sciences Textbook (2009)
73. Ramos, A.L.A., Dadiz, B.G., Santos, A.B.G.: Classifying emotion based on facial expression analysis using Gabor filter: a basis for adaptive effective teaching strategy. In: Computational Science and Technology, pp. 469–479. Springer, Singapore (2020)
74. Westover, M.B., Gururangan, K., Markert, M.S., Blond, B.N., Lai, S., Benard, S., et al.: Diagnostic value of electroencephalography with ten electrodes in critically ill patients. Neurocrit. Care. **33**(2), 479–490 (2020)
75. Masood, K., Alghamdi, M.A.: Modeling mental stress using a deep learning framework. IEEE Access. **7**, 68446–68454 (2019)
76. Read, J.C., MacFarlane, S.J.: Using the Fun Toolkit and other survey methods to gather opinions in child computer interaction. In: Interaction Design and Children, IDC2006. ACM Press, Tampere (2006)
77. Lang, P.J.: The cognitive psychophysiology of emotion: fear and anxiety. In: Tuma, A.H., Maser, J.D. (eds.) Anxiety and the Anxiety Disorders, pp. 131–170. Lawrence Erlbaum Associates (1985)
78. Kujala, S., Roto, V., Mattila, K., Karapanos, E., Sinnela, A.: UX curve: a method for evaluating long-term user experience. Interact. Comput. **23**, 473–483 (2011)
79. Vissers, J., De Bot, L., Zaman, B.: MemoLine: evaluating long-term UX with children. In: Proceedings of the 12th International Conference on Interaction Design and Children, New York, pp. 285–288 (2013)

Chapter 3
Software Engineering for Dynamic Game Adaptation in Educational Games

Vipin Verma, Ashish Amresh, Tyler Baron, and Ajay Bansal

Abstract Educational games and game-based assessments have evolved over the past several years and are continuing to evolve. They promote student engagement in the learning process by creating an interactive environment where they can learn in a fun and challenging way. This gives them the potential to yield diagnostic information to educators and feedback to students. During the game play process, game-based assessment (GBA) can be used to assess the learning imparted by the game to the students. A common strategy for GBA has been to utilize surveys and built-in quizzes to measure student learning during the game play. However, this impacts students' attention negatively as they need to change their attention from game play to the assessment and back. Stealth assessment provides a natural alternative for assessment of learning without breaking the delicate flow of engagement. It aims to blur the lines between assessment and learning by weaving them together within the game. Stealth assessment uses game play interaction data to build inferences about student performance and learning. As an advantage, it provides ways to assess hard-to-measure constructs such as learning proficiency, critical thinking, persistence, and other twenty-first-century skills. Designing and developing an educational game takes time, and repeating the process for every new content or concept can be inefficient. The authors provide a framework called content-agnostic game engineering (CAGE) that can be used to create multiple learning contents within a single game by reusing already developed educational game mechanics. CAGE helps reduce time for creating an educational game by building content-agnostic mechanics that could be used across multiple content topics. It does so by separating the game into three components of mechanics, content, and student modeling that operate independently. Additionally, stealth assessment can be integrated into CAGE as a part of the student model and can

V. Verma (✉) · T. Baron · A. Bansal
Arizona State University, Tempe, AZ, USA
e-mail: tjbaron@asu.edu; Ajay.Bansal@asu.edu

A. Amresh
Northern Arizona University, Flagstaff, AZ, USA
e-mail: amresh@asu.edu

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_3

also be content agnostic as a way to demonstrate the advantages of adopting a CAGE-based development framework. While CAGE can work with multiple content domains, it cannot work with every domain. The limit is decided by how the game mechanics are implemented. In this chapter, we discuss the software practices to implement CAGE architecture and ways to embed stealth assessment in a content-agnostic way.

Keywords Content agnostic · Stealth assessment · Dynamic adaptation · Bayesian network · Facial tracking

3.1 Introduction

Educational assessment has evolved over the past several years from traditional pen-and-paper tests to forms such as game-based assessment and continues to evolve. It must provide feedback to learners and diagnostic information to teachers [1]. Game-based learning offers an interactive environment for the students to learn in a fun and challenging way while keeping them engaged in the learning process. Game-based assessment (GBA) offers a way to assess them while they are interacting with the game. GBA may consist of built-in quizzes and surveys to assess the student learning while they are playing. However, such methods tend to break the students' attention from learning to complete the assessment. Stealth assessment is a way to assess the learners while they are playing the game without breaking their flow, eventually blurring the lines between learning and assessment by integrating them [2]. It utilizes the data generated during the game play to make inferences about the student learning and performance at various grain sizes. Further, it can be used to assess skills such as creativity [3], systems thinking [4], persistence [5], and other twenty-first-century skills, which are needed for success today [6]. These skills in general are hard to assess by traditional means of assessment.

3.1.1 Research Background

Evidence-centered design (ECD) is an instructional theory that assessments should be focused on evidence-based arguments [7]. Namely, that a student's understanding of the material can be measured by their reaction to changes in observable variables. Making these changes is something that GBL can do well as dynamically changing the problem during learning is easier with GBL than most traditional teaching methods. The issue that arises with this is the need to ensure that the change flows naturally within the game system and does not break the player's flow or distract from the learning process [2]. It is here that stealth assessment techniques can ensure that these measurements do not require overt actions that remind the player that they are currently being evaluated on their understanding of the given topic. Many methods of stealth assessment are also helpful in measuring student reactions to unobservable variables as these are harder to detect but also important to measure in order to gauge student understanding [7]. Finally, it is important that the learning assessment be well integrated into the game itself and not stand out as a separate experience [8]. Not only would that break student immersion, but games that integrate the two are both more effective teaching tools and generally better liked by students using them.

3.1.2 Motivation

Presently, serious games are designed with an emphasis on the educational content of the game, which is a good approach but has some issues [9]. Such games are usually not as engaging when compared to the entertainment games, as the educational content of the game takes precedence over entertainment while designing the game. Further, the game is strongly connected to the learning content, due to which the programming code, game mechanics, game design, and learning assessment cannot be effectively re-used for teaching another content. This issue is compounded by the complex level of observation needed to assess students' reactions to non-observable variables in order to properly assess student learning [7]. The setup required to do these measurements causes the measures themselves to also be strongly connected to the learning content. This creates a scenario where readopting existing games to other educational content becomes extremely difficult as the game mechanics need to be rebuilt and the learning assessment must be completely revisited as well. The authors delineate the software framework called content-agnostic game engineering (CAGE) that can be used to alleviate this problem. It uses content-agnostic mechanics across a set of learning contents to develop the game. Further, [10] observed that the explicit assessment of the learning content in the form of tests and questionnaires can impact the player engagement in a negative way. Therefore, [1] integrated stealth assessment in the CAGE framework. It was built in a content-agnostic manner so that the assessment can be re-used across multiple content domains and thus contribute to reduction in the overall game development time to a considerable extent.

3.1.3 Chapter Outline

This chapter will target the emerging software practices in GBA and game adaptation. It will cover three main points: conducting stealth assessment, the CAGE framework, and the student model. Within stealth assessment, this chapter will elaborate upon how it has been used to assess twenty-first-century skills, which are hard to measure otherwise, while keeping the learners engaged with the learning process [11]. Stealth assessment includes techniques like mouse and touch-tracking

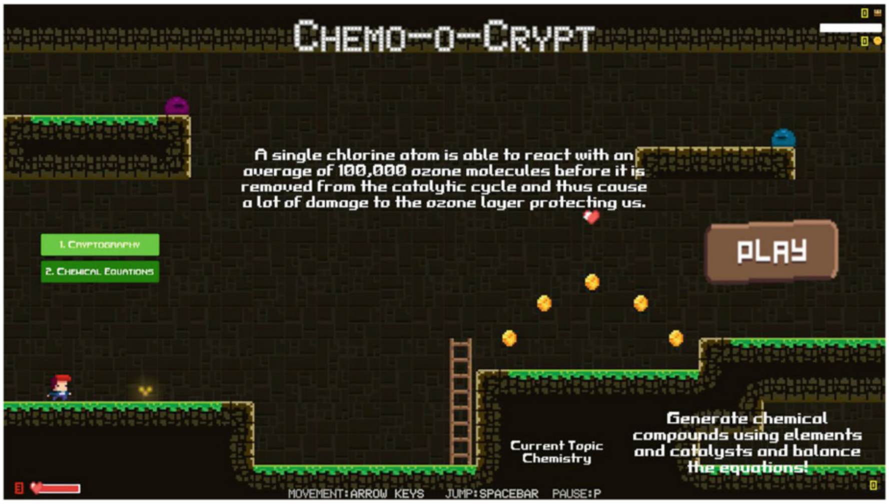

CHEMO-O-CRYPT
A single chlorine atom is able to react with an average of 100,000 ozone molecules before it is removed from the catalytic cycle and thus cause a lot of damage to the ozone layer protecting us.
1. CRYPTOGRAPHY
2. CHEMICAL EQUATIONS
PLAY
Current Topic Chemistry
Generate chemical compounds using elements and catalysts and balance the equations!
MOVEMENT:ARROW KEYS JUMP:SPACEBAR PAUSE:P

Fig. 3.1 A game developed using CAGE architecture

and how they have been used to measure the students' cognitive load without explicitly asking it from the learners [12]. The CAGE section will incorporate how a single assessment design can be used in the assessment of multiple content domains. Figure 3.1 shows a game developed using the CAGE architecture developed by Baron [10]. This game teaches cryptography and chemistry, one at a time to its learners. So, a single game is made with an effort of two and targets two content domains at once. Similarly, a single assessment strategy is embedded in the game that targets assessment of both the content domains. We delineate the process to implement a content-agnostic game and examine its feasibility for multiple serious games creation with an embedded assessment. We also probe into the effectiveness of game adaptation within the CAGE framework. Finally this chapter will discuss the inclusion of the student model, representing the learning state of a student at any point of time. This will include modeling the student's learning state using their performance parameters and how it can be used to dynamically adapt the game during run-time to accommodate the learning style and knowledge state of the student [13].

3.2 Stealth Assessment

A process that involves using data to determine if the learning goals are met or not is called assessment [14]. It is an equally important process when compared to designing the game mechanics and learning content of the game [1]. Stealth assessment is a technique in which the assessment is directly woven into the fabric of the game such that it is undetectable. It can be used to make inferences about the player performance by utilizing the data generated during the game play [15]. It has been used for the assessment of many twenty-first-century skills such as systems thinking [4], persistence [5], creativity [3], problem-solving skills [16], team performance [17], and causal reasoning [18]. With the help of large volumes of data generated during the game play, stealth assessment can be used to identify the process which a player follows to solve a problem in the game [12]. Other assessment techniques, such as survey questionnaires can break the flow of learners and disengage them from the learning process. However, stealth assessment can replace these obtrusive assessment techniques to keep their flow intact [19]. It can also be used to provide immediate feedback to the learners during their game play session, which they can immediately incorporate into their game play [20]. The real-time feedback can further be used to dynamically adjust the game difficulty in real time to the level appropriate for the learner [21].

3.2.1 *Stealth Assessment Implementation Techniques*

There are a number of ways that can be used to implement stealth assessment in serious games. In this section, these methods are discussed along with their software implementation techniques.

3.2.1.1 Mouse and Touch-Tracking

This technique can be used in games that employ a computer mouse or a touchscreen device to play. It has been used for the assessment of memory strength [22], positive and negative emotions [23], gender stereotypes [24], cognitive load [25], and numerical representation [26]. In this technique, the mouse or touch coordinates are tracked as the user moves their cursor across the screen. It can reveal the hidden bias or the intent of the user while they make a decision to reach a certain target.

To implement mouse-tracking in Web-browser-based games, *mousemove* event is available in the built-in Web APIs for JavaScript [27]. This event is fired whenever mouse is moved across a target element. Further, the *pageX* and *pageY* can be used to get the X and the Y coordinates of the point where the *mousemove* event was triggered [28]. Both *pageX* and *pageY* are available to use as a part of the *MouseEvent*. These X and Y coordinates can then be used to trace the path of the mouse as it is dragged across the screen by the user, which is used to assess user's intent or inherent bias in their thought process. Below is an example code snippet that tracks mouse movement within an html id element called *toBeTracked* and reads their x and y coordinates:

```
$('#toBeTracked').mousemove(trackMouseMovement);
function trackMouseMovement(event) {
  var eventDoc, doc, body;
  var areaHeight = $('#toBeTracked').height();
  if (event.pageX == null && event.clientX != null) {
        eventDoc = (event.target &&
          event.target.ownerDocument);
        eventDoc = document || eventDoc;
        doc = eventDoc.documentElement;
        body = eventDoc.body;
        event.pageX = event.clientX +
          (doc && doc.scrollLeft ||
            body && body.scrollLeft || 0) -
          (doc && doc.clientLeft ||
            body && body.clientLeft || 0);
        event.pageY = event.clientY +
          (doc && doc.scrollTop ||
            body && body.scrollTop || 0) -
          (doc && doc.clientTop ||
            body && body.clientTop || 0 );
  }
  // get the x-coordinate of the user
  var userX = event.pageX -
    $('#toBeTracked').offset().left;
  // get the y coordinate of the user
  var userY = event.pageY -
    ($('#toBeTracked').offset().top + areaHeight);
}
```

Unity3D, a game engine popularly used by the game developers, also has some built-in ways to get the mouse coordinates [29]. They have a *mousePosition* method within the Unity's input system that returns a vector corresponding to the user's mouse position [30]. Unreal Engine, another popular game engine, also exposes a way to get the mouse position [31]. However, it should be noted that excessive mouse-tracking can affect the game performance in a negative way and could even crash the system. A high temporal resolution will consume a lot of computer memory. For example, it is less expensive to collect mouse tracking data every 250 ms as compared to 100 ms. Further, it depends on the target computer memory configuration as well. A game with mouse-tracking temporal resolution of 200 ms that works as expected on a given computer system may crash when deployed on a system with lower available memory as maintaining all of the data about the mouse position will add up quickly.

3.2.1.2 Emotion Tracking

The Visage|SDK [32] from Visage Technologies and Affdex [33] are two readily available software solutions that can be used to recognize human emotions by utilizing the Facial Action Coding System [34]. These solutions can be used for offline or real-time analysis of human faces. For offline analysis, a recording of the facial stimulus is required, while real-time analysis needs a Web camera integrated with the application that can send the facial stimulus as an input to the software. Affdex can detect various emotions and expression [35], and this output can then be used to detect if a person is bored or frustrated during the game play. As an example, Baron [10] used an algorithm to categorized the observed emotions into states of boredom, flow, and frustration using the following algorithm:

- If all the emotions are below the threshold, then the player is classified in a BORED state unless they were in a state of FLOW previously.
- If any of the emotions is above the threshold, then the player is in a non-bored state.
  - If anger is above the threshold and happiness is below the threshold, then the player is classified to be in a FRUSTRATION state.
  - If surprise is above the threshold and sadness is below the threshold, then the player is classified to be in a FLOW state.
- If the above rules fail, then the player is classified to be in a state called NONE.

In another example, [36] used Affdex output data to predict the states of boredom, flow, and frustration during game play using binary logistic regression. This prediction can be used to adapt the game or content difficulty in real time [1]. Their results suggested that it was easier to predict boredom and flow, compared to detecting frustration. Below are some of the prediction equations that their analysis revealed:

$$ln(Flow/NonFlow) = -0.84 + (0.4 \times Fear) + (0.09 \times Happiness) + (-0.074 \times Sadness)$$

$$ln(Boredom/NonBoredom) = -1.24 + (-1.13 \times Fear) + (-0.38 \times Happiness) + (0.15 \times Sadness)$$

$$ln(Frustration/NonFrustration) = 1.85 + (-0.02 \times Attention) + (-0.03 \times BrowFurrow) + (0.02 \times EyeClosure) + (-0.067 \times LipPress) + (-0.03 \times LipPucker) + (0.03 \times LipSuck)$$

3.2.1.3 Player Data-Tracking

A large amount of background data can be collected when a video game is played. Such data may include (but not limited to) time spent on task, total time played, player death count, player score, and quiz response. Any data that can be attributed to an observed variable can be gathered in a log file, which can be used for online or offline analysis. These variables can also be used in conjunction with each other for dynamic adaptation. For example, [1] used player score and their emotions to change the game difficulty in real time according to the following rules:

- Step up the game difficulty only when they have achieved 50% of max achievable score and are detected as being bored.
- Step down only when their score is less than 20% of max achievable score and are detected as being frustrated.
- If they're in Flow, don't change anything, and wait a while before checking again.

3.2.1.4 Bayesian Modeling

This approach utilizes the conditional dependence of several variables on each other, to create a probabilistic graphical model of the system. A Bayesian network for knowledge tracing is shown in Fig. 3.2 [13]. This model can be used to make inferences about the player learning based on the other observed variables. The model consists of a two-quiz sequence with four parameters called prior knowledge P(L), slip rate P(S), guess rate P(G), and learn rate P(T). Prior knowledge is gauged using diagnostic tests, while guess rate accounts for the probability of guessing despite not possessing the knowledge. Slip rate represents the probability of answering incorrectly (slipping) even when a skilled student actually knows the correct answer. The learn rate accounts for the probability that the learning will occur in the next level, based on the learning from the previous levels.

There are solutions available for modeling Bayesian networks in many programming languages. Bayes Server [37] is one such software used by Verma et al. [38] in their study. It has programming APIs available in `C#`, Python, Java, R, MATLAB, Excel, Apache Spark, and JavaScript. The `C#` API can be easily integrated to a Unity3D game program [39] for dynamically generating Bayesian networks, setting

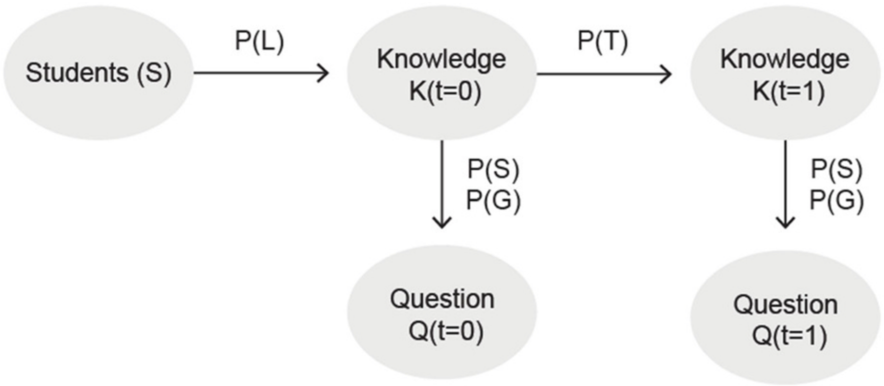


Fig. 3.2 Bayesian network depicting knowledge tracing model [13]

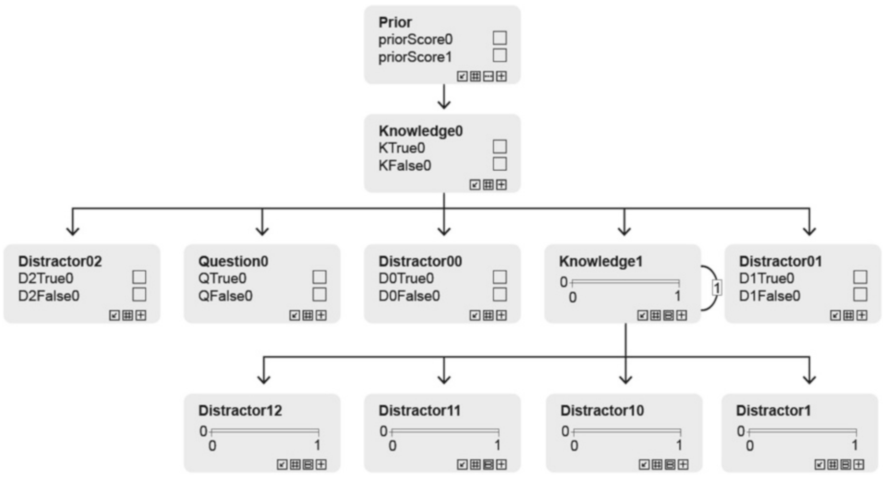


Fig. 3.3 Sample dynamic Bayesian network generated using Bayes Server

evidence as well as querying the network at any point in time. Figure 3.3 shows a dynamic Bayesian network that was programmed in Bayes Server.

3.2.1.5 Educational Data Mining

This technique uses a large number of data mining methods, which can be utilized to find out the patterns in bulk data captured during educational game play sessions [40]. It has been used for stealth [36, 41] as well as non-stealth measurements [42]. A given data mining technique applies to a given problem based on the assumptions and type of data. Therefore, the method should be carefully chosen based on the assumptions related to data.

3.3 Endogenous and Exogenous Games

The terms endogenous and exogenous games refer to how interconnected the game mechanics and the learning content are [8]. The more connected they are, the more the player both learns and enjoys the gaming experience [43]. Both levels of integration present their own problems from the perspective of game design, educational design, and game mechanics engineering.

3.3.1 Exogenous Games

The more exogenous a game is, the less of a connection there is between what the player is doing in the game and the content being taught [8]. Take as an example a math learning game where the player flies a spaceship through an asteroid field, destroying asteroids for points. At the end of the level, they are presented with a math quiz that must be completed in order to advance to the next level. This game would be considered completely exogenous. There is no connection between the game mechanics, flying the ship and shooting the asteroids, and the math content being taught. In particular, a player's ability to demonstrate skill at flying the ship and destroying the asteroids is not at all an indicator of how they will perform on the math quiz. Likewise, getting a high score on the math quiz does not in any way help the player fly the ship better on the next level [44].

Despite these drawbacks, exogenous games are more common in the classroom [44]. This is in part because they are easier from a design standpoint. When designing the game mechanics, the designers do not need to consider the educational content at all. These are often game designers who are focused and experienced at making entertaining gaming experiences, but often not at making educational content [10]. Likewise, the educational designers are experts at designing educational content and scaffolding the learning process, but are unfamiliar with what makes a quality gaming experience. With an exogenous game, these two parties can remain within their respective areas of expertise without needing to seek compromise with each other.

The impact exogenous design has on the actual construction of the game software is harder to judge [10]. When the game is truly exogenous, there is a need to develop two completely different experiences. The first is for the game itself and the mechanics that the player engages with, and the second is for the learning portion of the game. One benefit of this is that these types of educational experiences are often very straightforward, often taking the form of quizzes [44]. These then should be relatively simple to construct from a software perspective. The game mechanics in exogenous games are also often simpler, since they do not need to link with the educational content at all. This means that exogenous games require the time and effort to build two completely different gaming environments within the same project, with little space for overlap. However, these two pieces of the software are usually simpler to build than would often be required for a single combined code base.

3.3.2 Endogenous Games

The counterpart to exogenous games is endogenous games, where there is a strong connection between the game mechanics and the educational content [8]. This means that there is at least some connection between what the player does in the

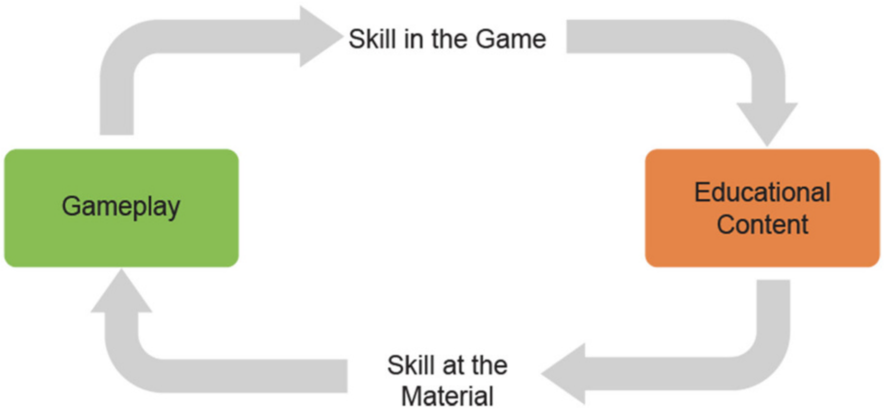


Fig. 3.4 Link between game mechanics and educational content

game and their skill in the educational content. Taking the previous example game, it is possible to make it more endogenous by adding two mechanics. The first would be to link skill at the math quizzes to the game play, and the second would be a way to reciprocate that link. Adding a mechanic where doing well on the math quiz would empower the player's spaceship would be a first example of a way to tie the educational content to the game play. The player's math skills would now have a direct effect on the game play. Likewise, adding a mechanic where destroying asteroids would earn the player hints on the math quizzes would be one possible way to link the game play to the quiz. With this mechanic in place, a player who does better in the spaceship portion of the game would be more likely to do well on the math quiz. This example is still not very endogenous [44], but this highlights a starting point for creating these links to create endogenous game mechanics. Figure 3.4 shows these links between the mechanics and educational content and how they need to feed into each other to create a cycle.

An example of a highly endogenous game [8] would be Typing of the Dead (Typing) by Sega [45]. In this game, players automatically walk between locations and are approached by zombies. Each zombie has a word displayed on them, and the player must type the word displayed, and their character will shoot the zombie. If the player is too slow, the zombie will damage them, and they will eventually die if this happens too many times in one level. In this case, the mechanics of how the player plays the game and what they are being taught are completely interlinked [44]. The goal of this game is to teach the player how to type and to increase their typing speed. The player must type quickly in order to avoid being killed in the game, showing a link between the game play and the learning content. At the same time, having a higher level of skill at typing will transfer to the player getting further in the game than another player with a lower level of skill would. This creates the ideal situation for an endogenous game where the player will learn the most effectively [43].

Endogenous games are less common than exogenous games, despite being more effective as teaching tools because they are harder to make for a variety of design reasons [44]. Many of the benefits that made exogenous games easy to design are reversed in this situation, making the design process much more difficult. If the game play and educational content are to be tightly linked, then it is necessary for the game designers and educational designers to work together. The game mechanics cannot be designed without consideration for the educational content, and likewise the educational content must take into account what the player will be doing in the game in order to decide how to present the content. Making these priorities line up with each other can take a considerable amount of time out of the development process.

The way in which endogenous design impacts the software structure is easier to judge than with exogenous games [10]. Due to their nature, exogenous games required the creation of two almost completely separate experiences with little overlap. Endogenous games by their nature then would instead consist of one larger integrated experience. The immediate benefit of this is that it removes the problem of creating almost two separate code structures with little overlap. The downside is that this one structure is likely much more complex than the other two would have been. Endogenous mechanics are often more complex due to the need to integrate the learning material directly. This both makes them more difficult to implement and also can force the way you design your high-level code structure. Having the educational content wrapped around the mechanics can make that material difficult to edit and possibly require mechanics changes as well. Learning assessment, bug tracing, and fixing can also be more complicated due to this connection. Endogenous design has been shown to be the better approach for teaching [43], but these issues make building them a challenge [10]. What would be best would be a way to create endogenous mechanics that are strongly linked in game play, but not in actual code. This is the principle which is followed in the CAGE architecture to develop the game rapidly while keeping the player engagement intact for learning purposes.

3.4 CAGE Architecture

The CAGE architecture can be used to develop games and assessments that can cater to multiple learning contents [9, 10]. It employs content-agnostic endogenous mechanics that are tied to game play and therefore allows creation of multiple games with an effort, which would be used to create a single game. CAGE follows a component-based architecture [10] shown in Fig. 3.5. It consists of three main components: the mechanics component, the content component, and the student model.

The mechanics component corresponds to the in-game mechanics, such as jumping, walking, sliding, or other actions players take in order to play and progress in the game. In CAGE, this component is designed to be content-agnostic and endogenic so that it can be used with multiple learning contents. The content

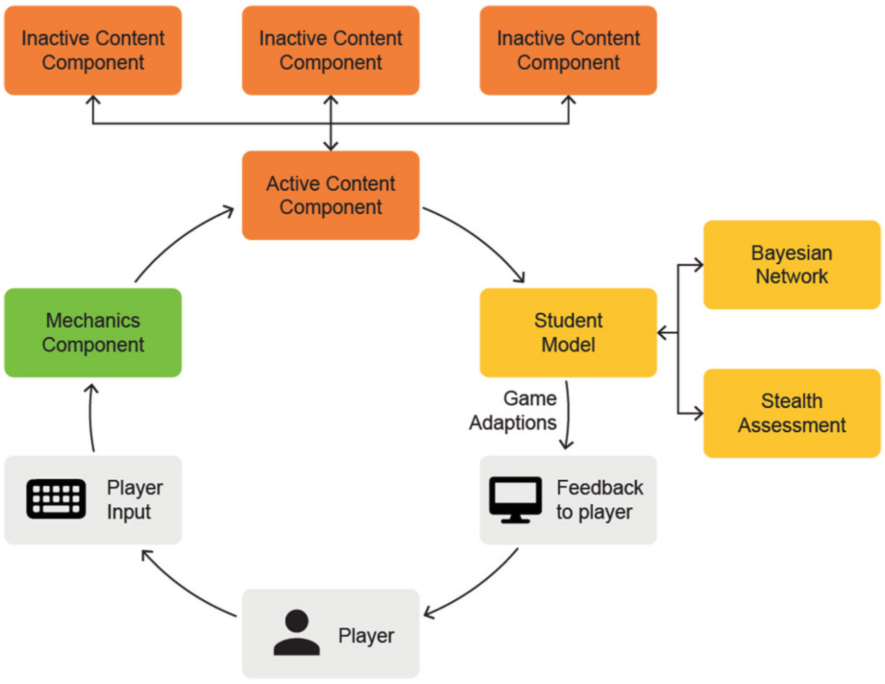


Fig. 3.5 Component-based CAGE architecture

component corresponds to one of the learning contents that the game is expected to teach. A CAGE game may consist of several learning components, but only one of them will be active at any point in time. The mechanic component passes the player action as input to the content component, which then evaluates the action as right or wrong. This assessment is then passed to the student model, which acts as an accumulator for the analysis to build a model of the student's understanding. The student model is also designed to be content-agnostic so that it can be re-used across multiple learning content domains, by containing multiple models to track understanding on the different topics.

The student model can be used for summative or formative analysis. As a summative analysis, it can be used to provide the play-through summary of the player, such as total time they took to learn the content, number of failures, success, scores, etc. As a formative analysis, it can be used to inform the game play session as well as the player feedback. When used for feedback, it can be used to provide hints or corrective actions to the players. When the student model is used to inform game play, it can make the game adjust dynamically based on the player's current skill level. This dynamic adaptation could be altering the game environment, the difficulty of the game or learning content, or all of these together.

There are several ways to make the game content-agnostic. In [10], one such process was using hooks for the game objects to communicate with each other.

Hooks are the generic messages that are generated by the game mechanics whenever an event occurs in the game, such as the player jumping across a chasm. The mechanics component passes these hooks to the content component, which uses them if they are relevant for that content component. Hooks is an abstract base class that implements polymorphism. Each content then implements their own version of the hooks, overriding only the methods that they need. Below is an example in C# in the Unity3D engine for a hook that corresponds to the player taking an action.

```
using UnityEngine;
using System.Collections;
/// <summary>
/// The hook used when you need to pass
/// actions to the content class or its derivatives
/// </summary>
public class ActionHook : Hook
{
    public string action { get; private set; }

    public ActionHook(string ac)
    {
        type = HookType.Action;
        action = ac;
    }

    public override string ToString()
    {
        return "Action performed: " + action;
    }
}
```

Another method to make the game content-agnostic is to use a database or JSON files to keep the content data. All the data for a particular learning content type will reside in their own JSON files. Players will be given the option to choose the learning content at the beginning of the game. When they later change the learning content, the active JSON files are swapped to use the one that the player selects. For example, consider the below piece of code showing two different content classes that inherit a common base class called Content. The source JSON file name is changed based on the content selected by the user.

```
public class Chemistry : Content
{
    public Chemistry(string name, string description,
    string jsonFile)
    {
        this.name = "Chemistry";
        this.description = "Balance the equations!";
        this.jsonFile = "Chemistry.json;
    }

}

public class Cryptography : Content {
    public Cryptography(string name, string description,
    string jsonFile)
    {
        this.name = "Cryptography";
        this.description = "Encode/decode the data!";
        this.jsonFile = "Cryptography.json";
    }
}
```

3.4.1 Student Model

A student model could be used to model various states of the student, such as their emotional state, gaming skill, and knowledge. It is an accumulator model that can collect the data from player interactions and mine it to store useful information about the student. This information could be used to gauge their behavior as well as learning and anything else that might be hidden from an external observer. Any stealth assessment technique discussed in Sect. 3.2.1 can be used to create a student model. These techniques can be combined with each other to create a more complex student model that can be used to inform about multiple aspects of a student. For example, it can be used to measure their game play proficiency and how it affects their learning performance governed by their cognitive and emotional states. This model can then be used to provide feedback and remediation to students. Additionally, it can be used to dynamically adapt the game play to create an optimal learning environment suitable for their needs.

3.4.2 Stealth Assessment in CAGE

As indicated in Sect. 3.4.1, various ways of stealth assessment can be implemented at once to create the desired assessment in the game and generate a student model. The code snippet below illustrates how the predictive equations from [36] were used with the algorithm implemented by Baron [10] to create a stealth assessment within a CAGE game.

```
private AffectiveStates determineState()
{
  AffectiveStates state = AffectiveStates.None;
  //probs is an array containing probability values
  //e.g. probs[ATTENTION] is probability of
  //player being in the state of attention
  float boredom = (float)(−8.44 + (0.07 *
        probs[ATTENTION]) +
        (0.02 * probs[BROWFURROW]) +
        (0.06 * probs[BROWRAISE]) +
        (0.02 * probs[INNERBROWRAISE]) −
        (0.028 * probs[MOUTHOPEN]) −
        (0.03 * probs[SMILE]));

  boredom = Mathf.Exp(boredom);
  boredom = boredom / (1 + boredom);

  float flow = (float)(1.5 − (0.02 *
    probs[ATTENTION]) − (0.025 * probs[EYECLOSURE]) −
        (0.037 * probs[INNERBROWRAISE]) +
        (0.02 * probs[LIPPUCKER]) −
        (0.02 * probs[LIPSUCK]) +
        (0.02 * probs[MOUTHOPEN]) +
        (0.08 * probs[SMILE]));

  flow = Mathf.Exp(flow);
  flow = flow / (1 + flow);

  float frustration = (float)(1.85 −
    (0.02 * probs[ATTENTION]) −
    (0.03 * probs[BROWFURROW]) +
        (0.02 * probs[EYECLOSURE]) −
        (0.067 * probs[LIPPRESS]) −
        (0.03 * probs[LIPPUCKER]) +
        (0.03 * probs[LIPSUCK]));

  frustration = Mathf.Exp(frustration);
  frustration = frustration / (1 + frustration);

  if (frustration > boredom)
  {
    if (frustration > flow)
      state = AffectiveStates.Frustration;
    else
```

```
        state = AffectiveStates.Flow;
    }
    else
    {
      if (boredom > flow)
        state = AffectiveStates.Boredom;
      else
        state = AffectiveStates.Flow;
    }
    return state;
}
```

3.5 Validation of the CAGE Framework

Baron [10] found that the code re-usability is a desired aspect of game creation process when developing multiple serious games, which is facilitated by the CAGE framework. This indicates that the CAGE framework was effective at creating multiple games for learning with the help of content-agnostic mechanics. He also found that the amount of programming required to create subsequent games from the code of the first game was drastically reduced. As a result, any further game creation needed lesser amount of time and effort when compared to creating the game for the first time [10]. This suggests that the CAGE framework was effective in the rapid development of educational games. However, the small number of participants for this study could limit the generalizability of findings. Further, the research was conducted in a classroom environment, and the participants were working alone instead of a team, to develop these games.

Verma [1] implemented the stealth assessment into the CAGE framework and used it to dynamically adapt the game. Verma et al. [36] found that the facial expression were a better predictor of the player states of boredom, flow, and frustration, as compared to the facial emotions. Therefore, they used facial expression to model the three states and subsequently used it to adapt the game play. However, in another study, they found that the game adaptation was only effective for players who had lower domain knowledge of the learning content [46]. For example, in a game that teaches chemical equation, the players who already are an expert in balancing chemical equations would not benefit from game adaptation. However, players who join the game play with a low prior knowledge about chemical equation balancing would benefit more. Therefore, the game adaptation should be designed primarily for the low-domain learners. A limitation of the finding was that the participants were not evenly distributed across the test and the control groups, which could have caused biased results.

Baron [10] found that it leads to reduced engagement when playing multiple CAGE games since they employ similar game play mechanics. However, game adaptation helped in alleviating the problem, and therefore player engagement was sustained across multiple CAGE game sessions [1]. Further, the adaptation helped improve player engagement when considered independent of the CAGE framework. The adaptation was implemented with the help of a student model built using dynamic Bayesian network. Therefore, it is recommended to create a dynamic game adaptation within the educational games to keep the player motivation levels intact. This study was conducted online during the pandemic and observed 35% dropout rate, which might have been caused by the issues in the UI or bugs in the game. Further, the effect size obtained in the results was low, and therefore, these results must be interpreted with caution.

Another experiment to establish the validity of the student model indicated that it can be applied in a content-agnostic way [38]. It involved comparing the inference from the embedded student model with an external assessment and found a significant correlation between the two. However, it depends on how the assessment is implemented in the game. Therefore, it is suggested to validate your own assessment before assuming its correctness. Nevertheless, the experiment shows that an assessment that has been designed to be content agnostic can be valid for multiple content domains.

References

1. Verma, V.: Content Agnostic Game Based Stealth Assessment. PhD thesis, Arizona State University (2021)
2. Shute, V., Spector, J.M.: Scorm 2.0 white paper: stealth assessment in virtual worlds. Unpublished manuscript (2008)
3. Kim, Y.J., Shute, V.: Opportunities and challenges in assessing and supporting creativity in video games. In: Video Games and Creativity, pp. 99–117. Elsevier, Amsterdam (2015)
4. Shute, V.: Stealth assessment in computer-based games to support learning. Comput. Games Instruct. **55**(2), 503–524 (2011)
5. Ventura, M., Shute, V., Small, M.: Assessing persistence in educational games. Des. Recommendations Adapt. Intell. Tutoring Syst. Learner Model. **2**(2014), 93–101 (2014)
6. Rotherham, A.J., Willingham, D.T.: 21st-century' skills. Am. Educ. **17**(1), 17–20 (2010)
7. Mislevy, R.J., Almond, R.G., Lukas, J.F.: A brief introduction to evidence-centered design. ETS Res. Rep. Ser. **2003**(1), i–29 (2003)
8. Malone, T.W., Lepper, M.R.: Making learning fun: a taxonomy of intrinsic motivations for learning. In: Aptitude, Learning, and Instruction, pp. 223–254. Routledge, Milton Park (2021)
9. Baron, T., Heath, C., Amresh, A.: Towards a context agnostic platform for design and assessment of educational games. In: European Conference on Games Based Learning, p. 34. Academic Conferences International Limited, Reading (2016)
10. Baron, T.: An Architecture for Designing Content Agnostic Game Mechanics for Educational Burst Games. PhD thesis, Arizona State University (2017)
11. Shute, V.J., Kim, Y.J.: Formative and stealth assessment. In: Handbook of Research on Educational Communications and Technology, pp. 311–321. Springer, Berlin (2014)
12. Verma, V., Baron, T., Bansal, A., Amresh, A.: Emerging practices in game-based assessment. In: Game-Based Assessment Revisited, pp. 327–346. Springer, Berlin (2019)
13. Pardos, Z.A., Heffernan, N.T.: Modeling individualization in a bayesian networks implementation of knowledge tracing. In: International Conference on User Modeling, Adaptation, and Personalization, pp. 255–266. Springer, Berlin (2010)

14. Chin, J., Dukes, R., Gamson, W.: Assessment in simulation and gaming: a review of the last 40 years. Simul. Gaming **40**(4), 553–568 (2009)
15. Shute, V., Masduki, I., Donmez, O., Dennen, V.P., Kim, Y.-J., Jeong, A.C., Wang, C.-Y.: Modeling, assessing, and supporting key competencies within game environments. In: Computer-Based Diagnostics and Systematic Analysis of Knowledge, pp. 281–309. Springer, Berlin (2010)
16. Shute, V., Wang, L.: Measuring problem solving skills in portal 2. In: E-Learning Systems, Environments and Approaches, pp. 11–24. Springer, Berlin (2015)
17. Mayer, I., van Dierendonck, D., Van Ruijven, T., Wenzler, I.: Stealth assessment of teams in a digital game environment. In: International Conference on Games and Learning Alliance, pp. 224–235. Springer, Berlin (2013)
18. Shute, V., Kim, Y.J.: Does playing the world of goo facilitate learning. In: Design Research on Learning and Thinking in Educational Settings: Enhancing Intellectual Growth and Functioning, pp. 359–387 (2011)
19. Chen, J.: Flow in games (and everything else). Commun. ACM **50**(4), 31–34 (2007)
20. Crisp, G.T., Assessment in next generation learning spaces. In: The Future of Learning and Teaching in Next Generation Learning Spaces. Emerald Group Publishing Limited, Bingley (2014)
21. Shute, V., Ventura, M., Small, M., Goldberg, B.: Modeling student competencies in video games using stealth assessment. Des. Recommendations Intell. Tutoring Syst. **1**, 141–152 (2013)
22. Papesh, M.H., Goldinger, S.D.: Memory in motion: movement dynamics reveal memory strength. Psychonomic Bull. Rev. **19**(5), 906–913 (2012)
23. Yamauchi, T., Xiao, K.: Reading emotion from mouse cursor motions: affective computing approach. Cognit. Sci. **42**(3), 771–819 (2018)
24. Freeman, J.B., Ambady, N.: Motions of the hand expose the partial and parallel activation of stereotypes. Psychol. Sci. **20**(10), 1183–1188 (2009)
25. Rheem, H., Verma, V., Becker, D.V.: Use of mouse-tracking method to measure cognitive load. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 62, pp. 1982–1986. SAGE Publications Sage CA, Los Angeles (2018)
26. Faulkenberry, T.J., Testing a direct mapping versus competition account of response dynamics in number comparison. J. Cognit. Psychol. **28**(7), 825–842 (2016)
27. Element: mousemove event (2022)
28. Mouseevent.pagex (2022)
29. Unity3d (2022)
30. Unity3d: Input.mouseposition (2022)
31. Unreal engine: get mouse position (2022)
32. Visage Technologies (2022)
33. Affectiva (2022)
34. Ekman, P., Friesen, W.V.: Facial Action Coding System: A Technique for the Measurement of Facial Movement, Consulting Psychologists Press. Palo Alto, Santa Clara (1978)
35. iMotions Inc. Affectiva channel explained (2018). https://help.imotions.com/hc/en-us/articles/360011728719-Affectiva-channel-explained. Accessed 07 Aug 2022
36. Verma, V., Rheem, H., Amresh, A., Craig, S.D., Bansal, A.: Predicting real-time affective states by modeling facial emotions captured during educational video game play. In: International Conference on Games and Learning Alliance, pp. 447–452. Springer, Berlin (2020)
37. BayesServer. Dynamic bayesian networks – an introduction (2022)
38. Verma, V., Amresh, A., Craig, S.D., Bansal, A.: Validity of a content agnostic game based stealth assessment. In: International Conference on Games and Learning Alliance, pp. 121–130. Springer, Berlin (2021)
39. BayesServer. Dynamic bayesian networks `c#` api in bayes server (2022)
40. Scheuer, O., McLaren, B.M.: Educational data mining. In: Encyclopedia of the Sciences of Learning, pp. 1075–1079 (2012)

41. Baker, R.S.J.D., Gowda, S., Wixon, M., Kalka, J., Wagner, A., Salvi, A., Aleven, V., Kusbit, G., Ocumpaugh, J., Rossi, L.: Sensor-free automated detection of affect in a cognitive tutor for algebra. In: Educational Data Mining 2012 (2012)
42. D'Mello, S.K., Graesser, A.: Mining bodily patterns of affective experience during learning. In: Educational data mining 2010 (2010)
43. Mislevy, R.J., Oranje, A., Bauer, M.I., von Davier, A.A., Hao, J.: Psychometric Considerations in Game-Based Assessment. GlassLabGames (2014)
44. Shaffer, D.W., Squire, K.R., Halverson, R., Gee, J.P.: Video games and the future of learning. Phi delta kappan **87**(2), 105–111 (2005)
45. Typing of the dead, the description (2022)
46. Verma, V., Craig, S.D., Levy, R., Bansal, A., Amresh, A.: Domain knowledge and adaptive serious games: exploring the relationship of learner ability and affect adaptability. J. Educ. Comput. Res. **60**(2), 406–432 (2022)

Chapter 4
Performance on Software Architecture Design to Serious Games for Mobile Devices

Leticia Davila-Nicanor, Irene Aguilar Juarez, Joel Ayala de la Vega, Abraham Banda Madrid, and Sochitl Cruz López

Abstract *Proposal*: This proposal has considered techniques to improve the software architecture performance in serious games. To validate and quantify the design approach have integrated the software architecture evaluation by design quality attributes complexity and coupling.

Design: Memory data handling on mobile devices is limited; this situation affects efficiency and slows interaction mechanisms of learning environments. In the software process, design patterns are a technique to solve this problem; the use of these in software architecture allows for the improvement of the distribution of device resources: memory and fast processing set objects at runtime.

Findings: The proposal describes a technique to perform and validate the design architecture; the advantage of evaluating the system in early phases like design is cost reduction to remove defects and better the software performance.

Limitations: The presented work has focused on the construction and evaluation of the quality of the software system; however, the aspects of pedagogical evaluation belong to another study.

Practical implications and value addition: If software architecture design improves, then the learning process also improves. In order to better the performance design, Wrapper, Singleton, and MVC are implemented. Quality evaluation is through software architecture analysis through graph theory and software metrics, the metrics of the resulting system architecture. The dispersion diagram shows us an architecture with acceptable quality levels.

L. Davila-Nicanor (✉) · A. B. Madrid
Laboratorio de Evaluación y Calidad de Software, Centro Universitario UAEMex Valle de México, Boulevard Universitario s/n. Predio de San Javier, Atizapán de Zaragoza, Estado de México, Mexico
e-mail: ldavilan@uaemex.mx; abandam@uaemex.mx

I. A. Juarez · J. A. de la Vega · S. C. López
Centro Universitario UAEMex Texcoco, Av. Jardín Zumpango s/n. Fraccionamiento El Tejocote, Texcoco, Estado de México, Mexico
e-mail: iaguilarj@uaemex.mx; jayalad@uaemex.mx

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_4

Keywords Serious game · Architectural performance · Design patterns · Design software metrics

4.1 Introduction

In the new normality reached by the Covid-19 pandemic, virtual environments use serious games focused on education at all levels. Serious games have been applied in a wide range of scenarios to improve learning and reaction mechanisms, for instance, fires and earthquakes, and this approach has been shown to statistically contribute to better decision-making [1–5]. In education research, the positive impact of serious games on training and education has been studied in several research papers [6–10]. Nowadays, serious game use is important to improve learning. The studies analyzed by [7–9] show how serious games have increased learning in many areas like computer science.

In serious games development, the collaboration between areas of knowledge is intrinsic and pedagogical, and software engineering experts participate in the software development process. From the definition of requirements, the quality attributes are specified, and this depends on the purpose of the system to be developed. In serious games, the *performance* is related to the learning process [10–12]; if the game manages to keep the attention of the player, the learning process improves. However, if the game is slow, loses the score, and does not update efficiently, the player's attention is lost, and implicit gamification is affected, so the learning process cannot mature [13].

Programming object-oriented approach and design patterns are a resource-optimizing technique because through a holistic vision, the performance of applications is improved, optimizing resources, for instance memory. After all, only executed functions are loaded at runtime, which differs from other programming paradigms that load all the system functions without being sure they will be used [14–16]. This scheme is adequate in a dynamic and random context like serious games because it is possible to build runtime scenarios depending on player preferences and game context variables.

The proposed approach addresses the following four research questions as the main contributions:

Do the *design patterns* contribute to improving the performance of software architecture on mobile serious games?

How does the software development process integrate *design patterns* on software architecture in mobile serious games?

Do *design patterns* contribute to improving the quality attributes complexity and coupling on software architecture to serious games on mobile applications?

Are software design architecture's complexity and coupling valid to determine an indirect study of the efficiency of software for serious games on mobile devices?

Answers to these research questions will contribute to improving studies and software development techniques on the performance of design architecture in serious games. This proposal has considered techniques to improve the software architecture performance and consequently the learning process, so the Wrapper, Singleton, and MVC design software patterns are implemented. To validate and quantify the design approach have integrated the software architecture evaluation by design quality attributes complexity and coupling, so we have implemented an evaluation process for architectural design described in [17]. This approach is applied to the case study of professional-level subjects.

This chapter has been organized as follows: Sect. 4.2 is about the didactic methodology used for the design proposed. Section 4.3 depicts the works that relate to a recent review of the literature where the needs that serious games have about architectural design are highlighted, as well as the analysis of design metrics proposed to date. In Sect. 4.4, the proposal is developed integrating design patterns on software architecture and their evaluation. In Sect. 4.5, the discussion of the results is presented, and finally, in Sect. 4.6, the conclusions are presented.

4.2 Motivation and Research

The serious game is based on information theory [18], which is a model that explains learning as a model analogous to the information processing of a computer in which there are temporary and permanent information storage units, as well as devices to capture, search, produce, and transform information. Under this approach, learning is understood as the process of incorporating new learning into memory and recovering and using it.

According to Benzanilla, J. M. et al. [19], the structural components definition of a serious game for the design of the formal model is as follows:

Objectives: they must be clearly defined and known by the player. In the context of a serious educational game, the objectives will be explicit in the competitions performed.

Rules: This component will determine the order, rights, and responsibilities of the players, as well as the objectives to be met by each player to achieve the challenge they face.

Challenge: Determines when the game ends. The player will face problems related to learning data structures for which solutions will be sought and, once all is resolved, will face the challenge. The endgame criteria, both partial and general, will be specified in the learning outcomes for the proposed serious game.

Interaction: It is the component that arises from the mechanics and dynamics of the game, which will give rise to all the experiences that the player will enjoy. These will continually surface because of the game's immediate feedback, which will reflect evidence of progress toward the final challenge.

Gu, S. M. et al. [20] approach a system like a pair (U, A), where U = {x_1, x_2,..., x_n} is a finite and non-empty set of objects called the universe of discourse, which in the case of serious games can be taken as the rules of operation, and A = {a_1, a_2,..., a_m} is a non-empty finite set of attributes, which in this case are the challenges, so that a: U → It goes for any ∈ A. That is, a software system fulfills its purpose based on its attributes. In serious game software design, the biggest problem is focused on the scenarios set that can be expected in the scope of the system to establish them, which has to do with the number of variables that intervene in the context and the number of possibilities to whom it is addressed, which makes biggest option set, which also can only be specified until the moment when the players select their arguments. A multi-scale software system is needed to represent data sets with hierarchical-scale structures measured at different levels of granularity at which each challenge and player interacts.

4.3 Background and Recent Review of the Literature

Software architectures set the necessary components that a software system must have, based on its functional requirements. The main goal is to reach the best interaction between components, so software architecture has been defined as "the fundamental organization of a system, embodied in its components, their relationships to each other, and the environment, and the principles and guidelines governing its design and evolution over time" [21]. Non-functional requirements are also considered because they are related to quality attributes, for example, complexity, coupling, performance, reusability, cohesion, and reliability [22]. Regarding software architectures for the design of serious games, there are a few contributions. In Mizutani's research [23], the authors reviewed about 512 studies from 3 publishers of prestige, and under reuse criteria, they found only 36. This study found the approach based on *data-driven design* is, by a considerable margin, the most common, present in 45% of the studies. In minor frequency, other practices like *entities based on inheritance*, *layered systems*, and *design patterns* use are present in between 20% and 30% of the selected studies. The highlighted aspect's study is the relative absence of *test-driven development*. Also, *design patterns* are rarely seen applied in studies on the development of digital game mechanics. These patterns are currently a technique that has shown completeness and robustness in the domain of applications that implement them. The study authors expressed concern regarding what they consider to be a lack of community interest in software engineering to apply their knowledge to the context of serious game design, development, and testing and how much you could benefit from your asportation in these applications.

Regarding the performance of the application in the quality evaluation of the serious game, the metric reported in [6] work is the performance rate. Efficiency is a quality attribute related to the computer equipment's resources, memory, and processing speed, at runtime faster response times, are expected in software systems. It is an important quality attribute in serious games, this is because several studies

[10–12], there are set a direct relationship between attention and improvement in learning. On the contrary, a slow game generates disinterest and a lack of concentration, which inversely affects the learning process. Efficiency refers to the ability of a software system to do its job quickly and effectively, without consuming too many resources, such as memory or processor power. On the other hand, complexity refers to the number of components and the interconnection of these components in a system. Generally speaking, the greater the complexity of a software system, the more resources may be required to maintain its efficiency [24, 25]. For example, a simple software system that performs a specific task can be very efficient because it is easy to understand and maintain. But a complex software system that has many components and dependencies may be less efficient, as it may require more resources to maintain and function properly. In summary, while complexity is not an obstacle to efficiency, it can increase the need for resources and time to maintain the efficiency of a software system. Therefore, in the development of object-oriented software systems, it is important to find a balance between complexity and efficiency to develop a system that is effective and easy to maintain and uses available resources efficiently [26].

Design patterns are a good technique to solve the efficiency of software architecture [16]. The use of a design pattern abstracts and identifies key aspects in the solution of a highly complex problem. The kind of patterns is structural, behavioral, and creation, some of these are applied to set new functionality at runtime, having they limited only by the size of the memory of the equipment where they are executed. Researchers from [14] and [15] analyze how pattern design gives a better software solution in architecture to solve functionality. There are patterns to abstract and solve problems that are repeated daily in the design of software systems.

4.3.1 Metrics to Evaluate Architectural Software Design

To determine software architectural quality, software metric computation is widely used [27]. The evaluation of software architectures is a different process from the testing phase; the evaluation of the design involves data about the relations of the components, class, and the method's software system at rest, without system execution. In this case, the inputs are algorithms, class diagrams, diagrams of flow, etc. The advantage of evaluating the system in early phases like design is *cost reduction to remove defects*. According to the Carnegie and Mellon University study [28], if the evaluation is carried out in the testing phase, it is 12 times more expensive, but if software evaluation is done during start-up, it is 20 times more expensive than in the design phase.

There exist many metrics that measure the complexity of software: The cyclomatic complexity metric provides a means of quantifying intra-modular software complexity, and its utility has been suggested in the software development and testing process. This work [29] proposes to measure complexity, which is based on cyclomatic *complexity* and the concept of interaction between modules through

Table 4.1 Relationship between design attributes and software metrics

Study	Authors	Quality attributes	Metrics used
Investigating object-oriented design metrics to predict fault-proneness of software modules	Santosh Sigh Rathore, 2012 [31]	Size, cohesion, coupling, complexity, and inheritance	CBO (Coupling Between Objects), RFC (Response For a Class), LCOM (Lack of Cohesion in Methods), CAM, DIT (Depth of Inheritance Tree), NOC (Number of Children), LOC (Line Of Code), WMC (Weighted Methods per Class), CC (Cyclomatic Complexity).
Coupling and cohesion metrics in Java for adaptive reusability risk reduction	M.Iyapparaja, 2012 [32]	Cohesion and coupling	EV (Explicit dependence), IV (Implicit dependence)
The prediction design quality of object-oriented software using UML diagrams	Vibhash Yadav, 2013 [33]	Size, cohesion, coupling, complexity, inheritance, and abstraction	CC, LCOM, WMC, LOC
Predict fault-prone classes using the complexity of UML class diagram	Halim, 2013 [34]	Complexity	CC, RC (reduced complexity), NC (Nick's class)

coupling. In this article [30], multidimensional metrics are identified and defined; in this case, complexity is one of the properties that are related to performance and efficiency, applied to health monitoring models at the system level with specific phases of the design. Under these conditions, attributes like performance could be assessed indirectly through, for instance, complexity and *coupling* [29].

Design software metrics have been developed to obtain information about the quality design of an object-oriented software application, which aims to quantitatively describe a system's design properties. Table 4.1 shows the studies that address the design approach based on the design attributes studied, about the metrics evaluated. It is possible to observe that the most used properties or attributes are complexity, coupling, cohesion, and inheritance. The metrics CC [31–35], CBO [31, 32, 34, 35], and WMC [31–33] are the metrics with the highest level of acceptance for the realization of the studies, followed by LCOM and LOC [31, 33].

4.4 Proposed Solution

Benzanilla, J. M. et al. [19] set the structural components for the design of the formal model of the game, which are objectives, rules, challenges, and interactions. In this case, the biggest problem is that the set of scenarios that can be expected in the scope of the system, to establish the challenges, has to do with the number of variables that intervene in the context and the number of possibilities to whom it is addressed, which makes for an infinite set of options, which can also only be specified up to the point where players interact with the game.

4.4.1 Didactic Requirements

The application is aimed at computer engineering students to acquire knowledge of traversing trees in dynamic data structures, through practical exercises in which they will be graded according to the score acquired. Any engineering and computer science student who wants to reinforce their knowledge through this application. The player must have previous knowledge of basic programming.

Table 4.2 detailed the functional requirements, and the application consists of three levels of interaction: beginner, intermediate, and advanced. According to this level, the theoretical content is related. The first level is the beginner, and the problems that arise are a function of the theoretical framework that corresponds to this level. Scenarios (games) are established according to the answers if they are correct, and incentives are obtained, concluding the game, and it is assumed that he already has the master of the said topic, promoting level promotions.

Hardware specification for development: AMD 9 processor, 8G RAM, 500G hard drive, RANDOM 5 video card.

Table 4.2 Serious game functional requirements

No.	Functional requirements
R1	The main goal of the application is made to solve practical exercises
R2	The application will have content about the theoretical foundations
R3	The application will have content about the use of game explanation, rules, levels, and scenarios of the game
R4	The game consists of three levels (beginner, intermediate, and advanced)
R5	A new gamer will be at a beginner level
R6	Nothing is saved for the player in the test game, nor are incentives given
R7	To propose problems of the topic
R8	Rating of the solution to see the score and go to another level
R9	At each level of the game, the score acquired is saved
R10	The incentives are awarded according to the grade of the exercises on the topic

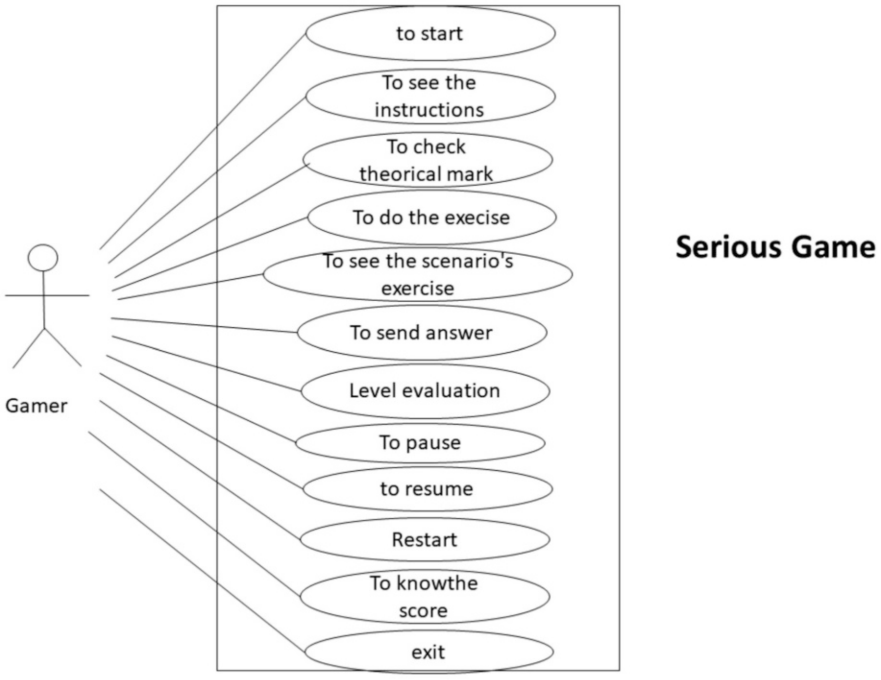


Fig. 4.1 Serious game functionality

Software specification for development: Windows 7, android-SDK-24-4-1-en-win, android-studio-ide-171.4443003-windows.

Specification for application: Have a smartphone that has an Android 4.2 (Jellybean) operating system, with 4G storage and 1G RAM.

The general functionality is shown in Fig. 4.1; the gamer (student) can select the play button, which shows him a screen where his initial score is presented, and the student can continue with the previous game or start a new game The serious game has three levels, before presenting the game scenario; first information about the topic to be evaluated is presented, and then the game scenario is presented, where several exercises accumulate points if they are solved satisfactorily. The player can pause the game, start a new game, or quit. In the case of just pausing the game, the accumulated points are saved. In other cases, the score is lost.

4.4.2 Didactic Design

In educational applications, defining the most appropriate pedagogical aspects for each project from the beginning is too important. In this case, the didactic design is specified by employing a descriptive letter of the didactic interaction in which the content of the game that is documented is specified. This application consists of three levels. In level 1 beginners, all the basic concepts of binary trees are addressed, which are necessary for the understanding of level 2 medium, where the path of binary trees is explained, so with this knowledge, you can go to level 3 advanced, and when you pass it, you can be evaluated as a student who masters the subject of binary trees in the data structure as shown in Table 4.3, the descriptive card of the game.

As a complement to the didactic letter, the navigation tree has been developed, using which it is possible to observe the proposed navigation for the player in the serious game. In Fig. 4.2, the player's activity in the three proposed levels is presented to accord letter of didactic interaction.

The serious game has been implemented in Android language. Design patterns are a technique that has demonstrated efficacy and reliability. The use of these in the software architecture allows for improvement of the distribution of device resources: memory and fast processing, the approach of setting set the needed objects at runtime. Design patterns have been successfully implemented in the architecture of our case studies; a serious game has been developed for the teaching of binary trees for the data structures subject of the computer engineering career. The Wrapper, Singleton, and MVC patterns are used in this work. They consider the functionality of the application. Better system evolution is another observed advantage of this approach; it is necessary throughout its useful life within the teaching-learning process.

In the proposed software architecture (Fig. 4.3), the *Wrapper pattern* allows to dynamically add functionality to the object to establish the scenarios, which allows only the base objects to be established in memory, and the pattern generates new combinations in the functionality in each new scenario so that the user has different views in each new game. Previous scenarios are dynamically removed from memory in this scheme. The *MVC pattern* (Model-View-Controller) allows for the separation of the operation of the user interface, the database, and the iteration between both; in our case, the database used was SQLite. The view shows the set of tools with which the player interacts, and the controller is responsible for communicating the *View_scenarios* actions and data where the game exercises are to the Wrapper pattern. Finally, certain global variables need to be kept in memory. When dealing with a dynamic schema, all the objects generated in memory will be eliminated except those that are handled by the singleton pattern. This scheme has allowed us to improve response times. The relationship of usability has been taken into account in studies that mark the most appropriate colors and texts [25].

When an object of type *Levels* is generated by the Exercise, in beginner level assigned zero points at the gamer through the *Assign_exercise*() method, according to Fig. 4.4. The gamer activity is evaluated by recording its responses, through the *Evaluate*() method, according to this action, the score is established, if the score's value is greater than 90%, a level rise can be granted through the *to_next_level*() method, this scheme operates when going from beginner to intermediate and from intermediate to advance via the *go_next_level*() method. When the player wants to pause, the *pause*() method is activated; if the gamer wants to leave temporarily and

Table 4.3 Descriptive letter of didactic interaction

Descriptive letter of didactic interaction					
Name of course: Data Structures		Duration: Variable		Modality: offline	
Instructor name:		Prerequisites: To the knowledge of basic programming			
Level 1 Beginner: Basic concepts of binary trees in data structure					
Topic	Content	Objective The participant:	Technique: Instructional	Activity: Instructor	Interaction: Mobile device
Rules game description	Game instructions	The student will know in a general way the composition of the game, the levels that it includes, and the learning that it will achieve	Through an explanation, the objectives of the game that will be used will be described to the student	The game allows interaction with the game rules description module	1. The student through the selection of options discovers the rules of the game 2. The application shows the academic content of each level and the challenges to solve 3. The student can listen to the general explanation of the game
Challenge Basic concepts	General concepts of trees: root, node, path, level, degree	Use the problem-based learning technique	The game will allow the student to listen to a reading of the basic concepts of binary trees The game provides questions related to the objectives of the problem	The application shows a reading with basic concepts	1. To show the student a fragmented image related to learning basic concepts 2. Randomly, the student is shown questions that, upon answering correctly, are given the corresponding image and he visualizes his puzzle little by little solved

Level 2 Intermediate: Binary tree traversal					
Rules' game presentation	Rules' game explanation	The student will know the rules of the game of level 2	Through the competence-based learning technique, through the explanation of the "rules of the game," the student will be able to solve the exercises by acquiring knowledge of the subject, based on their abilities, skills, and attitudes	The game provides audio where the rules of the game are explained	The game consists of an options menu where the player can select the instruction option where the student can listen to the general explanation of the game The game provides a video for each traversal where it explains the corresponding algorithm to traverse a binary tree
Binary tree traversal challenges	Tree traversal: Preorder, inorder, and postorder	The player will solve the binary tree traversal exercises provided by the application, thus being able to traverse the trajectory indicated in the game and go to the next level	It will be carried out with the competency-based learning technique	The game will provide a video with an explanation of each one of the routes that are carried out in the binary trees The application will provide the problems that the player will have to solve	The student will watch the video to learn how each of the routes of the tree is carried out, where each route that is correctly guessed will advance one square in the maze and will go to the next level once it reaches the goal square

(continued)

Table 4.3 (continued)

Descriptive letter of didactic interaction					
Name of course: Data Structures		Duration: Variable		Modality: offline	
Instructor name:		Prerequisites: To the knowledge of basic programming			
Level 3 Advanced: Balanced trees in data structures					
Know the rules of the game	Introduce the way of operating the game	The student will know the rules of the game	The discovery-based learning technique will be applied to solve the balanced tree exercises	The application will provide the rules of the game with audio explaining the goals that the player must meet as a measure of the knowledge obtained on the subject balanced trees in the data structure	The game describes the rules of the game
Balanced trees challenge	Balanced trees	The student will be able to acquire knowledge through the game; he will be able to discover the key and open the treasure chest, to solve the balanced tree exercises	Once the rules of the game are known, the learning technique based on discovery will be applied. The student must discover the knowledge and, in this way, guide him to the construction of his schemes. In this way, it organizes the information and relates it to previous knowledge	The game will provide a video of the academic knowledge of the subject so that the student can solve the exercises that the application will give him	The application shows a video with the content of the theme of the balanced tree; then it gives you the questions that you have to solve correctly to get a key; the more keys you have, the greater the probability of opening the treasure chest

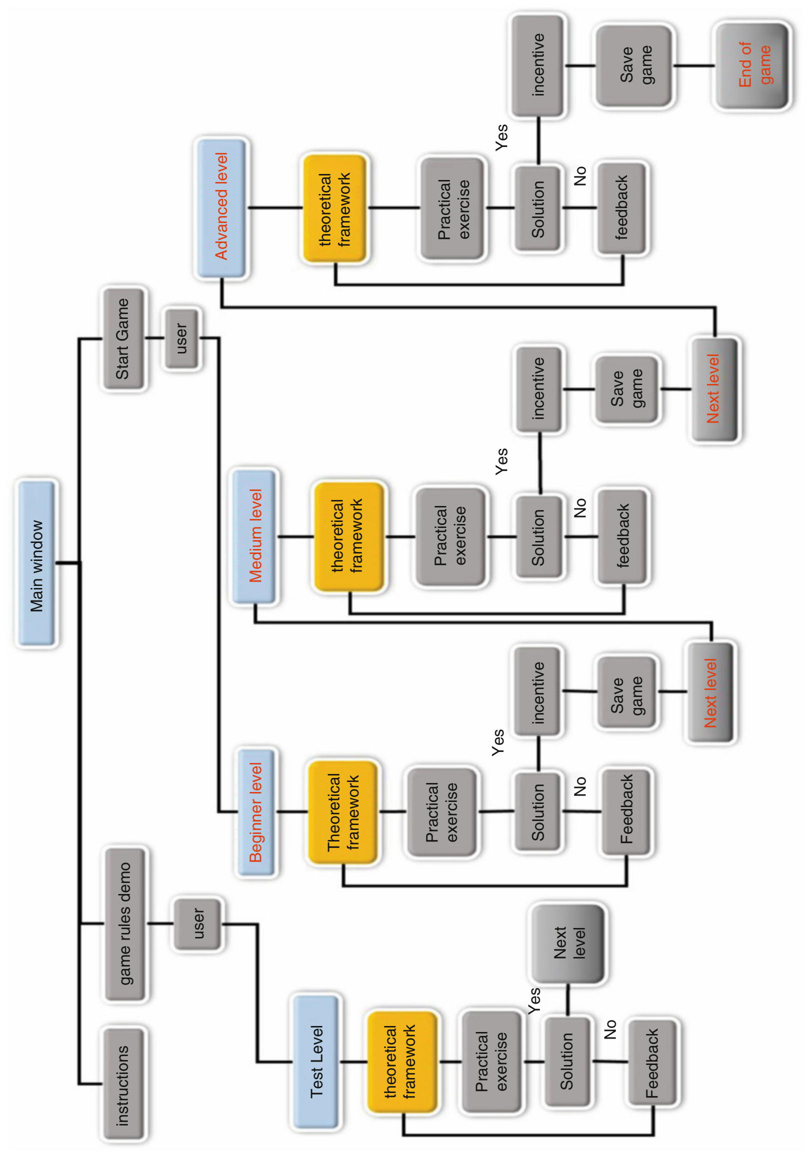


Fig. 4.2 Serious game navigation proposal

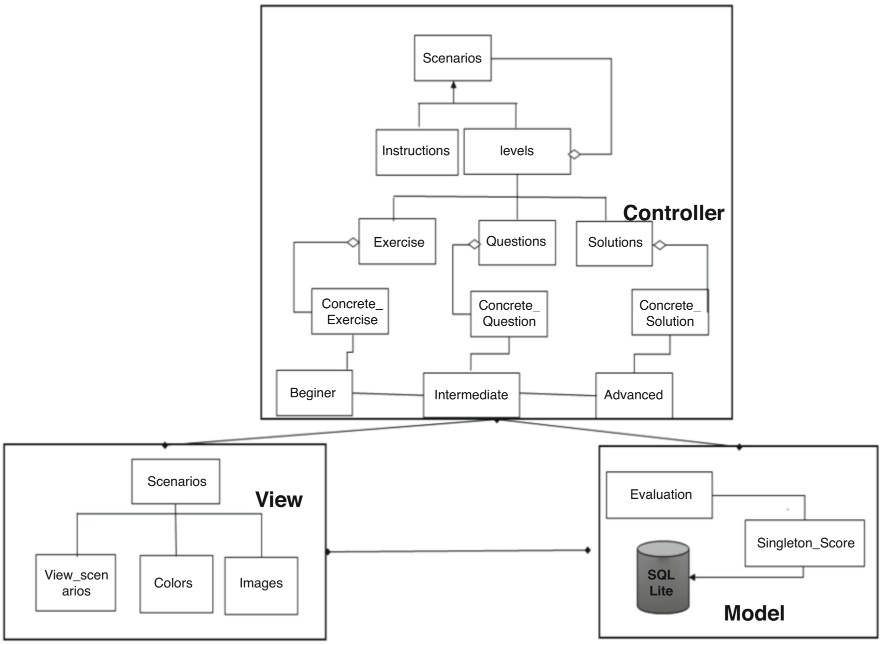


Fig. 4.3 Serious game proposal class diagram

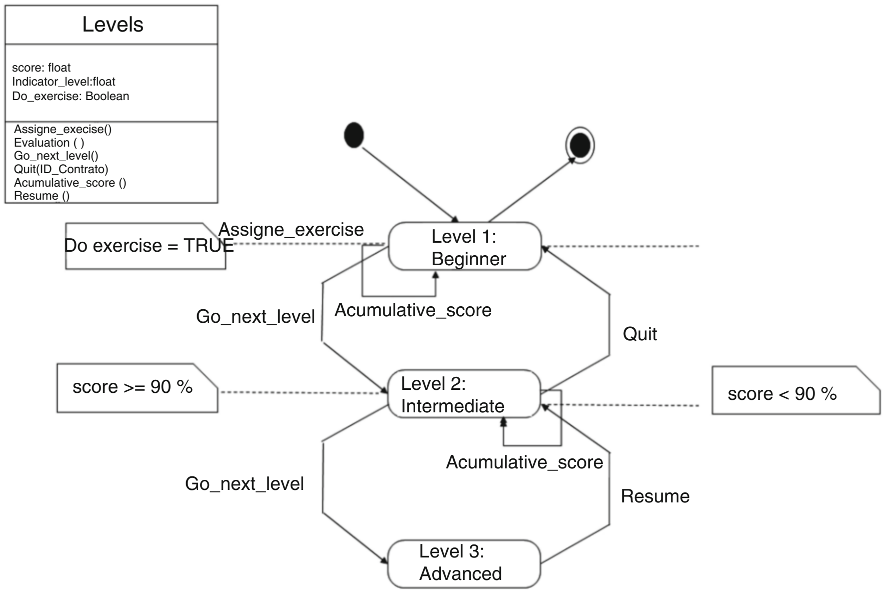


Fig. 4.4 State diagram levels class

then continue, his score is stored, and when he enters, he can *resume* from the level reached. Finally, if the gamer wants *to cancel* the game, the score is lost, and the game ends.

4.4.3 Serious Game Implementation

The serious game has been implemented in Android language. Figure 4.5 shows start-up interfaces and the beginner level. Regarding the usability of the game, the color chart has been selected that specifies the activity that motivates each of the colors according to [25]. For example, the blue color predominates, which generates calm and relaxation, which benefits the student to have concentration from the psychological didactic point of view. The green and yellow colors help the players in their retention and with greater ease, the theoretical contents of the game, to have more fluidity in the solution of the problems that each level of the game has.

Experimental Research on Design Evaluation
We have considered the proposal [17] to evaluate the quality architectural design of the proposed system. The proposal provides a predictive model to establish a quality scheme to assess the quality of oriented object systems based on the design stage using the software architecture analysis through graph theory and software metrics

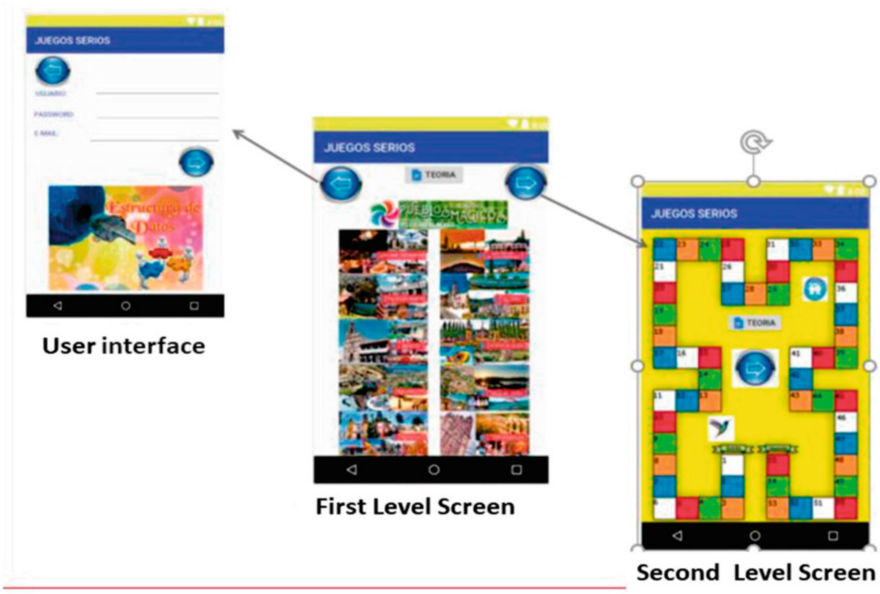


Fig. 4.5 Serious game interfaces

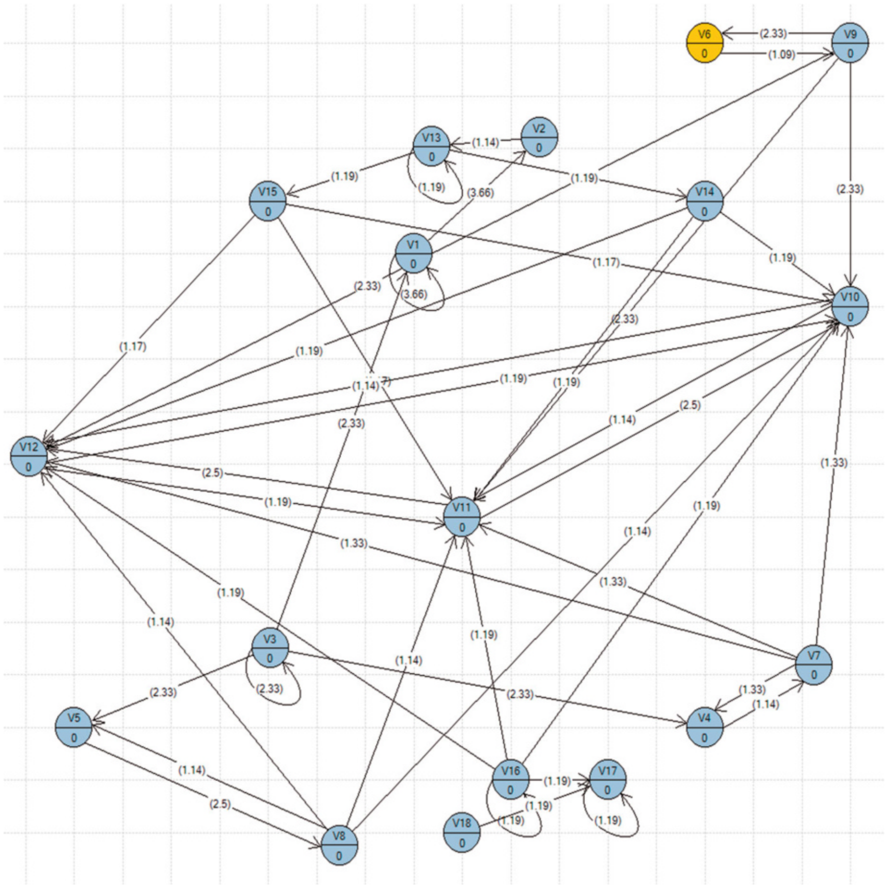


Fig. 4.6 ACG serious game proposal

of complexity and coupling. Based on the proposal, the following steps have been established:

Generation of the ACG. The architectural design of the system has been represented in Figs. 4.2, 4.3, and 4.4, so 18 components integrated the serious game, and the UML diagrams were the data gathering entry to establish the graph ACG_{SG} (see Fig. 4.6). The relationship between the vertices of the graph and the components of the system is presented in Table 4.4. The first column has the name of the vertex that relates to the component, and in the following columns, they are the complexity and coupling components metrics. The plug-in CodePro Analytix [36] was used to get the CCM and CBO metrics on each component.

Setting the Complex attribute on ACG. The CCM metrics are ACG's weight, so the Floyd-Warshall algorithm was running, and 75 critical paths were estimated, thus the less complex paths are discovered at the beginning and the more complex

Table 4.4 The relationship between the vertexes of the ACG_{SG} and the components of the system

Vertex	Component	CCM	CBO
v1	Scenarios	3.66	2
v2	Instructions	1.14	3
v3	Levels	2.33	1
v4	Exercise	1.33	2
v5	Questions	2.5	1
v6	Solutions	1.03	2
v7	Concrete_Exercise	1.14	3
v8	Concrete_Question	1.14	3
v9	Concrete_Solution	2.33	1
v10	Beginner	1.14	3
v11	Intermediate	2.5	1
v12	Advanced	1.19	2
v13	Scenarios_view	1.29	3
v14	View_scenarios	1.19	2
v15	Colors	1.17	3
v16	Images	1.19	2
v17	Evaluation	1.19	2
v18	Singleton_Score	1.19	2

paths at the end, these are later, according to the projected functionality, that we designated critical paths, these are the ones that reflect the functionality of the system embedded in the entire design.

Setting the CBO metric on paths localized. Following the proposal, to set *quality factors* in critical paths localized, the Spearman correlation coefficient has been estimated with the CBO data group and CCM data group in each path localized previously. Some results of the paths identified are shown in Table 4.5; in the first column are presented the identification path; in the next column, the CCM and CBO metrics on the sequence are shown, and the last column is presented the quality factor results. To set quality factors, Spearman's correlation has evaluated the relationship between the CCM and the CBO as ordinal variables. The results obtained have shown spectrum values between -1 and $+1$. If the value closes to -1, it indicated a weak relationship between complexity and coupling, so it was set at a low-quality level on the path evaluated. The quality parameter values close to $+1$ have a stronger relation, and the quality is higher. The plot on the dispersion quality parameter of Fig. 4.7 shows us a dispersion set has a stronger relationship. From these results, it was possible to infer the quality of the system was acceptable according to the attributes that were evaluated.

In Table 4.5, the results are presented. The first column has the ID of the critical path, the second the components sequence and their CMM and CBO metrics values, and finally the last the quality parameter. In the table, only some results are presented as an example. The total results of the 71 critical paths are the input data of the scatter dispersion plot in Fig. 4.7.

Table 4.5 Set the critical paths evaluated through quality parameters on serious game

ID path	Sequence					Quality parameter
1	v1	v2	v13	v15	v10	1
CCM	3.66	1.14	1.29	1.17	1.14	
CBO	2	3	3	3	3	
2	v1	v2	v13	v15	v11	0.6488
CCM	3.66	1.14	1.29	1.17	2.5	
CBO	2	3	3	3	1	
3	v1	v2	v13	v15	v12	0.2236
CCM	3.66	1.14	1.29	1.17	1.19	
CBO	2	3	3	3	2	
4	v1	v2				0.8750
CCM	3.66	1.14				
CBO	2	3				
5	v1	v2	v13			0.6488
CCM	3.66	1.14	1.29			
CBO	2	3	3			
n	v1	...	...	...	vn	Quality_factor
CCM	...	...	...	...	...	
CBO	...	...	...	...	...	
71	v18	v17				1
CCM	1.19	1.19				
CBO	2	2				

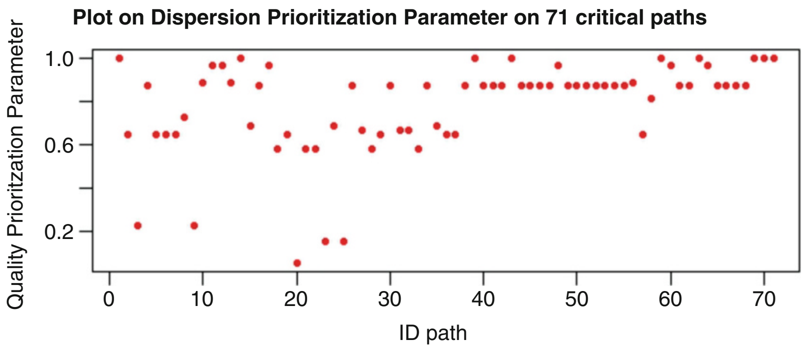


Fig. 4.7 Plot on dispersion quality parameter on serious game proposal

4.5 Discussion

In this proposal, the use of design patterns has been contemplated to improve the software architecture performance. Their holistic vision establishes an organization that requires optimization of the design, and the approach contemplates dynamic objects in memory generated at runtime [14–16]. This approach differs from others, where all the functions that the system can execute are loaded into memory, which in the case of serious games on mobile devices slows down the game dynamics.

To validate and quantify the design approach have integrated the software architecture evaluation method by design quality attributes complexity and coupling. The [17] proposal has been selected because the analysis has as its main axis a complexity approach strengthened with coupling. This relationship is related to efficiency [29–31], and the latter is another attribute that is desired in application performance. In the design stage, several studies are agreeing on the metrics Complexity [31–34], and Coupling [31, 32, 34, 35] are good options to estimate the quality and performance in oriented-object systems. In the design is not possible to carry out a dynamic evaluation, for instance on the testing, where it is possible because the code is available. The design doesn't have source code, and a static evaluation is made, taking the concept of concerns where the requirements are projected into the architecture of the system, and the approach is developed through UML diagrams, algorithms, and others [22]. The advantage of evaluating the design is cost reduction because it is 12 times lower than in the testing stage [28]. Another advantage is the defects localization and fixed before the code implementation, and finally, test prioritization is also available early, because the background is available to determine which components or their sequences that have a higher probability of failure.

The design architectural evaluation applied in this research provides a predictive model and formal method to assess a quality scheme on oriented object systems. According to the results obtained, it can be estimated that the quality parameter obtained through the evaluation model is based on the complexity and coupling attributes, so the architectural complexity graph (ACG) was established, through which a deterministic analysis is performed. This is the basis to set the quality parameter, and those component sequences with a quality parameter close to +1 have a stronger relationship between complexity and coupling attributes, also indicated at a good-quality level. With the implementation of the Wrapper and MVC design patterns in software architecture to the serious game, according to the scatterplot (Fig. 4.7), 86.6 percent of evaluated sequences close to +1, taking as an indicator the value of 0.5 of the dispensing graph, there are indicating a good quality level based on complexity and coupling quality design. The approach implemented to validate an indirect study of the efficiency of software design architectures using complexity and coupling of software for serious games is logically valid. Although the results are not enough to set quantified parameters for efficiency, it is necessary to estimate another analytical model to quantify the relationship.

4.6 Conclusions

The development of software for serious games in mobile applications implies a learning process like the process of incorporating new learning into memory, as well as retrieving and using it. This requires a software architecture designed to optimize memory and processing resources. The design patterns approach offers quality and performance. In the case of devices with limited storage, for instance, mobile phones, they allow the optimization of resources.

Answered the research questions, the *design patterns* contributed to improving the performance of software architecture on mobile serious games. The software development process to integrate *design patterns* on software architecture in mobile serious games involves an adequate abstraction of requirements employing a holistic vision projected in the design architecture, as well as the selection of quality attributes and design patterns that have the best performance. According to the purpose of the application that was developed at work, the Wrapper and MVC patterns focused on establishing scenarios at runtime. This approach optimizes the use of computing resources such as memory and processing. The process implemented to validate an indirect study of the efficiency of software design architectures through the complexity and coupling of software for serious games is logically valid. Although it is appropriate to determine an analytical model to quantify for efficiency, an additional study is required to complement the quantitative analysis to address the computational cost or to do traditional testing for this quality attribute.

As expressed by the study [23], our software engineering community must contribute studies and techniques to improve the performance of these applications from a formal point of view. The proposal describes a technique to perform and validate the design architecture. The advantage of evaluating the system in early phases like design is *cost reduction to remove defects* and better the software performance. The main limitation of the study is that there are attributes such as efficiency, which must be evaluated in phases after the design stage; however, validating that there is a good architectural design implies a better implementation and consequently a lower number of failures in its operation, which benefits the performance of the system.

The evaluation results set the pattern design implementation on design architecture to serious game software set a high quality related with Complexity and Coupling, allowing us to quantify the advantages of this approach which strengthens that the design techniques implemented are suitable in terms of software performance.

Acknowledgments This work was partially supported by the PNPC-CONACyT 2013–2016 and 2019 21 Master's thesis on Master of Degree Postgraduate of the Autonomous University of Mexico State.

Conflict of Interests We declare that we have no financial or personal conflicts of interest that could inappropriately influence the development of this research.

References

1. Gauthiera, A., Porayska-Pomsta, K., Mayer, S., et al.: Redesigning learning games for different learning contexts: applying a serious game design framework to redesign Stop & Think. Int. J. Child-Comput. Interact. **33**, 100503 (2022)
2. Daylamani-Zad, D., Spyridonis, F., Al-Khafaaji, K.: A framework and serious game for decision making in stressful situations; a fire evacuation scenario. Int. J. Hum.–Comput. Stud. **162**(2022), 102790 (2022)
3. Czauderna, A., Budke, A.: How digital strategy and management games can facilitate the practice of dynamic decision-making. Educ. Sci. **10**(4), 99 (2020)
4. Clark, E.M., Merrill, S.C., Trinity, L., Bucini, G., Cheney, N., Langle-Chimal, O., Shrum, T., Koliba, C., Zia, A., Smith, J.M.: Using experimental gaming simulations to elicit risk mitigation behavioral strategies for agricultural disease management. PLoS One. **15**(3), e0228983 (2020)
5. Mendonca, D., Beroggi, G.E., Van Gent, D., Wallace, W.A.: Designing gaming simulations for the assessment of group decision support systems in emergency response. Saf. Sci. **44**(6), 523–535 (2006)
6. Yamoul, S., Ouchaouka, L., Radid, M., Moussetad, M.: Implementing a serious game as a learner motivation tool, The 4th International Workshop of Innovation Technologies. Procedia Comput. Sci. **210**(2022), 351–357 (2022)
7. Jarnac de Freitas, M., Mira da Silva, M.: Systematic literature review about gamification in MOOCs. Open Learn. J. Open Distance e-Learn., 1–23 (2020)
8. Sailer, M., Homner, L.: The gamification of learning: a meta-analysis. Educ. Psychol. Rev. **32**(1), 77–112 (2020)
9. Connolly, T.M., Boyle, E.A., MacArthur, E., Hainey, T., Boyle, J.M.: A systematic literature review of empirical evidence on computer games and serious games. Comput. Educ. **59**(2), 661–686 (2012)
10. Kara, N.: A systematic review of the use of serious games in science education. Contemp. Educ. Technol. **13**, 2 (2021)
11. Graafland, M., Schraagen, J.M., Schijven, M.P.: Systematic review of serious games for medical education and surgical skills training. Br. J. Surg. **99**(10), 1322–1330 (2012)
12. Bellotti, F., Berta, R., De Gloria, A.: Designing effective serious games: opportunities and challenges for research. Int. J. Emerg. Technol. Learn. (2010)
13. Krath, J., Schürmann, L., von Korflesch, H.F.O.: Revealing the theoretical basis of gamification: a systematic review and analysis of theory in research on gamification, serious games, and game-based learning. Comput. Hum. Behav. **125**(2021), 106963 (2021)
14. Wedyan, F., Abufakher, S.: Impact of design patterns on software quality: a systematic literature review. IET Softw. **14**(1), 1–17 (2021)
15. Fletcher, J., Cleland-Huang, J.: Soft goal traceability patterns. In: 17th International Symposium on Software Reliability Engineering (ISSRE'06), pp. 363–374. IEEE, Raleigh, NC (2006)
16. Gamma, E., Henry, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley, New York, NY (1994)
17. Dávila-Nicanor, L., Orozco Aguirre, H.R., Quintana López, M., Banda Madrid, A.: Enhancement to test case prioritization through object-oriented software architectural design. In: 10th International Conference on Software Process Improvement (CIMPS) 2021, pp. 131–138 (2021)
18. Galvis Panqueva, A.H.: Ingeniería de Software Educativo. Ediciones Uniandes - Universidad de los Andes, Santafé de Bogotá, Colombia (1992)
19. Bezanilla, M.J., Arranz, S., Rayon, A., Rubio, I., Menchaca, I., Guenaga, M., Aguilar, E.: Propuesta de evaluación de competencias genéricas mediante un juego serio. New Approach. Educ. Res., 44–54 (2014)

20. Gu, S.M., Wu, Y., Wu, W.Z., Li, T.J.: Knowledge approximations in multi-scale ordered information systems. In: Rough Sets and Knowledge Technology, Shanghai, China, 2014, pp. 525–534 (2014)
21. IEEE Std 1471-2000: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Computer Society (2000)
22. Oliver, V., Ingo, A., Arif, C.K.: Timo Software Architecture. Springer Nature, New York, NY (2011)
23. Mizutani, W.K., Daros, V.K., Kon, F.: Software architecture for digital game mechanics. Entertain. Comput. **38**, 100421 (2021)
24. Dzaferagic, M., Kaminski, N., Macaluso, I., Marchetti, N.: Relation between functional complexity, scalability, and energy efficiency in WSNs. In: 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 2017, pp. 675–680 (2017)
25. Bansiya, J., Davis, C.G.: A hierarchical model for object-oriented design quality assessment. IEEE Trans. Softw. Eng. **28**(1), 4–17 (2001)
26. Jayalath, T., Thelijjagoda, S.: A modified cognitive complexity metric to improve the readability of object-oriented software. In: International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, 2020, pp. 37–44 (2020)
27. Arvanitou, E.M., Ampatzoglou, A., Chatzigeorgiou, A., Galster, M., Avgeriou, P.: A mapping study on design-time quality attributes and metrics. J. Syst. Softw. **127**, 52–77 (2017)
28. Park, R.E., Wolfhart, B.: Goal-Driven Software Measurement. A Guidebook. Software Engineering Institute, Carnegie Mellon University, Pittsburgh (1996)
29. Madi, A., Zein, O.K., Kadry, S.: On the improvement of cyclomatic complexity metric. Int. J. Softw. Eng. Appl. **7**(2) (2013)
30. Lewis, A.D., Groth, K.M.: Metrics for evaluating the performance of complex engineering system health monitoring models. Reliab. Eng. Syst. Saf. **223**, 108473 (2022)
31. Singh Rathore, S., Gupta, A.: Investigating object-oriented design metrics to predict fault-proneness of software modules. In: 2012 CSI Sixth International Conference on Software Engineering (CONSEG), Indore, India (2012)
32. Iyapparaja, M., Sureshkumar, D.: Coupling and cohesion metrics in java for adaptive reusability risk reduction. In: IET Chennai 3rd International on Sustainable Energy and Intelligent Systems (SEISCON 2012), Tiruchengode, India (2012)
33. Yadav, V., Singh, R.: The prediction design quality of object-oriented software using UML diagrams. In: 3rd International Advance Computing Conference (IACC), pp. 1462–1467, Ghaziabad, India (2013)
34. Halim, A.: Predict fault-prone classes using the complexity of UML class diagram. In: International Conference on Computer, Control, Informatics and Its Applications (IC3INA), Jakarta, Indonesia (2013)
35. Kumar, G.P., Joshi, G.: QMOOD metric sets to assess the quality of the java program. In: International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India (2014)
36. Google Developers 2018. https://sites.google.com/a/strategiesinsoftware.com/site/commentary/googlecodeproanalytix (2022). Accessed Aug 2022

Chapter 5
ENTRUST: Co-design and Validation of a Serious Game for Assessing Clinical Decision-Making and Readiness for Entrustment

Edward F. Melcer, Cara A. Liebert, Samuel Shields, Oleksandra G. Keehl, Jason Tsai, Fatyma Camacho, Hyrum Eddington, Amber Trickey, Melissa Lee, Sylvia Bereknyei Merrell, James R. Korndorffer Jr., and Dana T. Lin

Abstract Graduate medical education is moving toward a competency-based paradigm, predicated upon multiple real-time assessments to verify clinical and technical proficiency (i.e., readiness for entrustment of residents). This requires not only assessment of technical skills and medical knowledge but also critical clinical decision-making skills in preoperative, intraoperative, and postoperative settings. However, most medical education programs have adopted reductionist approaches, reducing assessment of readiness for entrustment to only assessing technical skill performance. As such, there is a growing need for tools that can provide more comprehensive and objective evaluations of the proficiency of residents to perform medical procedures. This chapter presents *ENTRUST*, our serious game-based online platform to assess trainees' decision-making competence across various Entrustable Professional Activity (EPA) domains. Specifically, we discuss (1) the design of *ENTRUST*; (2) insights identified and lessons learned throughout the development process that can aid collaboration between serious game developers and subject matter experts; and (3) results from a pilot study of *ENTRUST*—demonstrating the tool's capability to discriminate between levels of surgical expertise and providing initial validity evidence for its use as an objective assessment for clinical decision-making.

E. F. Melcer (✉) · S. Shields · O. G. Keehl · J. Tsai · F. Camacho
University of California, Santa Cruz, Santa Clara, CA, USA
e-mail: eddie.melcer@ucsc.edu; samshiel@ucsc.edu; okeehl@ucsc.edu; ctsai32@ucsc.edu; fcamach1@ucsc.edu

C. A. Liebert · H. Eddington · A. Trickey · M. Lee · S. B. Merrell · J. R. Korndorffer Jr. · D. T. Lin
Stanford University School of Medicine, Stanford, CA, USA
e-mail: carap@stanford.edu; hyrumedd@stanford.edu; atrickey@stanford.edu; melchlee@stanford.edu; sylviab@stanford.edu; korndorffer@stanford.edu; danalin@stanford.edu

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_5

Keywords Game-based assessment · Games for Health · Serious game design · Co-design · Clinical decision-making · Entrustable Professional Activities

5.1 Introduction

In recent years, medical education has moved toward a competency-based paradigm predicated upon multiple, real-time assessments to verify proficiency [43]. Within this new paradigm, Entrustable Professional Activities (EPAs) –or units of professional practice that constitute what clinicians do as daily work– were created to bridge the gap between competency frameworks and clinical practice [44]. EPAs are effective tasks or responsibilities to be entrusted to a trainee once they have attained competence at a specific level and embody a more global integration of the Accreditation Council for Graduate Medical Education (ACGME) core competencies [43]. Notably, there has been a widespread initiative to adopt and incorporate EPAs in graduate medical training as a means of transitioning toward a more competency-based educational paradigm. In 2018, the American Board of Surgery (ABS) initiated a nationwide pilot tasking 28 general surgery programs to explore the use and implementation of 5 core general surgery EPAs, with the intention of formalizing EPAs as a requirement for all general surgery training programs by 2023 [31].

The determination of readiness for entrustment is typically predicated upon direct observation and assessment of behaviors by faculty in the clinical setting [11]. While frequent, real-time microassessments are ideal in assessment of EPAs and readiness for entrustment, this approach places a sizeable and continuous burden on faculty to regularly complete evaluations for the many individual interactions they have with multiple trainees who are to be graded across a variety of clinical skills and EPAs. In addition, there is variability in the types and severity of patient cases encountered in the real-world clinical setting, making it difficult to reliably evaluate trainees' ability to manage rare diseases or complications [45]. Conversely, virtual patient simulations enable trainees to demonstrate their clinical and surgical decision-making in an objective, reproducible, and measurable way while decompressing the assessment burden off faculty raters [4]. In addition, standardized scenarios may be deployed to minimize implicit bias and subjectivity, reduce test anxiety, and test infrequently encountered, yet critical, clinical conditions [27, 49].

Given these challenges, many pilot institutions have operationalized EPAs by adopting reductionistic approaches and focusing on assessment of operative performance only, as readily available tools exist to measure this construct, e.g., [6, 15, 18, 32, 37, 38, 47]. One mobile operative microassessment application, SIMPL (System for Improving and Measuring Procedural Learning) [6, 17, 18], has been widely utilized by surgical training programs to rate trainee's technical skills. While it possesses robust validity evidence for evaluating operative autonomy [6, 15, 18], it does not assess clinical decision-making. However, based on the EPA definitions and essential functions articulated by the ABS, clinical decision-making competence in the preoperative, intraoperative, and postoperative setting constitutes critical components of entrustment. As a result, readiness for entrustment should include assessment of both operative autonomy and clinical decision-making. Therefore, there is a great need for evidence-based EPA-aligned tools that specifically address clinical decision-making, as a complement to existing technical skills evaluations.

To address this need for an objective, efficient, and scalable means to assess clinical and surgical decision-making, we developed *ENTRUST*—a virtual patient authoring and serious game-based assessment platform to deploy rigorous, case-based patient simulations for evaluation of EPAs. In this chapter, we present (1) the design of *ENTRUST*; (2) insights identified and lessons learned throughout the development process that can aid collaboration between serious game developers and subject matter experts; and (3) results from a pilot study of *ENTRUST*—demonstrating its capability to discriminate between levels of surgical expertise and providing initial validity evidence for its use as an objective assessment for clinical decision-making.

5.2 Background

5.2.1 Entrustable Professional Activities

In 2018, the ABS commenced a multi-institutional pilot to implement five general surgery EPAs, each with defined levels of entrustment from Level 0 to Level 4, in surgical residency [1, 7]. These initial five ABS EPAs include (1) evaluation and management of a patient with inguinal hernia, (2) evaluation and management of a patient with right lower quadrant pain, (3) evaluation and management of a patient with gallbladder disease, (4) evaluation and management of a patient with blunt/penetrating trauma, and (5) providing general surgical consultation to other healthcare providers [7]. Additionally, the ABS has given individual residency programs the ability to determine how EPAs are piloted and assessed at their institution. While tools exist for the intraoperative assessment of technical skills and operative autonomy [6, 18, 32, 37, 38, 47], they do not directly not assess clinical decision-making across the preoperative, intraoperative, and postoperative settings. The assessment of technical skills is necessary, but is not sufficient, to determine entrustment [45]. Therefore, there is a notable gap in the literature and need for efficient, objective, evidence-based, EPA-aligned tools that assess clinical decision-making across the entire course of surgical care, as a fitting complement to existing technical skill and intraoperative evaluations.

5.2.2 Game-Based Assessment in the Health Domain

Educational assessment has evolved over the past decade from traditional pen-and-paper-based tests to the use of technology such as games to assess various competencies in the form of game-based assessment [48]. Notably, due to the technological enhancement of what can be measured, game-based assessment provides promising possibilities for more valid and reliable measurement of students' skills, knowledge, and attributes compared to the traditional methods of assessment such as paper-and-pencil tests or performance-based assessments [13]. Within the health domain, game-based assessment has been utilized in a variety of contexts including assessment of patient health [46], assessment of motor skills and ability to perform first aid [9], neuropsychological assessment [16], and assessment of health-related knowledge/learning [33], to name a few. However, in the context of clinical reasoning and decision-making, the predominant focus of serious games has been on training and learning, e.g., [10, 20, 21, 23, 24, 26, 29, 30]. This surprising lack of game-based assessment for clinical reasoning and decision-making highlights the notable gap in the literature and further emphasizes the need for evidence-based EPA-aligned game-based assessment tools that specifically address clinical decision-making—such as *ENTRUST*. Furthermore, such game-based assessment tools offer a number of potential benefits over traditional forms of assessment if employed correctly including reduced test anxiety [27] and more authentic contexts for assessing competency, which is crucial for acquiring more accurate assessments of skill [40].

5.3 Design of *ENTRUST*

ENTRUST is a serious game-based online virtual patient simulation platform to both train and assess medical trainees' decision-making competence within EPAs. It is therefore targeted at training and assessing the competency of the next generation of clinicians at the medical student and resident levels.

5.3.1 Co-design Process

We utilized a co-design approach for the design and development of *ENTRUST*—which is a widely used approach within the health field [41]. Co-design stems from participatory design, where the people destined to use the system play a critical role in designing it [39]. However, in a co-design process, stakeholders are treated as equal collaborators or can even take the lead in the design process rather than have limited roles [42]. In this way, co-design involves a shift in the locus of responsibility and control so that "clients" or users of services become active partners in designing, shaping, and resourcing services, rather than being passive recipients of pre-determined services [8]. For *ENTRUST*, we worked directly with medical education experts and continue to do so as co-designers (i.e., full partners in the entire design process [41]) on the project. We found this approach to be critical for the successful design and development of *ENTRUST* as the subject matter of clinical decision-making and entrustment is too complex for a serious game development team to successfully design, develop, and maintain on their own. As such, our research team utilizes the following co-design and agile development process (with a number of the steps drawn from [5]):

1. Contextual inquiry in the form of informational interviews and weekly artifact review meetings with medical education experts to identify latent needs, challenges experienced, and desired future state/artifact creation.
2. Generation of design and rapid prototyping to address identified needs and challenges. This is done through the development of new *ENTRUST* artifacts (e.g., creating an authoring platform to complement the game-based assessment tool) or incorporation of desired features into existing artifacts (e.g., adding a new vital sign algorithm to the simulation mode and authoring platform). Repeat steps 1 and 2 weekly.
3. Sharing ideas and receiving feedback through periodic presentations of design and development work on *ENTRUST* to larger subsections of the medical education community.
4. Conducting studies and data analysis to empirically validate *ENTRUST* designs.
5. Interpreting results for requirements translation, i.e., identifying action items, feasible priorities, and feeding back into steps 1 and 2.

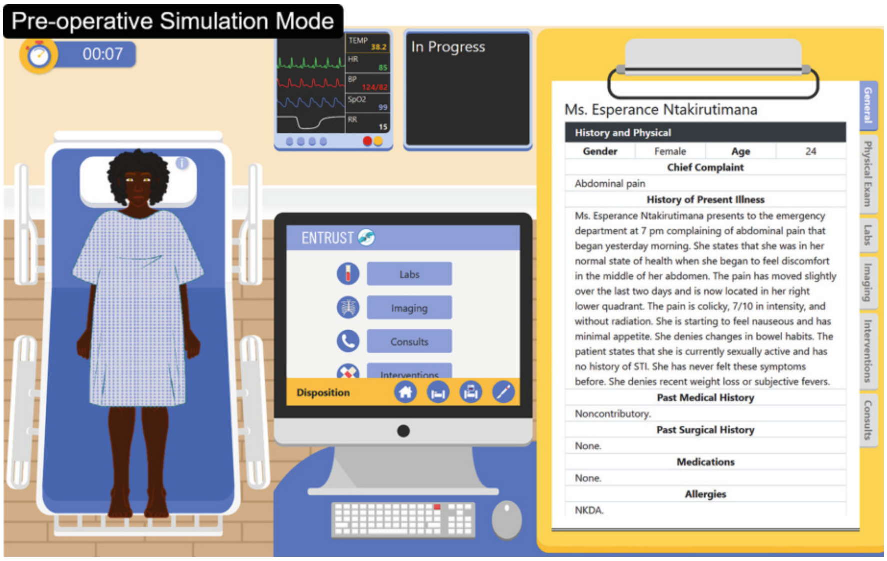
This co-design process has resulted in the current iteration of *ENTRUST* as described below.

5.3.2 Assessment Platform

The current *ENTRUST* platform includes two primary phases: simulation mode and question mode.

5.3.2.1 Simulation Mode

In simulation mode (see Fig. 5.1), the examinee engages with patient case scenarios starting from the preoperative setting. This setting can be in either the emergency department or the outpatient clinic, where the examinee initiates a physical examination and full workup of the patient. During workup, the examinee can order diagnostic tests, administer fluids and medications, perform bedside procedures, and request consultation. All actions –both player evoked (such as conducting a physical exam) and game evoked (such as changing vital signs due to deteriorating



Ms. Esperance Ntakirutimana

History and Physical			
Gender	Female	**Age**	24
Chief Complaint			
Abdominal pain			
History of Present Illness			
Ms. Esperance Ntakirutimana presents to the emergency department at 7 pm complaining of abdominal pain that began yesterday morning. She states that she was in her normal state of health when she began to feel discomfort in the middle of her abdomen. The pain has moved slightly over the last two days and is now located in her right lower quadrant. The pain is colicky, 7/10 in intensity, and without radiation. She is starting to feel nauseous and has minimal appetite. She denies changes in bowel habits. The patient states that she is currently sexually active and has no history of STI. She has never felt these symptoms before. She denies recent weight loss or subjective fevers.			
Past Medical History			
Noncontributory.			
Past Surgical History			
None.			
Medications			
None.			
Allergies			
NKDA.			

Fig. 5.1 The simulation mode within *ENTRUST*. Enables examinees to engage with patient case scenarios starting at the preoperative setting, including physical examination and full patient workup

patient condition)– are recorded and scored on the back-end database according to an expert-consensus-derived scoring algorithm (see Sect. 5.3.4). Points are earned for ordering relevant labs and key interventions; conversely, points are lost for performing inappropriate, unnecessary, or harmful actions.

Notably, the *ENTRUST* interface in this mode consists of six key features that enable examinee input for assessment and provide feedback from the simulation:

1. **Timer** (Fig. 5.1 Top Left)—the timer displays the amount of time the examinee has been active in the preoperative setting. During play, 1 second of game time displayed on the timer equates to 1 minute of time taken in a real-world scenario.
2. **Patient/physical exam** (Fig. 5.1 Middle Left)—the virtual patient enables examinees to conduct a physical examination and see results in the medical chart. As examinees move their mouse over the virtual patient, various icons and images will appear to indicate that a physical examination can be conducted on that part of the body with a mouse click. Patient facial expressions also change depending on their health status throughout the course of the preoperative setting.
3. **Notifications** (Fig. 5.1 Bottom Left)—notifications appear in the bottom-left corner of the screen after each physical examination to report the results. This is done to remove the need to go to the right side of the screen to view a physical exam result in the medical chart before returning to continue examining the patient on the left side of the screen, i.e., to reduce extrinsic cognitive load [22, 34].

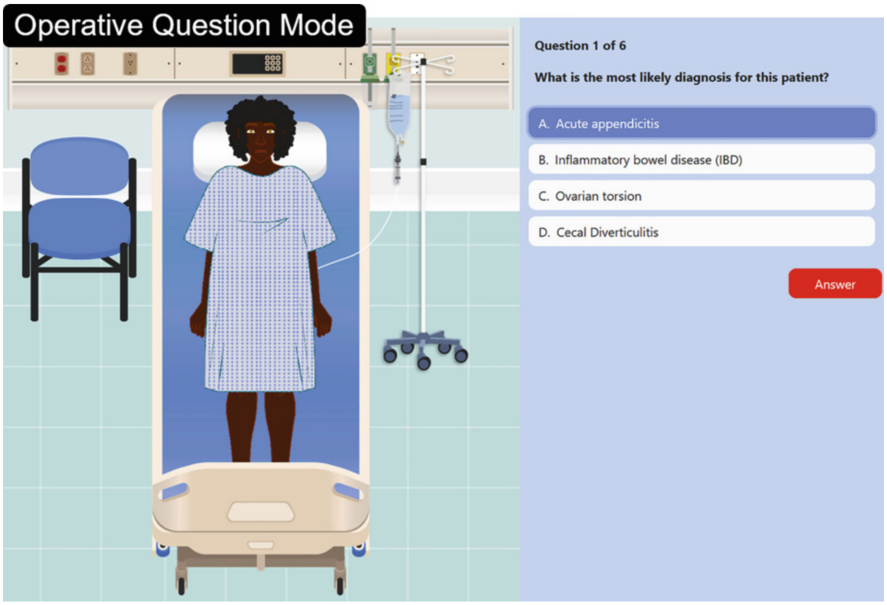
4. **Vital monitor and order progress monitor** (Fig. 5.1 Top Middle)—the vital monitor shows the virtual patient's vital signs throughout the preoperative simulation. Vitals are updated in real time (relative to game time) and can deteriorate due to lack of or improper treatment as well as improve due to performing appropriate bedside procedures or administering appropriate fluids or medications. An audible alarm (similar sounding to real-world vital machine alarms) can also be heard when patient vitals reach a dangerous level. The order progress monitor shows the time remaining for any diagnostic test, administration of fluids and medications, bedside procedures, or consultations ordered. The exact amount of seconds remaining is shown in the progress bar and mirrors typical real-world times taken for each order at a rate of one game second to one real-world minute.
5. **Order console** (Fig. 5.1 Bottom Middle)—the order console enables the examinee to order diagnostic tests, administer fluids/medications, perform bedside procedures, and request consultation. It also allows the examinee to make decisions about disposition, e.g., whether the patient should go home, to the operating room (OR), or to the intensive care unit (ICU) or proceed with nonoperative management. Selecting a disposition or causing the patient to go into cardiac arrest will proceed to the question mode of *ENTRUST*.
6. **Medical chart** (Fig. 5.1 Right)—the medical chart maintains and displays all relevant information regarding the virtual patient. This includes their medical history and initially reported health complaint as well as the results from all physical exams and orders placed. Examinees can click the tabs on the right side of the chart to toggle between this information. Whenever there is a change to the medical chart, such as when a physical exam or order is completed, the corresponding tab displays a red dot to indicate new information is available.

5.3.2.2 Question Mode

ENTRUST switches to question mode (Fig. 5.2) when the examinee opts to proceed to the operating room. In question mode, the examinee is tested on intraoperative and postoperative knowledge, decision-making, and management of complications via a series of single-best answer multiple-choice questions. Points are awarded for answering correctly and deducted for answering incorrectly.

5.3.3 Authoring Platform

ENTRUST also features an online authoring portal that is designed to be accessible for clinicians and content experts to create and deploy new case scenarios without requiring programming experience or directly modifying the game (Fig. 5.3). This portal provides user-friendly, easy creation, and customization options for a variety



Question 1 of 6

What is the most likely diagnosis for this patient?

A. Acute appendicitis
B. Inflammatory bowel disease (IBD)
C. Ovarian torsion
D. Cecal Diverticulitis

Fig. 5.2 The question mode within *ENTRUST*. Examinees are tested on intraoperative and postoperative knowledge, decision-making, and management of complications via a series of single-best answer multiple-choice questions

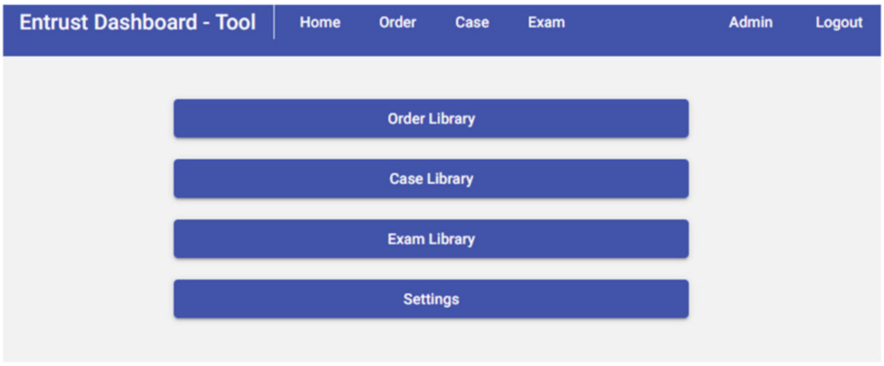


Fig. 5.3 The *ENTRUST* authoring platform. Enables clinicians and content experts to easily create and deploy new case scenarios without requiring programming experience or direct modification of the game

of aspects needed for assessment of clinical decision-making skills. Specifically, the portal provides (1) an order library for creation and management of orders that can be used in case scenarios; (2) a case library that allows for creation and management of all aspects related to a case scenario for assessment; and (3) an exam library that enables the sequencing of case scenarios to create a wide spectrum of exams. These numerous customization options allow for virtually unlimited cases and to be crafted, providing control of aspects ranging from varying patient age, appearance, and apparel via a novel patient character generating tool to specialized labs and orders on the displayed intervention menu.

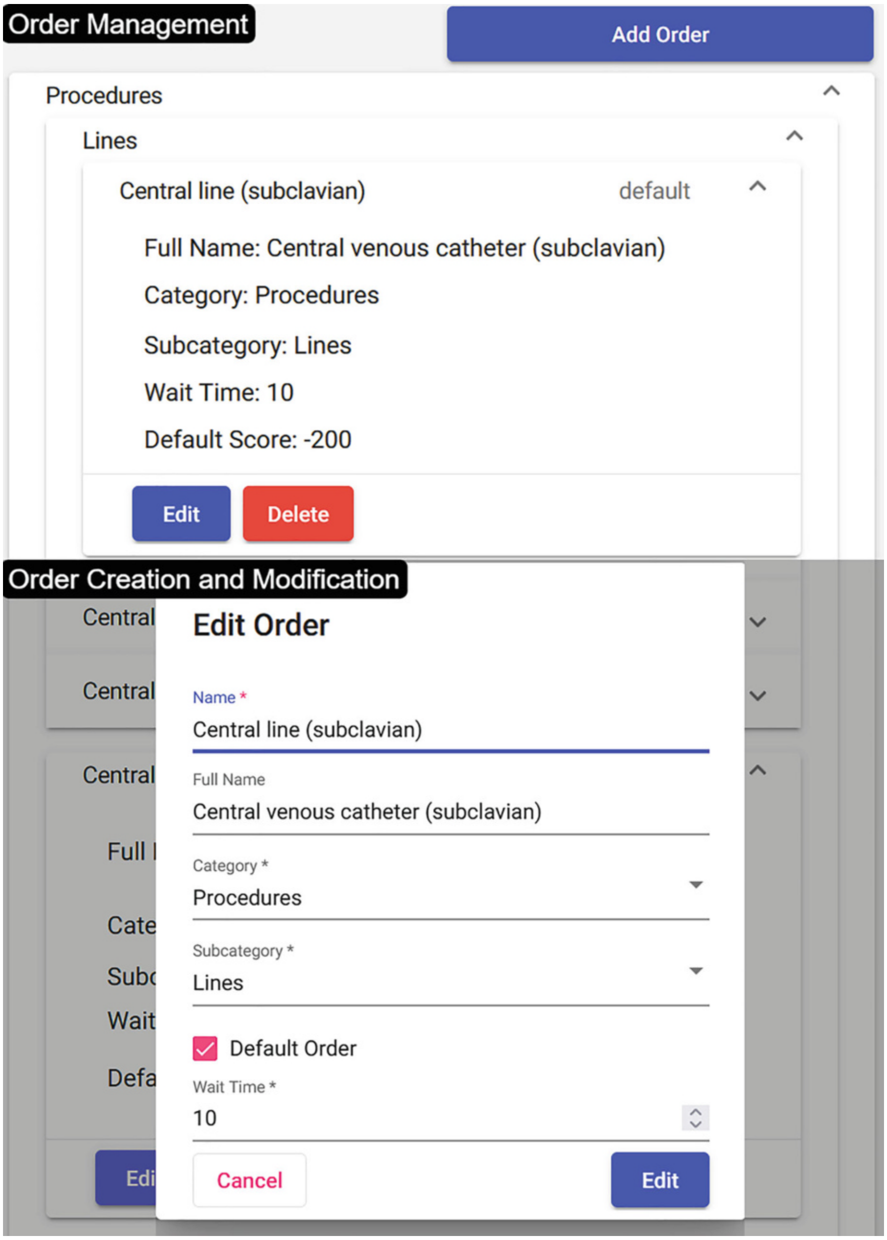
5.3.3.1 Order Library

The order library enables authors to create, manage, and modify a database of orders for use in any case scenario (see Fig. 5.4). The order library is designed to be modular and reusable, enabling authors to specify all default information necessary for a particular order to work within any case while leaving scenario specific details (such as scoring or abnormal results) to be specified in a case-by-case basis within the case library. Specifically, the order library enables authors to easily specify:

- **Order Name**
- **Order Category** (Procedure, Lab, Imaging, Medication, Transfusion, Consult)
- **Order Subcategory**, which is dependent upon what order category was selected
- **Default Order**, i.e., whether it should be included by default when creating any new case scenario in the case library)
- **Wait Time** in seconds for the order to complete during simulation
- **Default Score** when the order is made during simulation
- **Default Result** when the order is made during simulation—there are also additional options to specify if the result should randomly fall within a number range or use a default image if applicable or if there should be multiple default results provided simultaneously
- **Unit** of the default result if applicable

5.3.3.2 Case Library

The case library enables authors to create, manage, and modify a database of case scenarios for use in any examination (see Fig. 5.5). The case library is designed to enable authors to specify all core aspects of a case scenario, including designating effects of interventions on vital signs and determining the appropriateness of actions by rewarding and penalizing examinees on a tiered scoring system. Clinical vignettes and multiple-choice questions can be entered and edited with ease and flexibility as well. Additionally, media files such as photographs and radiology images can be uploaded to be interpreted by the examinee. The specific configuration options the case library provides are:



Full Name: Central venous catheter (subclavian)

Category: Procedures

Subcategory: Lines

Wait Time: 10

Default Score: -200

Name *: Central line (subclavian)

Full Name: Central venous catheter (subclavian)

Category *: Procedures

Subcategory *: Lines

✓ Default Order

Wait Time *: 10

Fig. 5.4 The *ENTRUST* authoring platform order library tool. Enables authors (e.g., clinicians and content experts) to easily create and manage modular orders that can be used in any case

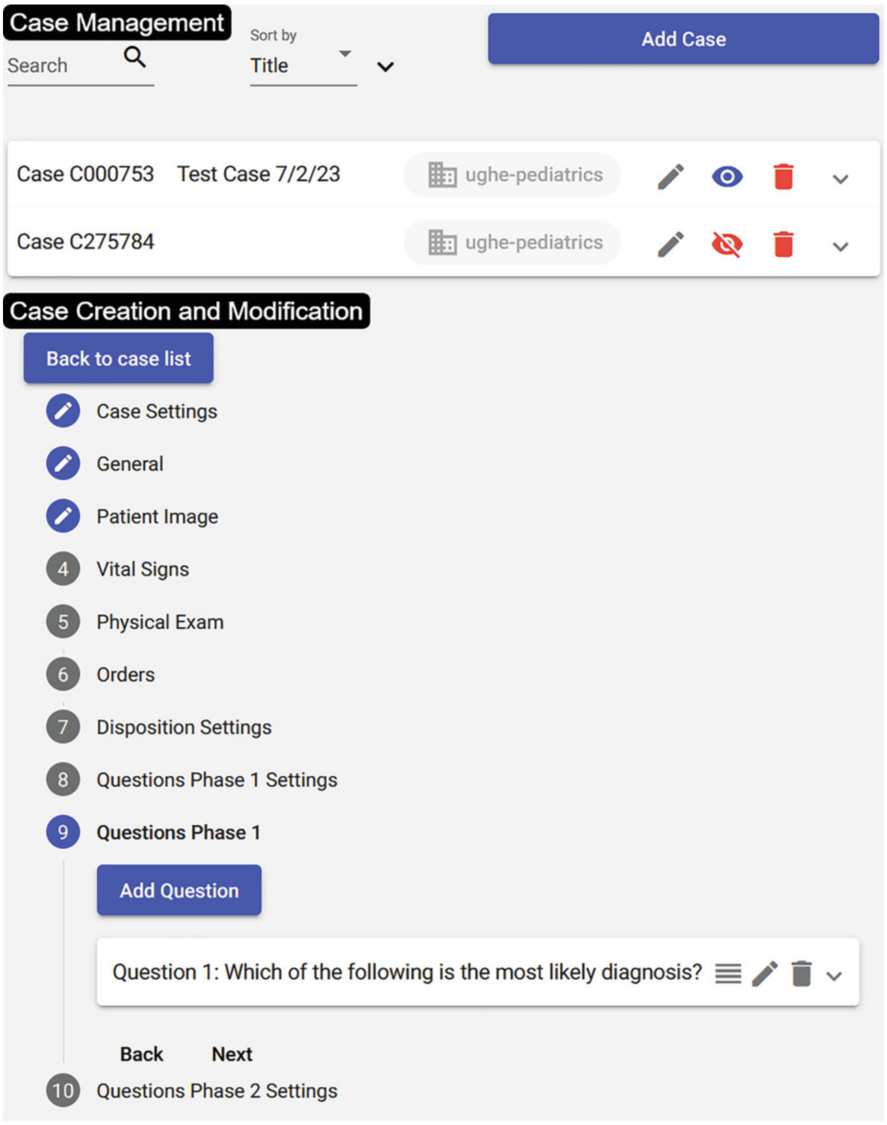
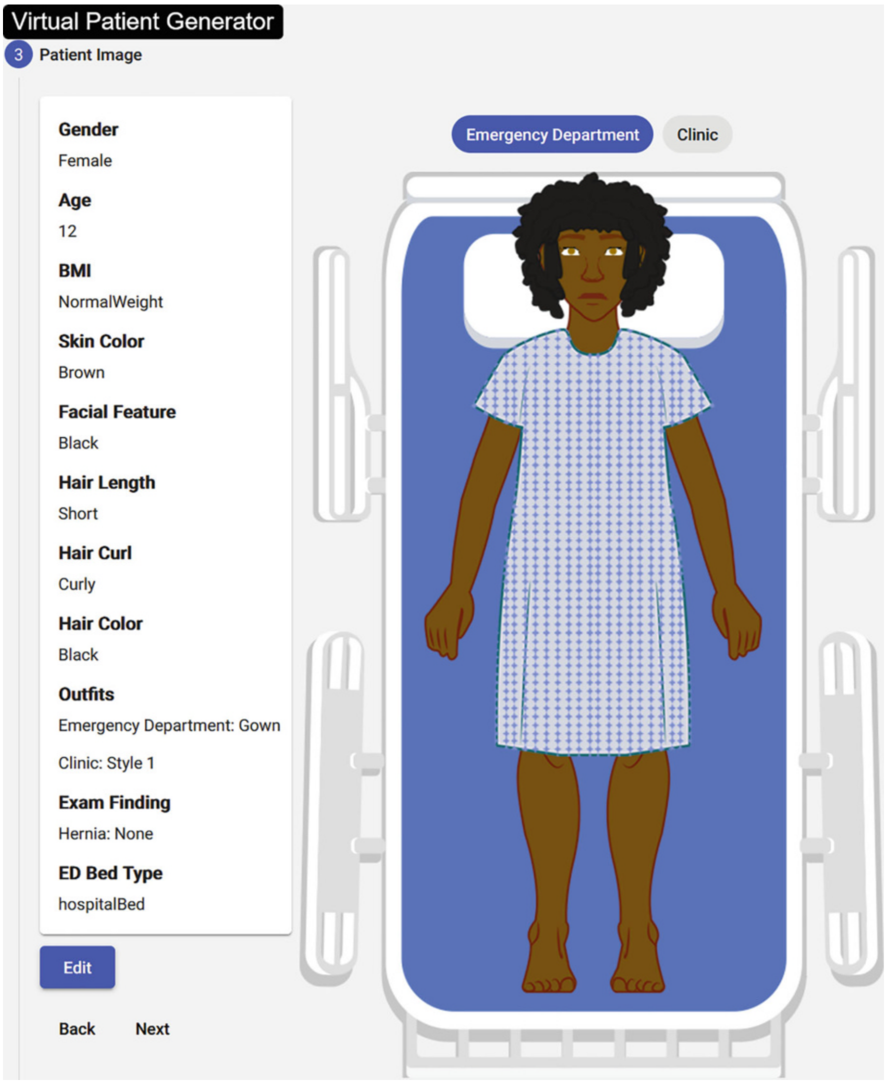


Fig. 5.5 The *ENTRUST* authoring platform case library tool. Enables authors (e.g., clinicians and content experts) to easily create and manage case scenarios for examinations

1. **General information**—this section enables authors to specify basic information about the case scenario (such as title, summary, whether it occurs in the emergency room or clinic, and so forth) as well as general information about the virtual patient (such as patient name, their reported ailment, present illness, past medical and surgical history, medications, allergies, and so forth).
2. **Patient image**—this section provides a *virtual patient generation tool* (see Fig. 5.6) that enables authors to customize a wide range of details about the virtual patient such as their sex, age, BMI, skin color, facial features, hair, what they wear when on-screen, visible physical abnormalities during a physical exam (such as a hernia), where the incision site will be displayed during the question mode, and if they will have a C-collar or backboard for certain kinds of injuries. Notably, the broad range of customization options allows for representation of a diverse range of patients from infant to elderly, underweight to morbidly obese, and so forth (see Fig. 5.6 Right for some examples).
3. **Vital sign settings**—this section enables authors to specify the starting vitals for the virtual patient in the simulation mode, as well as specify a *vital sign update algorithm* that specifies how the patient's vitals will change throughout the simulation mode. Vital sign algorithms realistically replicate how certain vitals would change over time in the real world for certain conditions. Current options include clinic patient, stable ED patient, isolated tachycardia, hemorrhagic shock, sepsis, and septic shock.
4. **Physical exam**—this section enables authors to specify the results and score for performing various physical examinations on the virtual patient. Current physical examinations available to the examinee include general, HEENT (head, ears, eyes, nose, and throat), breast, cardiovascular, pulmonary, abdomen, left/right genitourinary, and extremities.
5. **Orders**—this section enables authors to specify what orders (i.e., procedures, labs, imaging, medications, transfusions, consults, or fluids) are available to the examinee in a specific case scenario, as well as the results and positive/negative score effect placing that order will have. By default, when a new order is added to a case scenario, it uses the default details, result(s), and score change specified in the order library. However, authors are also able to modify an order, for that specific case scenario only, to specify sophisticated result and scoring logic (see Fig. 5.7). Specifically, authors can (1) *customize results*, such as change findings or add a different image if applicable to show patient abnormalities for a case scenario; (2) *set new scoring logic* for use of an order, including setting additional penalties for extraneous, repeated use of an order when not appropriate; (3) *set pretest effects* if applicable; and (4) *set vital sign changes* that will occur upon making an order if applicable, e.g., by ordering fluids.



Gender
Female
Age
12
BMI
NormalWeight
Skin Color
Brown
Facial Feature
Black
Hair Length
Short
Hair Curl
Curly
Hair Color
Black
Outfits
Emergency Department: Gown
Clinic: Style 1
Exam Finding
Hernia: None
ED Bed Type
hospitalBed

Fig. 5.6 The *ENTRUST* authoring platform virtual patient generation tool. Enables authors (e.g., clinicians and content experts) to easily define key patient details and visualizes how these will look in real time. Notably, the broad range of options allows for representation of a diverse range of patients from infant to elderly, underweight to morbidly obese, and so forth

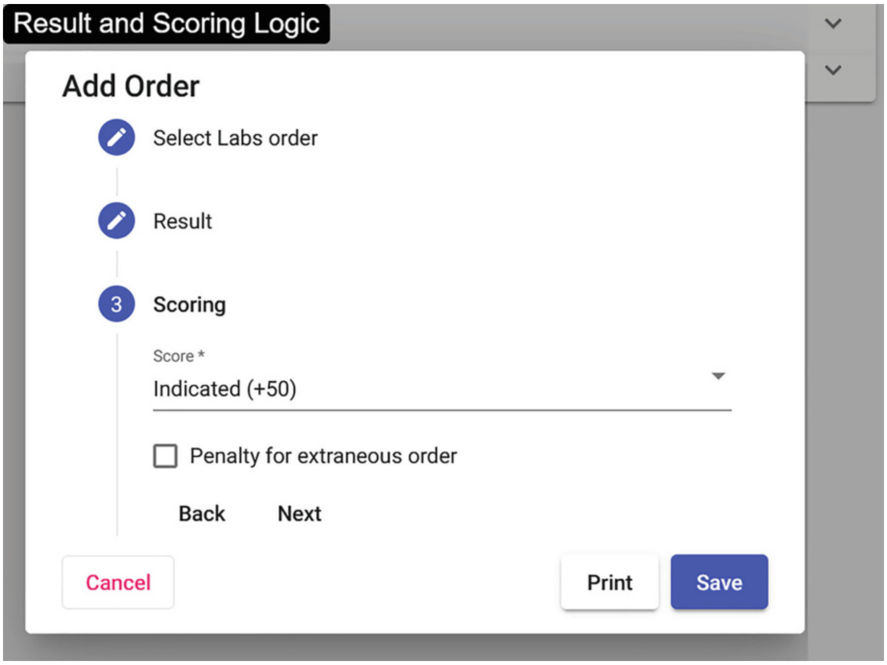


Fig. 5.7 The customization of results and scoring logic within the *ENTRUST* authoring platform. Enables authors (e.g., clinicians and content experts) to easily define how a specific order will impact results, score, vital sign changes, and so forth for a particular case scenario

6. **Disposition settings**—this section enables authors to specify scoring for each potential disposition choice made by the examinee. Current disposition options include sending the patient home, to a ward, to the ICU, or to the OR or to proceed with nonoperative management.
7. **Intraoperative and postoperative questions**—these sections enable authors to specify single-best answer multiple-choice questions and related settings for questions that will appear in the question mode.

5.3.3.3 Exam Library

The exam library enables authors to create, manage, and modify a database of exams for use in assessment (see Fig. 5.8). Authors are able to create a new exam, select any case scenario from the case library to include in the exam, and modify the order of case scenario appearance. During play, examinees are given a prompt at completion of a case to start the next case (if applicable) upon clicking the “Next” button.

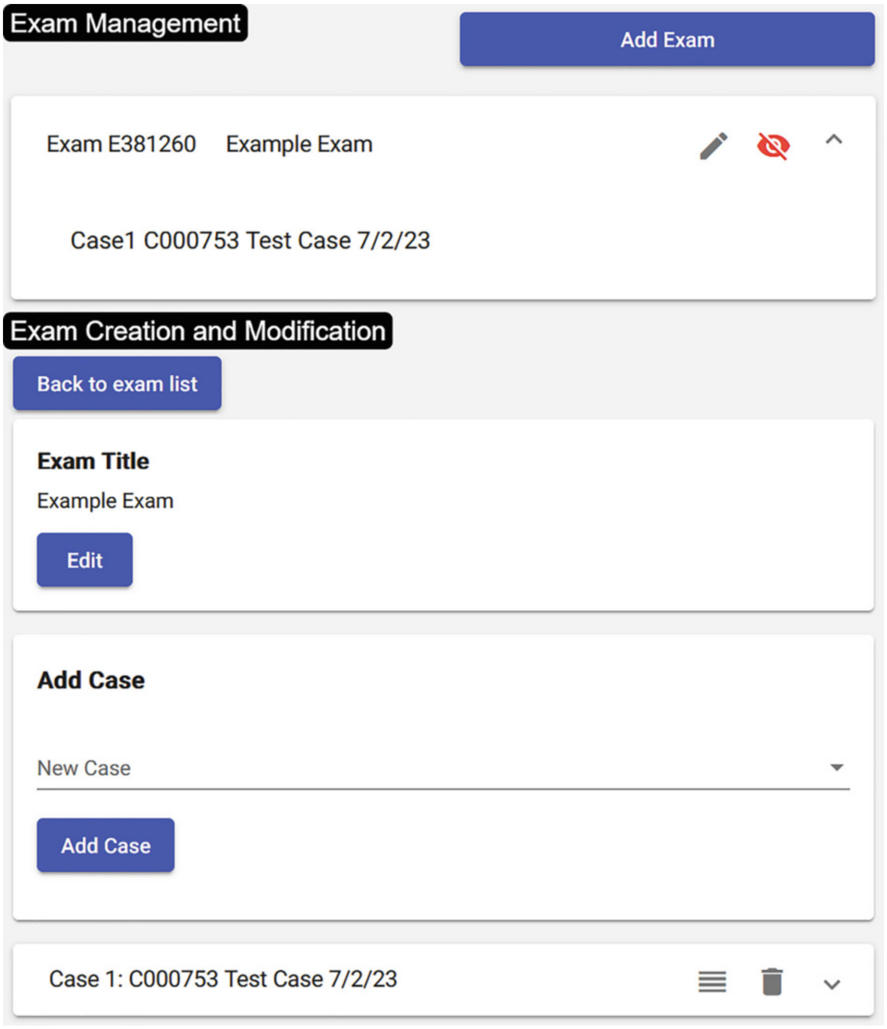

Exam Management
Add Exam
Exam E381260 Example Exam
Case1 C000753 Test Case 7/2/23
Exam Creation and Modification
Back to exam list
Exam Title
Example Exam
Edit
Add Case
New Case
Add Case
Case 1: C000753 Test Case 7/2/23

Fig. 5.8 The *ENTRUST* authoring platform exam library tool. Enables authors (e.g., clinicians and content experts) to easily create and manage a specified series of case scenarios in the form of an exam for use in assessment

5.3.4 Case Creation and Scoring Algorithm

Over a dozen cases have already been authored and iteratively refined to align with EPA standards for inguinal hernia, thyroid disease, and breast disease as articulated by the American Board of Surgery [7]. Based on feedback from an expert panel, the cases were iteratively revised with the final case scenarios reviewed and approved by the case authors.

A scoring algorithm for *ENTRUST* was also designed to reflect appropriateness of actions, patient clinical status, and accuracy of multiple-choice question responses. This scoring algorithm was vetted by two board-certified surgeons with formal training in surgical education to reflect appropriateness of clinical interventions and multiple-choice question responses. The case scenario and scoring algorithm have also been beta-tested internally by the research team prior to studies and data collection to ensure proper functionality of each case. Specifically, for diagnostic studies and interventions employed during the simulation mode, scoring was categorized using the following framework:

- **Critical** [+200]
- **Indicated** [+100]
- **Optional** [0]
- **Not Indicated but Not Harmful** [−50]
- **Mild to Moderate Harm** [−100]
- **Severe Harm** [−200]
- **Death/Cardiac Arrest** [−500]

Additionally, during simulation mode, points are deducted for each instance of failure to address and correct vital sign abnormalities [−200]. During question mode, multiple-choice questions were awarded +200 points for correct responses and −200 for incorrect responses.

5.3.5 Technical Specifications and Data Collection

ENTRUST utilizes a JavaScript and P5.js front end to provide an interactive simulation interface, as well as a Google Cloud Platform backend for secure data logging and analysis of demographic data, gameplay actions, and scores during gameplay. The platform works on most modern browsers (Chrome, Firefox, and Edge) and is easily distributable to a wide range of participants through a simple Web link. *ENTRUST* requires minimal computational resources to deploy the simulations and can therefore be run on almost any modern computer. The ease of distribution through Web browsers coupled with low computational needs makes *ENTRUST* ideal for deployment in most countries around the world.

ENTRUST's secure backend database logs detailed player performance data including a time stamp of all examinee actions, changes in patient vital signs, points awarded or deducted for an action or intervention, and responses to all multiple-choice questions. The database may be queried to extract data in aggregate format for program-specific or research purposes.

5.4 Study: *ENTRUST* Inguinal Hernia EPA Assessment Pilot

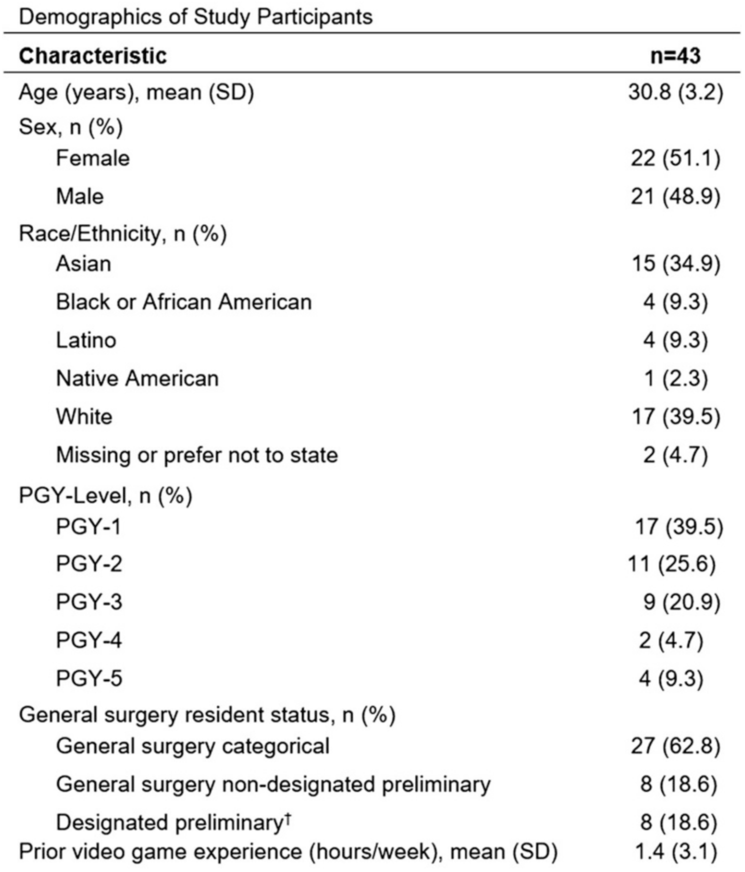
In order to provide initial validity evidence for *ENTRUST*'s capabilities as a tool for assessment of clinical decision-making skills and entrustability, we conducted an initial pilot study of an Inguinal Hernia EPA Assessment developed on *ENTRUST*—this study and results were initially reported in [25]. We hypothesized that *ENTRUST* possesses validity evidence for use in the assessment of clinical decision-making for general surgery residents. As a result, we posited the following research questions:

1. Do users of a game-based assessment tool such as *ENTRUST* need to have prior video game experience to successfully engage with the tool?
2. Does score-based performance on *ENTRUST* discriminate between levels of surgical expertise, e.g., prior operative experience or post-graduate year of training?
3. Is *ENTRUST* able to assess critical surgical decision-making performance?

5.4.1 Methodology

5.4.1.1 Participants

A total of 43 surgical residents at a US-based academic institution participated in the study. Participants included general surgery categorical residents, general surgery preliminary residents, and designated surgical subspecialty residents in the general surgery residency program. Designated surgical subspecialty residents were in post-graduate year 1 (PGY-1) or PGY-2 of training and included residents from cardiothoracic surgery, ophthalmology, orthopedic surgery, otolaryngology, plastic surgery, urology, and vascular surgery. Participants ranged from PGY-1 though PGY-5, with representation from all PGY-levels. Participants reported their PGY-level based on number of clinical years of surgical residency training completed with research time omitted. The mean (SD) age was 30.8 (3.2) years; 51.1% of the participants were female; 2.3% identified as Native American, 9.3% as Latino, 9.3% Black or African American, 34.9% Asian, and 39.5% White (see Fig. 5.9). Two participants preferred not to report their ethnicity. The self-reported prior video game experience of the participants ranged from 0 to 15 hours per week with mean 1.4 (SD 3.1) hours.



Demographics of Study Participants

Characteristic	n=43
Age (years), mean (SD)	30.8 (3.2)
Sex, n (%)	
Female	22 (51.1)
Male	21 (48.9)
Race/Ethnicity, n (%)	
Asian	15 (34.9)
Black or African American	4 (9.3)
Latino	4 (9.3)
Native American	1 (2.3)
White	17 (39.5)
Missing or prefer not to state	2 (4.7)
PGY-Level, n (%)	
PGY-1	17 (39.5)
PGY-2	11 (25.6)
PGY-3	9 (20.9)
PGY-4	2 (4.7)
PGY-5	4 (9.3)
General surgery resident status, n (%)	
General surgery categorical	27 (62.8)
General surgery non-designated preliminary	8 (18.6)
Designated preliminary†	8 (18.6)
Prior video game experience (hours/week), mean (SD)	1.4 (3.1)

Fig. 5.9 Demographics of study participants for the *ENTRUST* Inguinal Hernia EPA Assessment Pilot. Values reported as n (%) or mean (SD). *Acronymns*—Post-graduate Year (PGY) & standard deviation (SD). † Includes PGY-1 or PGY-2 cardiothoracic surgery, ophthalmology, orthopedic surgery, otolaryngology, plastic surgery, urology, and vascular surgery trainees in the general surgery residency program

5.4.1.2 Measures

- **Demographic Survey**—a demographic survey was created to collect information pertaining to the age, gender, ethnicity, PGY-level, surgical specialty, self-reported inguinal hernia operative case volume, and prior video game experience of participants.
- ***ENTRUST* Inguinal Hernia EPA Assessment**—an *ENTRUST* Inguinal Hernia EPA Assessment containing four cases was developed and piloted to collect initial validity evidence using Messick's framework [12, 28]. The case scenarios consisted of (1) an outpatient elective unilateral inguinal hernia, (2) an elective bilateral inguinal hernia, (3) an acutely incarcerated inguinal hernia, and (4)

a strangulated inguinal hernia. The four case scenarios for inguinal hernia, including all multiple-choice questions, were authored and iteratively developed by a board-certified general surgeon with formal training in surgical education. Cases were also carefully created in alignment with EPA descriptions and essential functions for inguinal hernia outlined by the American Board of Surgery [7]. The case content and multiple-choice questions were then reviewed and discussed by an expert panel ($n = 5$) of board-certified general surgeons representing a variety of practice settings. The case was iteratively revised based on this feedback, with the final case scenario reviewed and approved by the authors. The following scores logged by *ENTRUST* were analyzed to compare differences in performance between PGY-levels:

1. *Preoperative sub-score*—the score a participant received on just the simulation mode of *ENTRUST* for a single case scenario
2. *Preoperative total score*—the combined score for all four case scenarios that a participant received on just the simulation mode of *ENTRUST*
3. *Intraoperative sub-score*—the score a participant received on just the intraoperative questions during the question mode of *ENTRUST*
4. *Intraoperative total score*—the combined score for all four case scenarios that a participant received on just the intraoperative questions during the question mode of *ENTRUST*
5. *Postoperative sub-score*—the score a participant received on just the postoperative questions during the question mode of *ENTRUST*
6. *Postoperative total score*—the combined score for all four case scenarios that a participant received on just the postoperative questions during the question mode of *ENTRUST*
7. *Total case score*—the combined score for preoperative sub-score, intraoperative sub-score, and postoperative sub-score for a single case scenario
8. *Grand total score*—the combined total case score for all four case scenarios

5.4.1.3 Procedure

This study was conducted at a US-based academic institution in a proctored exam setting on laptop computers. Participants started by consenting to participate and then completing the demographic survey. After viewing a standardized video tutorial to orient participants to the *ENTRUST* platform, they then completed a non-scored practice case, which enabled them to interact firsthand with *ENTRUST* and familiarize themselves with the platform interface and functionality. Once finished with the practice case, participants completed the *ENTRUST* Inguinal Hernia EPA Assessment. The study protocol (#53137) was reviewed and approved by the Institutional Review Board at the authors' institution.

5.4.2 Data Analysis

Demographics are reported as mean and standard deviation for continuous variables and proportions for categorical variables. Descriptive statistics for total and sub-scores, including median and interquartile range, were calculated for each PGY-level. To assess the relationship between *ENTRUST* scores and resident level of training, Spearman rank correlations were calculated to examine the relationship between *ENTRUST* scores and ordinal PGY-level (1–5). These analyses were performed for *ENTRUST* grand total score, preoperative total score, intraoperative total score, and postoperative total score. Additionally, total case score, preoperative sub-score, intraoperative sub-score, and postoperative sub-score were calculated for individual case scenarios. Associations of *ENTRUST* grand total score and intraoperative total score with self-reported total inguinal hernia operative cases performed and video game experience were examined using Spearman rank correlations. Correlation between score and self-reported inguinal hernia operative experience was visualized using locally estimated scatterplot smoothing (LOESS). We assessed variations in scores between categorical and non-categorical PGY-1 and PGY-2 residents using Wilcoxon rank-sum tests.

A critical clinical decision-making action relevant for entrustment, specifically, the decision to attempt to manually reduce a hernia in the emergency department, was evaluated in additional analyses for the acutely incarcerated and strangulated inguinal hernia case scenarios. For these cases, the percentage of trainees selecting the correct answer was calculated by PGY-level. Wilcoxon rank-sum tests were calculated to examine whether participants who responded correctly on this critical action had significantly higher total and preoperative sub-scores than those who responded incorrectly. For this analysis, the preoperative score was adjusted to remove the score reward or penalty related to this critical action to eliminate the effect of the critical action itself on participant score. For all statistical tests, a significance threshold of $p < 0.05$ was utilized. All analyses were conducted using R v.4.0.2 (Vienna, Austria) [35].

5.4.3 Results

5.4.3.1 Relationship Between Performance and Prior Video Game Experience

Prior video game experience did not correlate with performance on *ENTRUST* (rho = 0.094, $p = 0.56$). This indicates that video game experience is not a prerequisite to successfully engage with *ENTRUST*.

5.4.3.2 Relationship Between Scores and Prior Operative Experience

Grand total score and intraoperative total score were correlated with self-reported prior inguinal hernia operative experience for participants (Fig. 5.10a, rho = 0.65, $p < 0.0001$, and Fig. 5.10b, rho = 0.59, $p < 0.0001$, respectively).

5.4.3.3 Relationships Between Scores and PGY-Level

ENTRUST Inguinal Hernia EPA Assessment grand total score was positively correlated with PGY-level (Fig. 5.11, rho = 0.64, $p < 0.0001$). Preoperative, intraoperative, and postoperative total scores were also positively correlated with PGY-level (preoperative, rho = 0.51, =; intraoperative, rho = 0.50, $p = 0.0006$; postoperative, rho = 0.32, $p = 0.038$). Total case scores were positively correlated with PGY-level for cases representing elective unilateral inguinal hernia (rho = 0.51, $p = 0.0004$), strangulated inguinal hernia (rho = 0.59, $p < 0.0001$), and elective bilateral inguinal hernia (rho = 0.52, $p = 0.0003$) (Fig. 5.12a). No statistically significant difference was found in acutely incarcerated inguinal hernia case total score by PGY-level (Fig. 5.12a, rho = 0.10, $p = 0.50$). Descriptive statistics for all *ENTRUST* Inguinal Hernia EPA Assessment scores are shown in Fig. 5.13.

For each of the four case scenarios, preoperative sub-score and intraoperative sub-score were additionally analyzed by PGY-level. Preoperative sub-scores were significantly correlated with PGY-level for all cases: elective unilateral inguinal hernia (rho = 0.43, $p = 0.004$), acutely incarcerated inguinal hernia (rho = 0.41, $p = 0.0066$), strangulated inguinal hernia (rho = 0.40, $p = 0.007$), and elective bilateral inguinal hernia (rho = 0.40, $p = 0.008$) (Fig. 5.12b). Intraoperative sub-scores were significantly correlated with PGY-level for the strangulated inguinal hernia (rho = 0.50, $p = 0.0007$) and elective bilateral inguinal hernia (rho = 0.54,

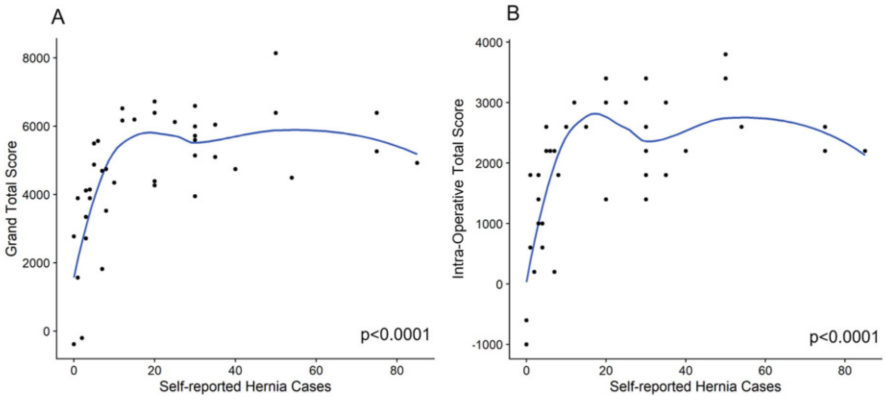


Fig. 5.10 Correlation of *ENTRUST* inguinal hernia EPA score performance to self-reported inguinal hernia operative case experience. (**a**) Grand total score. (**b**) Intraoperative total score

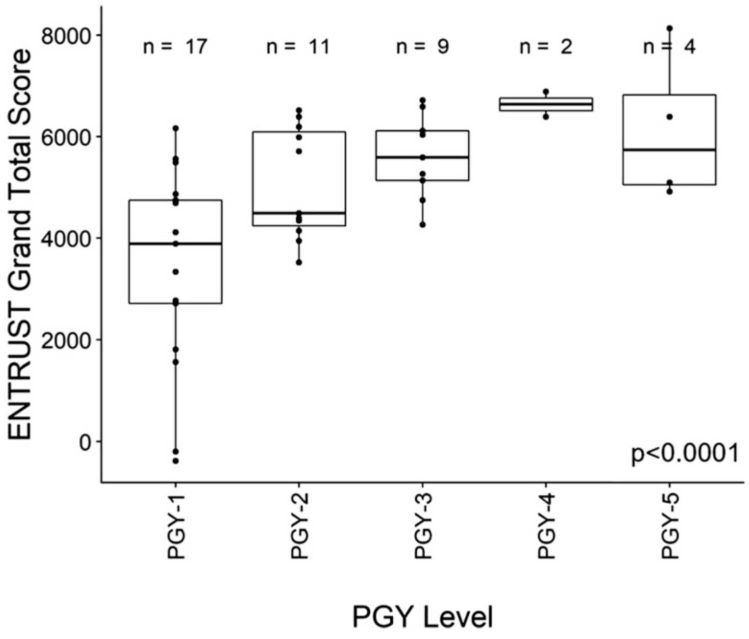


Fig. 5.11 *ENTRUST* Inguinal Hernia EPA Assessment grand total score by PGY-Level

$p = 0.0002$) case scenarios, but was not statistically significant for elective unilateral or acutely incarcerated inguinal hernia cases (Fig. 5.12c).

5.4.3.4 Categorical vs Non-categorical General Surgery Trainee Performance

Median grand total score for PGY-1 categorical general surgery trainees was higher than PGY-1 non-categorical surgery trainees (5190 vs 3178, $p = 0.014$). There was no statistically significant difference in score performance between PGY-2 categorical and non-categorical surgery trainees (6040 vs 4243, $p = 0.23$).

5.4.3.5 Critical Surgical Decision-Making Performance

For the critical clinical decision-making choice of whether to attempt manual reduction of an acutely incarcerated inguinal hernia in the emergency department, this was performed correctly by 100% of PGY-3 through PGY-5 residents, 88% of PGY-2 residents, and 67% of PGY-1 residents (Fig. 5.14a). Unadjusted total case score and preoperative sub-score for the acutely incarcerated inguinal hernia case were both significantly higher for those trainees correctly attempting manual

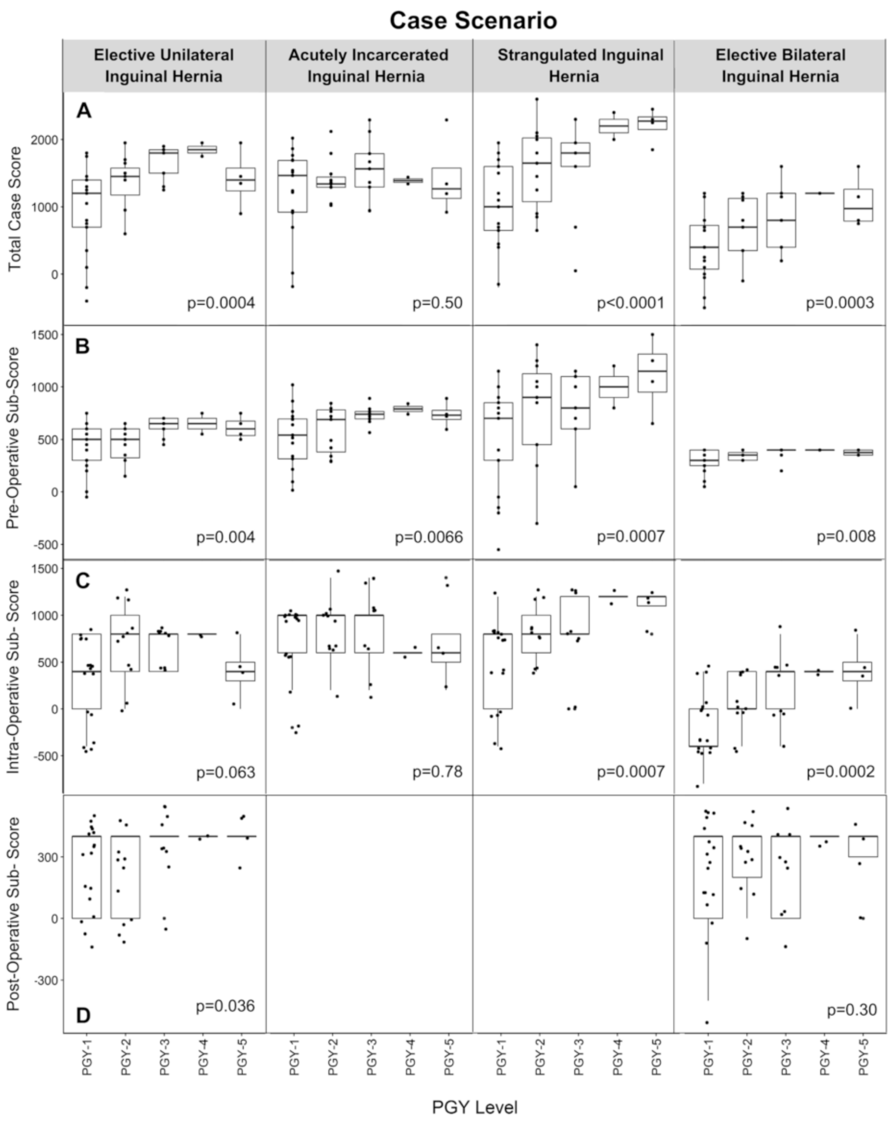
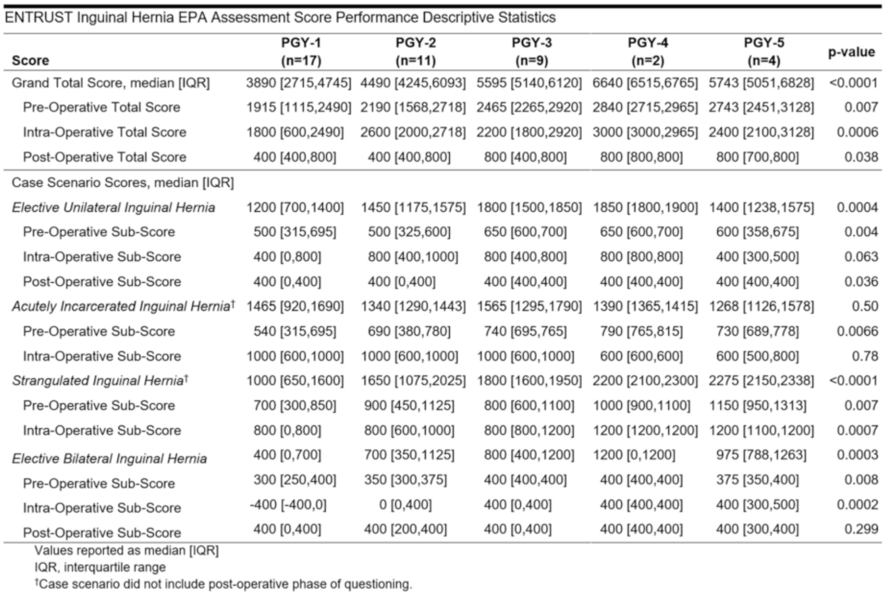


Fig. 5.12 *ENTRUST* Inguinal Hernia EPA Assessment case scenario total and sub-scores by PGY-Level. Total case score (**a**). Preoperative sub-scores (**b**). Intraoperative question sub-scores (**c**). Postoperative question sub-scores (**d**). The acutely incarcerated inguinal hernia and strangulated inguinal hernia case scenarios did not include postoperative questions



ENTRUST Inguinal Hernia EPA Assessment Score Performance Descriptive Statistics

Score	PGY-1 (n=17)	PGY-2 (n=11)	PGY-3 (n=9)	PGY-4 (n=2)	PGY-5 (n=4)	p-value
Grand Total Score, median [IQR]	3890 [2715,4745]	4490 [4245,6093]	5595 [5140,6120]	6640 [6515,6765]	5743 [5051,6828]	<0.0001
Pre-Operative Total Score	1915 [1115,2490]	2190 [1568,2718]	2465 [2265,2920]	2840 [2715,2965]	2743 [2451,3128]	0.007
Intra-Operative Total Score	1800 [600,2490]	2600 [2000,2718]	2200 [1800,2920]	3000 [3000,2965]	2400 [2100,3128]	0.0006
Post-Operative Total Score	400 [400,800]	400 [400,800]	800 [400,800]	800 [800,800]	800 [700,800]	0.038
Case Scenario Scores, median [IQR]						
Elective Unilateral Inguinal Hernia	1200 [700,1400]	1450 [1175,1575]	1800 [1500,1850]	1850 [1800,1900]	1400 [1238,1575]	0.0004
Pre-Operative Sub-Score	500 [315,695]	500 [325,600]	650 [600,700]	650 [600,700]	600 [358,675]	0.004
Intra-Operative Sub-Score	400 [0,800]	800 [400,1000]	800 [400,800]	800 [800,800]	400 [300,500]	0.063
Post-Operative Sub-Score	400 [0,400]	400 [0,400]	400 [400,400]	400 [400,400]	400 [400,400]	0.036
Acutely Incarcerated Inguinal Hernia†	1465 [920,1690]	1340 [1290,1443]	1565 [1295,1790]	1390 [1365,1415]	1268 [1126,1578]	0.50
Pre-Operative Sub-Score	540 [315,695]	690 [380,780]	740 [695,765]	790 [765,815]	730 [689,778]	0.0066
Intra-Operative Sub-Score	1000 [600,1000]	1000 [600,1000]	1000 [600,1000]	600 [600,600]	600 [500,800]	0.78
Strangulated Inguinal Hernia†	1000 [650,1600]	1650 [1075,2025]	1800 [1600,1950]	2200 [2100,2300]	2275 [2150,2338]	<0.0001
Pre-Operative Sub-Score	700 [300,850]	900 [450,1125]	800 [600,1100]	1000 [900,1100]	1150 [950,1313]	0.007
Intra-Operative Sub-Score	800 [0,800]	800 [600,1000]	800 [800,1200]	1200 [1200,1200]	1200 [1100,1200]	0.0007
Elective Bilateral Inguinal Hernia	400 [0,700]	700 [350,1125]	800 [400,1200]	1200 [0,1200]	975 [788,1263]	0.0003
Pre-Operative Sub-Score	300 [250,400]	350 [300,375]	400 [400,400]	400 [400,400]	375 [350,400]	0.008
Intra-Operative Sub-Score	-400 [-400,0]	0 [0,400]	400 [0,400]	400 [400,400]	400 [300,500]	0.0002
Post-Operative Sub-Score	400 [0,400]	400 [200,400]	400 [0,400]	400 [400,400]	400 [300,400]	0.299

Values reported as median [IQR]
IQR, interquartile range
†Case scenario did not include post-operative phase of questioning.

Fig. 5.13 *ENTRUST* Inguinal Hernia EPA Assessment score performance descriptive statistics. Values reported as median [IQR]. Acronym—interquartile range (IQR). † Case scenario did not include postoperative phase of questioning

reduction ($p = 0.007$ and $p < 0.0001$, respectively). However, these differences in total case score and preoperative sub-score were not statistically significant when scores were adjusted to remove the scoring impact of the decision to manually reduce the incarcerated hernia ($p = 0.11$ and $p = 0.17$, respectively).

For the decision of whether to attempt manual reduction of a strangulated inguinal hernia, this was performed correctly by 100% of PGY-3, PGY-4, and PGY-5 residents, 91% of PGY-2 residents, and 75% of PGY-1 residents (Fig. 5.14b). Unadjusted total case score and preoperative sub-score for the strangulated inguinal hernia case were significantly higher for those trainees correctly deciding not to attempt manual reduction ($p = 0.009$ and $p = 0.0019$, respectively). After adjustment to remove the scoring impact of the decision to manually reduce the strangulated hernia, a statistically significant difference in preoperative sub-score remained between those who attempted reduction and those who did not attempt reduction ($p = 0.032$).

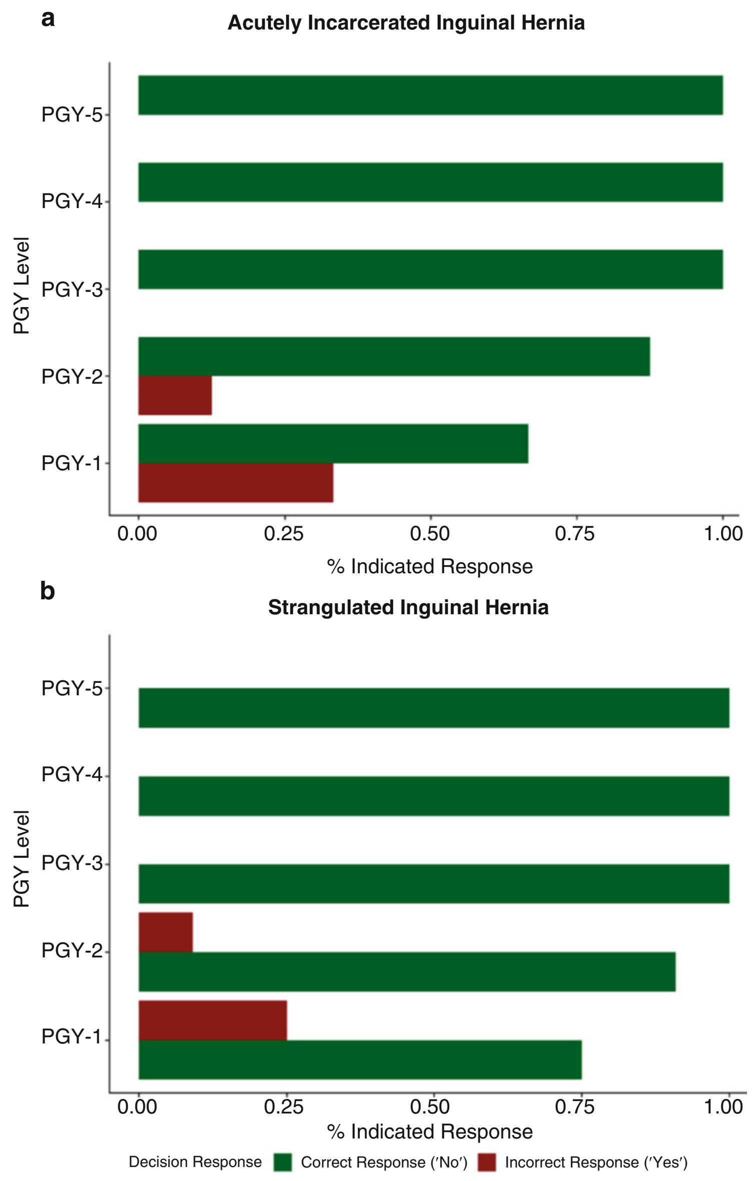


Fig. 5.14 *ENTRUST* Inguinal Hernia EPA Assessment grand total score by PGY-Level

5.5 Discussion

5.5.1 Lessons Learned from the Co-design Process

We identified a number of insights and lessons learned throughout the *ENTRUST* co-design process as follows:

- **Early development of tools to empower stakeholders**—one common technique within game development is to abstract content, design, and logic from core game engine code (e.g., through use of a level editor to create and edit levels or external script files to maintain game parameters, logic, and character dialogues). This is typically done with the intent of modularizing aspects game development as well as making that development more accessible to individuals with limited programming skills. We found this approach to be especially critical for our co-design process since stakeholders tend to have no prior programming experience, making it difficult to add or update content in the game otherwise. However, of equal importance was the creation of sophisticated tools that empowered stakeholders to easily create, edit, and view changes to the serious game in real time. For instance, during the *ENTRUST* design and development process, we initially abstracted the creation and management of case scenarios to a spreadsheet template. While this did enable stakeholders to create content for the game, it also effectively disempowered them since working with a spreadsheet was cumbersome, difficult for reusability (e.g., required reentering default orders and other repeated details for every new case scenario), and forced stakeholders to wait a substantial amount of time to view changes—as a programmer had to input spreadsheet information into the game. This process also introduced a lot of confusion and communication overhead as a by-product. These issues were not remedied until the creation of an authoring tool that enabled stakeholders to quickly and easily edit case scenario information directly in the *ENTRUST* game database. By enabling stakeholders without programming experience to easily create, edit, and view changes to *ENTRUST* in real time, we empowered them to be more directly involved with and provide input into the design and development process. This in turn greatly increased productivity, reduced errors in identifying and addressing latent needs, and ultimately improved overall development speed. Importantly, it also enabled new stakeholders (such as the College of Surgeons of East, Central and Southern Africa) to get involved with various aspects of the project far more easily. This insight also falls in line with existing research, which has highlighted the importance of empowering stakeholders for successful co-design [2].
- **Benefits of frequent review meetings with stakeholders**—another key aspect of *ENTRUST*'s successful co-design and development was the incorporation of weekly review meetings with stakeholders. Initially, *ENTRUST*'s co-design and development involved monthly review meetings with stakeholders. However, the long duration between co-design/development and stakeholder review proved

problematic as it often led to errors in identifying the appropriate items for the sprint and product backlogs. Switching to a more frequent weekly review meeting with stakeholders at the end of each sprint helped to greatly reduce such errors. While frequent review meetings are not always feasible due to time constraints for stakeholders, some form of frequent communication and review (even asynchronously) can result in similar benefits [14, 19, 36, 41].

5.5.2 Validity Evidence for Assessing Clinical Decision-Making Skills

Our pilot data indicates that *ENTRUST* score performance is correlated to PGY-level and inguinal hernia operative experience, i.e., there was a statistically significant increase in total score with successively higher PGY-level. This trend was observed for grand total score, preoperative total score, intraoperative total score, and individual total case scores. However, while surgical decision-making skills tend to develop over time with increasing PGY-level, it is not a strictly time-based construct, and the variation in score within PGY-level may be explained by differences in clinical decision-making ability and readiness for entrustment. Theoretically, a junior resident with high *ENTRUST* performance who objectively demonstrates surgical decision-making competence may be entrusted with greater autonomy earlier than a senior resident with low *ENTRUST* score performance for a particular EPA domain. Thus, *ENTRUST* has potential to be employed as a tool to inform entrustment decisions as surgical training shifts from a time-based model toward a competency-based paradigm.

Additionally, as demonstrated by the clinical decision-making surrounding whether or not to attempt manual reduction of an incarcerated or strangulated inguinal hernia, *ENTRUST* also holds potential to evaluate and query specific key surgical decision-making points important in determining readiness or lack of readiness for entrustment. By logging all trainee actions and querying specific decisions, *ENTRUST* may assist program directors and surgical educators in assigning ABS EPA Levels, independent of PGY-level. This information can be used to inform decisions on entrustment and autonomy.

Ultimately, this study provides initial validity evidence for use of *ENTRUST* as an objective measure of surgical decision-making for EPAs. Content evidence for the case scenarios was established by alignment of case content with published ABS EPA descriptions and essential functions [7], expert review, and group consensus of case content and scoring algorithm. The ability of the *ENTRUST* assessment to discriminate between PGY-levels, as well as its correlation to inguinal hernia operative case experience provides evidence of its relationship to other established variables in surgical education. Importantly, there was also no difference in score performance based on prior video game experience, indicating that video game experience is not required to utilize *ENTRUST* effectively.

5.6 Limitations

There are some limitations of this work, particularly with the pilot study. This includes the single institution study design and self-reported inguinal hernia operative experience. Additionally, there were notably lower numbers of participants at higher PGY-levels (see Fig. 5.9). All of these could impact generalizability of the results to some extent.

5.7 Future Work

5.7.1 ENTRUST Development

Future development plans for *ENTRUST* include expansion of the platform to encompass all ABS general surgery EPAs, as well as creation of additional environments, assets, and functionality to accommodate higher acuity case scenarios situated in the trauma bay and ICU settings. Ultimately, this will enable *ENTRUST* to evaluate a broader spectrum of trainees' readiness for entrustment in a more diverse range of scenarios. We also plan to make *ENTRUST* more scalable for distribution by adding additional functionality and security to manage multiple organizations and allow them to maintain their own examinee assessment data and order, case, and exam libraries. Finally, we plan to extend *ENTRUST* beyond just a game-based assessment platform into a game-based learning platform as well. This will include the development of new tools to visualize player actions both individually and in aggregate to support self-regulated learning [3].

5.7.2 ENTRUST Research

Future research directions include collection and analysis of additional validity evidence for *ENTRUST* using Messick's unified framework of construct validity, including response process evidence, internal structure, and consequences [28]. In future studies, we intend to further investigate relationship of *ENTRUST*'s assessment capabilities/scores to other objective assessment variables such as ACGME Case Logs, ABS Inservice Training Exam (ABSITE) scores, Accreditation Council for Graduate Medical Education (ACGME) Milestones, and ABS board pass rates. Additionally, we plan to correlate performance on *ENTRUST* to individual trainee performance on micro-assessments such as SIMPL or other platforms for actual clinical interactions. Results from this pilot will inform the design of future multi-institutional studies featuring a larger set of case scenarios for the Inguinal Hernia EPA to further collect validity evidence, conduct standard setting, and map gameplay patterns and specific key decision-making actions to EPA levels and readiness for entrustment.

5.8 Conclusion

This chapter presented the design of and preliminary validity evidence for *ENTRUST*—a virtual patient authoring and serious game-based assessment platform to deploy rigorous, case-based patient simulations for evaluation of EPAs. Our results with *ENTRUST* demonstrate feasibility and initial validity evidence for objective assessment of surgical decision-making for inguinal hernia EPA. We also discussed insights and lessons learned from the co-design and development of *ENTRUST*, as well as highlighted future directions for the game-based platform. Importantly, the *ENTRUST* authoring and assessment platform holds potential to inform readiness of entrustment for American Board of Surgery EPAs in the future and to support the ongoing transformation of surgical education to a competency-based paradigm.

Acknowledgments The authors would like to thank Yulin Cai, Ruonan Chen, Pichy Jumpholwong, Ananya Anand, and Sherry Wren for their contributions to the development of the ENTRUST platform.

References

1. ABS E-News – Spring 2018. In: News from the American Board of Surgery. The American Board of Surgery (2018). http://www.absurgery.org/quicklink/absnews/absupdate0518.html#epa. Accessed 12 Jan 2021
2. Ardito C., Buono P., Costabile M.F., Lanzilotti R., Piccinno A.: End users as co-designers of their own tools and products. J. Visual Lang. Comput. **23**(2), 78–90 (2012)
3. Barnard-Brak, L., Paton, V.O., Lan, W.Y.: Profiles in self-regulated learning in the online learning environment. Int. Rev. Res. Open Distrib. Learn. **11**(1), 61–80 (2010)
4. Berman, N.B., Durning, S.J., Fischer, M.R., Huwendiek, S., Triola, M.M.: The role for virtual patients in the future of medical education. Acad. Med. **91**(9), 1217–1222 (2016)
5. Bird, M., McGillion, M., Chambers, E.M., Dix, J., Fajardo, C.J., Gilmour, M., Levesque, K., Lim, A., Mierdel, S., Ouellette, C., Polanski, A.N., Reaume, S.V., Whitmore, C., Carter, N.: A generative co-design framework for healthcare innovation: development and application of an end-user engagement framework. Res. Involv. Engagem. **7**(1), 1–12 (2021)
6. Bohnen, J.D., George, B.C., Williams, R.G., Schuller, M.C., DaRosa, D.A., Torbeck, L., Mullen, J.T., Meyerson, S.L., Auyang, E.D., Chipman, J.G., Choi, J.N.: The feasibility of real-time intraoperative performance assessment with SIMPL (system for improving and measuring procedural learning): early experience from a multi-institutional trial. J. Surg. Educ. **73**(6), e118–e130 (2016)
7. Brasel, K.J., Klingensmith, M.E., Englander, R., Grambau, M., Buyske, J., Sarosi, G., Minter, R.: Entrustable professional activities in general surgery: development and implementation. J. Surg. Educ. **76**(5), 1174–1186 (2019)
8. Burkett, I.: An Introduction to Co-design, vol. 12. Knode, Sydney (2012)
9. Charlier, N.: Game-based assessment of first aid and resuscitation skills. Resuscitation **82**(4), 442–446 (2011)

10. Chon, S.H., Timmermann, F., Dratsch, T., Schuelper, N., Plum, P., Berlth, F., Datta, R.R., Schramm, C., Haneder, S., Späth, M.R., Dübbers, M., Kleinert, J., Raupach, T., Bruns, C., Kleinert, R.: Serious games in surgical medical education: a virtual emergency department as a tool for teaching clinical reasoning to medical students. JMIR Serious Games **7**(1), 1–11 (2019)
11. Cianciolo, A.T., Kegg, J.A.: Behavioral specification of the entrustment process. J. Grad. Med. Educ. **5**(1), 10–12 (2013)
12. Cook, D.A., Zendejas, B., Hamstra, S.J., Hatala, R., Brydges, R.: What counts as validity evidence? Examples and prevalence in a systematic review of simulation-based assessment. Adv. Health Sci. Educ. **19**(2), 233–250 (2014)
13. de Klerk, S., Kato, P.M.: The future value of serious games for assessment: Where do we go now?. J. Appl. Testing Technol. **18**(S1), 32–37 (2017)
14. Domecq, J.P., Prutsky, G., Elraiyah, T., Wang, Z., Nabhan, M., Shippee, N., Brito, J.P., Boehmer, K., Hasan, R., Firwana, B., Erwin, P., Eton, D., Sloan, J., Montori, V., Asi, N., Dabrh, A.M.A., Murad, M.H.: Patient engagement in research: a systematic review. BMC Health Serv. Res. **14**(1), 1–9 (2014)
15. Eaton, M., Scully, R., Schuller, M., Yang, A., Smink, D., Williams, R.G., Bohnen, J.D., George, B.C., Fryer, J.P., Meyerson, S.L.: Value and barriers to use of the SIMPL tool for resident feedback. J. Surg. Educ. **76**(3), 620–627 (2019)
16. Ferreira-Brito, F., Fialho, M., Virgolino, A., Neves, I., Miranda, A.C., Sousa-Santos, N., Caneiras, C., Carrico, L., Verdelho, A., Santos, O.: Game-based interventions for neuropsychological assessment, training and rehabilitation: which game-elements to use? A systematic review. J. Biomed. Inform. **98**, 103287 (2019)
17. George, B.C., Bohnen, J.D., Williams, R.G., Meyerson, S.L., Schuller, M.C., Clark, M.J., Meier, A.H., Torbeck, L., Mandell, S.P., Mullen, J.T., Smink, D.S.: Readiness of US general surgery residents for independent practice. Ann. Surg. **266**(4), 582–594 (2017)
18. George, B.C., Bohnen, J.D., Schuller, M.C., Fryer, J.P.: Using smartphones for trainee performance assessment: a SIMPL case study. Surgery **167**(6), 903–906 (2020)
19. Guise, J.M., O'Haire, C., McPheeters, M., Most, C., LaBrant, L., Lee, K., Cottrell, E.K.B., Graham, E.: A practice-based tool for engaging stakeholders in future research: a synthesis of current practices. J. Clin. Epidemiol. **66**(6), 666–674 (2013)
20. Hwang, G.J., Chang, C.Y. Facilitating decision-making performances in nursing treatments: a contextual digital game-based flipped learning approach. Interactive Learn. Environ. **31**(1), 1–16 (2020)
21. Johnsen, H.M., Fossum, M., Vivekananda-Schmidt, P., Fruhling, A., Slettebø, Å.: Teaching clinical reasoning and decision-making skills to nursing students: design, development, and usability evaluation of a serious game. Int. J. Med. Inform. **94**, 39–48 (2016)
22. Kalyuga, S., Plass, J.L.: Evaluating and managing cognitive load in games. In: Handbook of Research on Effective Electronic Gaming in Education, pp. 719–737. IGI Global, Pennsylvania (2009)
23. Lagro, J., van de Pol, M.H., Laan, A., Huijbregts-Verheyden, F.J., Fluit, L.C., Rikkert, M.G.O.: A randomized controlled trial on teaching geriatric medical decision making and cost consciousness with the serious game GeriatriX. J. Am. Med. Direct. Assoc. **15**(12), e1–957.e6 (2014)
24. Liebert, C.A., Mazer, L., Merrell, S.B., Lin, D.T., Lau, J.N.: Student perceptions of a simulation-based flipped classroom for the surgery clerkship: a mixed-methods study. Surgery **160**(3), 591–598 (2016)
25. Liebert, C.A., Melcer, E.F., Keehl, O., Eddington, H., Trickey, A.W., Lee, M., Tsai, J., Camacho, F., Merrell, S.B., Korndorffer, Jr. J.R., Lin, D.T.: Validity evidence for ENTRUST as an assessment of surgical decision-making for the inguinal hernia entrustable professional activity (EPA). J. Surg. Educ. **79**(6), e202–e212 (2022)
26. Lin, D.T., Park, J., Liebert, C.A., Lau, J.N.: Validity evidence for surgical improvement of clinical knowledge ops: a novel gaming platform to assess surgical decision making. Am. J. Surg. **209**(1), 79–85 (2015)

27. Mavridis, A., Tsiatsos, T.: Game-based assessment: investigating the impact on test anxiety and exam performance. J. Comput. Assist. Learn. **33**(2), 137–150 (2017)
28. Messick, S.: Standards of validity and the validity of standards in performance asessment. Educ. Measur. Issues Pract. **14**(4), 5–8 (1995)
29. Middeke, A., Anders, S., Schuelper, M., Raupach, T., Schuelper, N.: Training of clinical reasoning with a serious game versus small-group problem-based learning: a prospective study. PLoS One **13**(9), e0203851 (2018)
30. Nemirovsky, D.R., Garcia, A.J., Gupta, P., Shoen, E., Walia, N.: Evaluation of surgical improvement of clinical knowledge ops (SICKO), an interactive training platform. J. Digit. Imag. **34**(4), 1067–1071 (2021)
31. New model of surgical resident autonomy coming in 2023. In: ACS Clinical Congress News (2021). Published October 23, 2021. Accessed 21 Jan 2022. https://www.acsccnews.org/new-model-of-surgical-resident-autonomy-coming-in-2023/
32. Nikolian, V.C., Sutzko, D.C., Georgoff, P.E., Matusko, N., Boniakowski, A., Prabhu, K., Church, J.T., Thompson-Burdine, J., Minter, R.M., Sandhu, G.: Improving the feasibility and utility of OpTrust–a tool assessing intraoperative entrustment. Am. J. Surg. **216**(1), 13–18 (2018)
33. Oestreich, J.H., Guy, J.W.: Game-based learning in pharmacy education. Pharmacy **10**(1), 11 (2022)
34. Plass, J.L., Moreno, R., Brünken, R.: Cognitive Load Theory. Cambridge University Press, Cambridge (2010)
35. R Core Team: R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna (2013). https://www.R-Project.org/
36. Salsberg, J., Parry, D., Pluye, P., Macridis, S., Herbert, C.P., Macaulay, A.C.: Successful strategies to engage research partners for translating evidence into action in community health: a critical review. J. Environ. Pub. Health **2015**, 1–15 (2015)
37. Sánchez, R., Rodríguez, O., Rosciano, J., Vegas, L., Bond ,V., Rojas, A., Sanchez-Ismayel, A.: Robotic surgery training: construct validity of Global Evaluative Assessment of Robotic Skills (GEARS). J. Robot. Surg. **10**(3), 227–231 (2016)
38. Sandhu, G., Nikolian, V.C., Magas, C.P., Stansfield, R.B., Sutzko, D.C., Prabhu, K., Matusko, N., Minter, R.M.: OpTrust: validity of a tool assessing intraoperative entrustment behaviors. Ann. Surg. **267**(4), 670–676 (2018)
39. Schuler, D., Namioka, A.: Participatory Design: Principles and Practices. CRC Press, Boca Raton (1993)
40. Seelow, D.: The art of assessment: using game based assessments to disrupt, innovate, reform and transform testing. J. Appl. Testing Technol. **20**(S1), 1–16 (2019)
41. Slattery, P., Saeri, A.K., Bragge, P.: Research co-design in health: a rapid overview of reviews. Health Res. Policy Syst. **18**(1), 1–13 (2020)
42. Steen, M.: Co-design as a process of joint inquiry and imagination. Des. Issues **29**(2), 16–28 (2013)
43. Ten Cate, O., Scheele, F.: Competency-based postgraduate training: can we bridge the gap between theory and clinical practice? Acad. Med. **82**(6), 542–547 (2007)

44. Ten Cate, O., Chen, H.C., Hoff, R.G., Peters, H., Bok, H., van der Schaaf, M.: Curriculum development for the workplace using entrustable professional activities (EPAs): AMEE guide no. 99. Med. Teach. **37**(11), 983–1002 (2015)
45. Ten Cate, O., Carraccio, C., Damodaran, A., Gofton, W., Hamstra, S.J., Hart, D.E., Richardson, D., Ross, S., Schultz, K., Warm, E.J., Whelan, A.J., Schumacher, D.J.: Entrustment decision making: extending Miller's pyramid. Acad. Med. **96**(2), 199–204 (2021)
46. Vallejo, V., Wyss, P., Rampa, L., Mitache, A.V., Müri, R.M., Mosimann, U.P., Nef, T.: Evaluation of a novel Serious Game based assessment tool for patients with Alzheimer's disease. PLoS One **12**(5), e0175999 (2017)
47. Vassiliou, M.C., Feldman, L.S., Andrew, C.G., Bergman, S., Leffondré, K., Stanbridge, D., Fried, G.M.: A global assessment tool for evaluation of intraoperative laparoscopic skills. Am. J. Surg. **190**(1), 107–113 (2005)
48. Verma, V., Baron, T., Bansal, A., Amresh, A.: Emerging practices in game-based assessment. In: Game-Based Assessment Revisited, pp. 327–346. Springer, Cham (2019)
49. Yeo, H.L., Dolan, P.T., Mao, J., Sosa, J.A.: Association of demographic and program factors with American Board of Surgery qualifying and certifying examinations pass rates. JAMA Surg. **155**(1), 22–30 (2020)

Chapter 6
Engineering Adaptive Serious Games Using Machine Learning

Michael A. Miljanovic and Jeremy S. Bradbury

Abstract The vast majority of serious games (SGs) do not feature any form of machine learning (ML); however, there is a recent trend of developing SGs that leverage ML to assess learners and to make automated adaptations during game play. This trend allows serious games to be personalized to the learning needs of the player and can be used to reduce frustration and increase engagement. In this chapter, we will discuss the development of new ML-based SGs and present a generalized model for evolving existing SGs to use ML without needing to rebuild the game from scratch. In addition to describing how to engineer ML-based SGs, we also highlight five common challenges encountered during our own development experiences, along with advice on how to address these challenges. Challenges discussed include selecting data for use in an ML model for SGs, choosing game elements to adapt, solving the cold start problem, determining the frequency of adaptation, and testing that an adaptive game benefits from learning.

Keywords Adaptation · Machine learning · Personalized learning · Serious games

6.1 Introduction

Serious games (SGs), also known as educational games, are games designed with a purpose other than entertainment. Most commonly, these are video games that have been made to help aid with learning in a variety of contexts, including science, healthcare, business, and more [15]. Developers of SGs have a difficult challenge: to create a product that is first and foremost a game that not only engages players but equally importantly improves their understanding and competency with non-game content. Game-based learning (GBL) using SGs differs from the related

M. A. Miljanovic (✉) · J. S. Bradbury
Ontario Tech University, Oshawa, ON, Canada
e-mail: michael.miljanovic@ontariotechu.ca; jeremy.bradbury@ontariotechu.ca

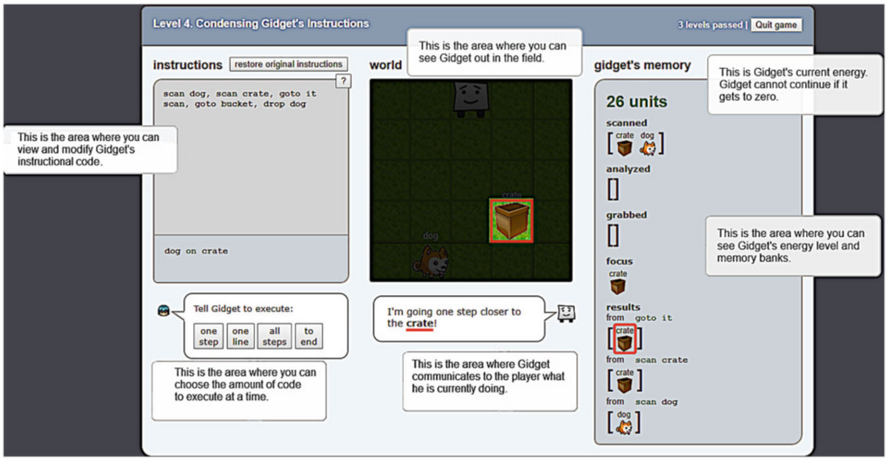
K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_6

field of gamification [1], where game elements such as badges, points, and avatars are applied to non-game contexts. Using these definitions, an application such as Duolingo would not be considered an SG since it was not designed as a game, but it would be considered an example of gamification because it includes elements such as achievements and a leaderboard to encourage players to use the app frequently. On the other hand, IBM's CityOne is an SG designed to help players learn about transportation, environmental, and logistical issues that are relevant to city leaders and businesses. The distinction between these applications is important—Duolingo seeks to use gamification to improve engagement, while CityOne is intended to be intrinsically motivating by its nature of being a game.

The target audience for SGs can include any demographic, but the players of these games are considered learners in the context of the game's educational environment. One of the greatest challenges of serious games is learner assessment, which often takes the form of stealth assessment [13] in order to avoid disrupting the game play experience.

The importance of learner assessment is even more significant for the purposes of adaptation. Adaptive SGs are a new form of SG that use learner data collected before or during game play to modify the game automatically [14]. This strategy can be used to reduce the difficulty level for players who are struggling with the game's content or to increase the challenge for players who have demonstrated a high level of competence. The concept of dynamic difficulty adjustment has already been implemented in entertainment games like Left 4 Dead, which has a "Director" that constantly adjusts the game as it assesses the player's performance (Figs. 6.1 and 6.2).

Automatically assessing learning is a particularly difficult task, which has led to the use of techniques such as machine learning (ML) to analyze learners based on



```
scan dog, scan crate, goto it
scan, goto bucket, drop dog
```

```
dog on crate
```

This is the area where you can see Gidget out in the field.

This is the area where you can view and modify Gidget's instructional code.

This is Gidget's current energy. Gidget cannot continue if it gets to zero.

This is the area where you can see Gidget's energy level and memory banks.

I'm going one step closer to the crate!

This is the area where you can choose the amount of code to execute at a time.

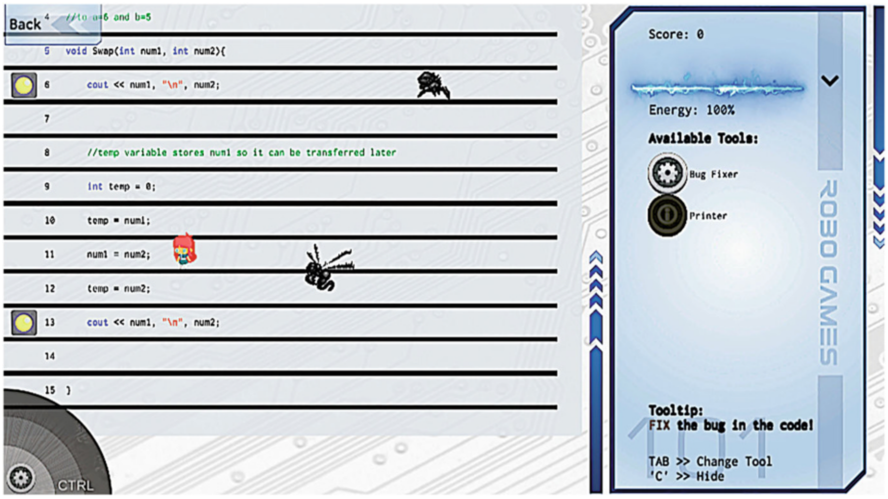
This is the area where Gidget communicates to the player what he is currently doing.

from goto it

from scan crate

from scan dog

Fig. 6.1 The GidgetML game—an ML-enhanced version of the Gidget programming game that personalizes the game play and game elements to meet the learning needs of the player



```
//to a=6 and b=5
void Swap(int num1, int num2){
    cout << num1, "\n", num2;

    //temp variable stores num1 so it can be transferred later
    int temp = 0;
    temp = num1;
    num1 = num2;
    temp = num2;
    cout << num1, "\n", num2;

}
```

```
Score: 0
Energy: 100%
Available Tools:
Bug Fixer
Printer

Tooltip:
FIX the bug in the code!

TAB >> Change Tool
'C' >> Hide
```

Fig. 6.2 The RoboBug game—an ML-enhanced program debugging game that personalizes the game play and the hint system to meet the learning needs of the player

data provided to a player experience model [3]. A variety of ML algorithms might be appropriate for this purpose, but each presents its own challenges. Different learning models require various forms of data, and it can be challenging to determine what data points are relevant to assessing learner competency.

This chapter will discuss some of the challenges that come from engineering adaptive SGs, and to support our discussion of engineering adaptive serious games, we will use two case studies throughout this chapter to illustrate and provide context to the issues with using ML for adaptation. Both case studies are examples of SGs for computer science education; however, the lessons learned can be generalized to other domains.

The first case study is an SG called GidgetML [12], which is a modified version of the serious programming game Gidget created by Michael Lee and Amy Ko [5, 6]. The game requires players to correct bugs in code to navigate the Gidget character and provide instructions to complete various tasks on a two-dimensional grid map. GidgetML uses an ML algorithm that takes information about failed player attempts and code efficiency to categorize players into different competency categories. It then modifies subsequent levels of the game based on the player's competency—players demonstrating higher levels of competence start each level with additional bugs in their starter code and stricter limitations on how efficient their solution must be to be accepted as correct.

The second case study is an SG called RoboBUG [9], which was designed to help players gain familiarity with various techniques for debugging source code. Players navigate their character through lines of code akin to navigating a text editor and must identify lines of code that contain semantic or logical errors. Two adaptive variants of RoboBUG were created, each using data including time on task and failure counts to categorize players [8]. One adaptive variant modified the game's code and obstacles to increase or decrease challenge by obfuscating bugs or making them easier to see, while the second variant provided an increasing number of hints to players depending on their assessed competency level.

6.2 Data Models

With a traditional non-game-based learning assessment, it can be difficult to make judgments about a learner's competence—questions can be phrased poorly, or the answers to a multiple-choice question might be deduced by context clues. However, in an educational game, the multitude of actions taken by a player over time can be examined in a variety of different ways in order to gauge their performance. For example, one might consider how much idle time elapses during game play, how much time a player spends tinkering with different parts of a puzzle, or how many mistakes are made before the player is able to progress in a game. These data features and others can be sampled across a game play session and used to predict a player's level of competence based on a well-designed game play model.

The theoretical basis for modelling player behavior in an SG lies in competence-based knowledge space theory (CbKST) [4]. This theory distinguishes between assessment problems/activities and the desired competencies that pertain to them. A given assessment problem might have a number of different solutions that pertain to different competencies, and similarly any given competence might be able to provide a solution to many different assessment problems. In addition, there are some competencies which are composed of other competencies, representing a skill that can only be developed after others have already been learned.

As an example, consider the assessment problem of summing a list of numbers using a program. One solution might involve iterating through each element of the list, adding that number to an initial sum of zero, and examining the sum after all elements have been added. This solution would require the learner to demonstrate competency in arithmetic and iteration; however, this is not the only way to solve the problem. Another solution could be a recursive approach to adding the numbers together; although this solution might be equally correct, it requires an understanding of recursion that is not necessary to solve the problem. Therefore, we see that many assessment problems may be linked to many different competencies and that not all of these links are equal—clearly the ability to sum two numbers is required, but knowledge of iteration or recursion is each optional.

6.2.1 Game Task Model

In the context of educational games, most game "tasks" are intended to foster the development of or test a player for specific competencies. Thus, to assess any player, one must first examine each task (which is a form of the assessment problem) to determine the linked competencies that are prerequisites or otherwise associated.

This can be particularly difficult if there are multiple different approaches that might be suitable for overcoming the task, in which case, it is necessary to identify the approach selected by the player. However, restrictions placed upon the player may be used to limit the space of solutions that could be used to solve a problem. For example, in a block-based coding game, removing any "loop" blocks would force the player to rely on a recursive solution to solve the problem and eliminate the competency relationship between the task and iteration.

Thus, the first necessary step for applying adaptivity to an educational game is to consider the tasks in the game and determine the relevant competencies for each. This means identifying the necessary prerequisite competencies needing to attempt a solution for a task and the desired outcomes from completing the task. Ideally, the first tasks in the game should require as few prerequisites as possible (or zero), and subsequent tasks should build upon what has been learned from the previous ones. If there are any gaps in competencies between tasks, then this should be addressed by the introduction of additional tasks or some other way to help the player develop the necessary prerequisite. For example, if a task requires a prerequisite competency that the player would only learn if they completed an optional game play task, then that task should either be made mandatory, or another task should be added that can compensate for the missing exercise.

6.2.2 Player/Learner Model

Once there is a suitable model for game tasks, it then becomes possible to create a model for the player based on CbKST. A player's behavior in-game can be logged for both post-game assessment as well as in-game assessment, although the latter presents a greater challenge due to the time sensitivity of the evaluation and incompleteness of the data. A best estimate of a player's competency should be gleaned based on their performance on tasks, whether successful or unsuccessful. Game play data has a high degree of granularity, and while some games may be suited to simply differentiating between successful and unsuccessful attempts at a task, it is often preferable to examine specifically how a player attempts to complete a task and evaluate the player based on both their approach and whether or not it was successful. Two students who both succeed or both fail at a task will not necessarily be at an equal level of competency, considering their performance and how efficient or how close they were to solving the problem, as well as how easily they were able to reach that solution can provide further insight as to their competency level.

6.3 A Generalized Methodology for Evolving Existing Serious Games to Use ML

To support the development of new adaptive SGs, we have devised a methodology to guide the process of adding automatic adaptation to existing non-adaptive SGs [10]. The methodology has four key phases (see Fig. 6.3):

1. **Identify** a potential adaptive game
2. **Model** game play tasks and the ability of learners
3. **Build** ML and supporting functionality into the existing code base.
4. **Evaluate** the benefits of the adaptive modifications.

6.3.1 Identify

A number of technical and learning factors should be considered when identifying if a serious game is appropriate for adding adaptation via ML.

The three most important technical factors are source code availability, software quality, and game playability. First, the source code will need to be publicly available. This is a non-issue if the game was written by the same developers who are extending it; however, if the developer implementing adaptation is someone other than the original developer, then it is necessary to ensure that the software license for the chosen game allows for modification and redistribution. Second, the serious game needs to be of significant quality and robustness to support a planned redevelopment. The quality of the software can be assessed by reviewing available software artifacts including documentation, source code, and issue tracking data. Third, the playability of the game should also be considered, and existing playability studies are an asset.

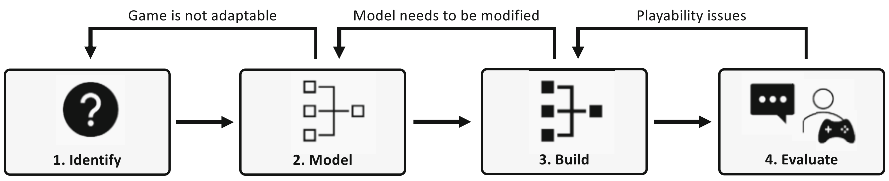


Fig. 6.3 An overview of our methodology for evolving adaptive serious games. Once a game has been identified for adaptation, a model and plan is developed that connects the game play tasks with a method for assessing learners. After the model has been created, the adaptation functionality is built into the existing code base. Finally, the new adaptive serious game should be evaluated to determine its efficacy (e.g., learning, engagement), and the evaluation results should be compared with the efficacy results of the original non-adaptive serious game

The main learning factors that need to be considered are:

- *Learning outcomes:* Adapting the learning content of a game requires a clear understanding of the required knowledge, topics, and learning outcomes that are present in the original game.
- *Learner experience and demographics.* Making informed decisions about adapting the chosen game requires detailed knowledge about the learners who will play the game. Learners of various age groups may respond differently to in-game adaptations. Furthermore, knowledge about the level of experience of the game's audience is needed in order to make good decisions about how to adjust learning content, including accommodation for an audience with no experience. Special consideration should also be given to adapting for learners of diverse educational backgrounds outside of computer science. Finally, we suggest choosing games that are inclusive in order to reach a diversified audience of learners.
- *Learning evaluation:* In order to properly evaluate the final adaptive serious game in phase four, it is best to choose an existing game that has already been evaluated with respect to learning. The existing evaluation can serve as a baseline in assessing the learning benefits of adaptation later.

6.3.2 Model

Once a game has been selected, the next step is to determine how viable it is to make it adaptive. This is based on the ability to create an accurate model of a game play task as well as the ability to model learners using the collected data.

Game play tasks can be deconstructed into several key features that are relevant for evaluating learning. In alignment with CbKST, each task should be associated with one or more competencies that are prerequisites; in other words, the task cannot or should not be completed unless the learner has demonstrated (and been given opportunity to do so) competency in those prerequisites. The successful completion of a task should provide evidence to both prerequisite and other associated competencies, so that learners can be evaluated based on the tasks they have completed. In addition to associated competencies, tasks also are targets for adaptation, and so the parameters of the task that can be changed should be taken into account. For instance, a task might have a time limit, feedback/hints, or other constraints that can be adjusted based on the perceived level of player competency. If a game is unable to adjust these task features (or has no tasks at all, such as a non-structured "open world" game), it may not be well suited for adaptation.

In addition to being able to create models of tasks, there must be sufficient available data sources to create an accurate model of the learner. The primary source for modeling learner competency comes from their completion of tasks. The number of successful attempts and failures on tasks associated with specific competencies generates a significant amount of data that can help classify the learner and predict their level of competency in different areas. In addition to successes and failures, the degree to which a learner is successful or unsuccessful provides more useful information than a binary feature. For example, the speed at which a learner completes tasks, or the quality of their solution, might help distinguish mid-competency learners from those with high competency. It is important, however, to separate data that reflects competency in learning versus competency in games—for example, it may be the case that faster completion of tasks only indicates that the learner is better at playing games, but does not have a higher level of competency than a slower learner that has higher-quality solutions. The degree to which data can inform a learner model has a significant impact on whether or not a game can accurately assess the learner and therefore make good decisions about how and when to adapt.

6.3.3 Build

This phase includes integrating key modelling functionality into the existing code base, for example, logging player behavior (if not already present), initializing the learner assessment model, and then applying an adaptation strategy.

Learner-specific adaptation requires a constant gathering and assessment of learner information. The data gathered is categorized based on what it is measuring and how often the measurements can take place For example, data may be gathered during a task, between a task, or between game play sessions.

There are different options that a developer might consider for initializing a player's assessment model. This is a challenging part of the build phase and will be discussed in detail in Sect. 6.4.3. The key consideration is to determine how to initialize the SG until enough data is collected to adapt to an individual player.

Once data is being logged and the SG initialization has been established, the developer can proceed to apply the previously chosen adaptive strategy. This involves increasing restrictions on steps, work, errors, and time for players who have demonstrated high competence and are seeking a greater challenge. Conversely, these restrictions should be reduced for players who exhibit low competence in order to accommodate their needs and reduce the level of challenge.

6.3.4 Evaluate

One of the challenges with serious game development is the need for accurate and reliable evaluation. One benefit to our approach of evolving existing serious programming games is that many have existing studies that can be replicated and reproduced for the adaptive versions, thus allowing us to evaluate the benefits of the adaptive modifications by comparing the study results from the original and adaptive versions of a game. In cases where the original version of a serious game did not have an evaluation, we recommend following best practices, which may include questionnaires, skill tests, interviews, and controlled experiments.

6.4 Challenges in Engineering Adaptive Serious Games

Engineering adaptive serious games presents a number of unique challenges that are not present in the development of non-adapative SGs. In this section, we will discuss five of the most common challenges faced by SG developers planning to add adaptation to existing SGs:

- **Selecting data for use in an ML model for SGs.** The selection of user data can have a huge impact on the success of ML as a method for adapting to a learner's needs in SGs. We will describe how to select the relevant features from potentially dense game play data based on assessing the data with respect to a learner's competence and understanding of a learner's mindset.
- **Choosing game elements to adapt.** Modern video games use player performance data to modify game play difficulty, and this can also be used in the context of SGs to modify game play elements that facilitate learning. For example, this might include changing the frequency or verbosity of in-game hints or modifying game play tasks based the abilities of the learner.
- **Solving the cold start problem.** Many ML methods require an initial data set for training, but at the beginning of game play, there is often little or no information available. We will discuss different strategies for addressing the cold start problem as well as how the generation of synthetic data sets can assist with making ML features viable sooner.
- **Determining how frequently to adapt.** It is possible to adapt an SG between game sessions, between game levels, or even during a game play task. One issue with determining frequency is that frequent adaptations can be computationally expensive, may frustrate the user, and can even negatively impact learning. Another issue is that infrequent adaptations, while mitigating performance issues, can be too late to handle frustrated players who fail to complete a task that is not suited for their level of competence. We will share our experience with selecting the adaptation frequency based on an analysis of the game play and the target learners of the game.
- **Evaluating that an adaptive game benefits learning.** The best practices in playtesting non-adaptive SGs are insufficient to provide confidence in the efficacy of ML-based SG features. Therefore, additional testing on top of traditional playtesting is needed. We will discuss our experiences with testing an ML-based SG including how to identify the relevant metrics and how to assess the results to determine if the ML adaptations are working correctly.

6.4.1 Challenge #1: Selecting Data

How do you decide what SG data to select for use in an ML model?

As with many machine learning applications, it is generally the case that the data available is not going to perfectly suit the needs of the ML model Fig. 6.4 for examples of available game data. For example, a player who fails a task that requires an understanding of a particular learning concept is not necessarily incompetent at that concept—human error must be considered. Thus, a probabilistic approach is best to account for cases where a player makes a careless mistake or when some tasks vary in their difficulty. It is likely the case that a player who can succeed at multiple challenging tasks is competent in the prerequisite competences, even if they occasionally fail simpler tasks based on the same competences.

Depending on the genre of game and the design of the task, some data features may not be relevant to the assessment of the player. For example, one should be wary of using time elapsed as a metric of competency if the task itself is untimed, and there is a possibility that the player may sit idle because they have opted to physically step away from the game for some time period. It can be difficult to distinguish between idle time and time spent thinking about a problem, but generally given the fast-paced nature of games (compared to traditional learning activities), one can expect that a lengthy delay is likely the case of the player being absent. This should be accounted for when automating any form of data collection from game play, as it will likely cause a significant effect on the data set if a 10-minute break from the game is measured as 10 minutes thinking about a problem. A player taking

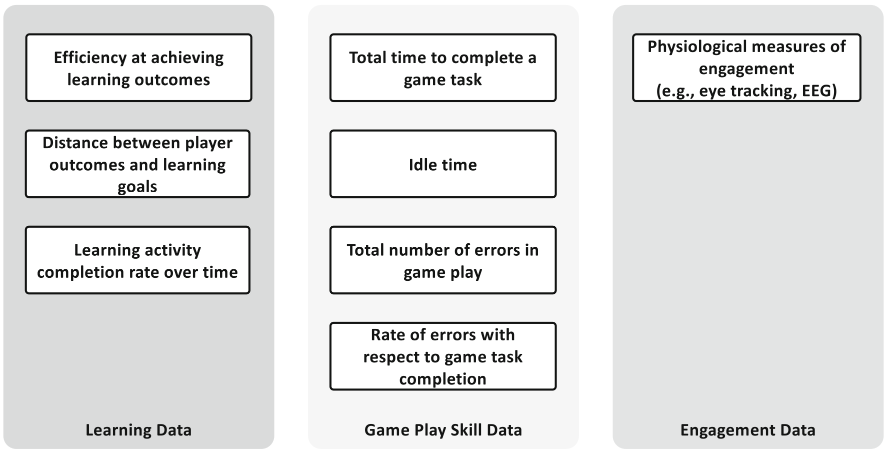

Efficiency at achieving learning outcomes
Distance between player outcomes and learning goals
Learning activity completion rate over time
Learning Data
Total time to complete a game task
Idle time
Total number of errors in game play
Rate of errors with respect to game task completion
Game Play Skill Data
Physiological measures of engagement (e.g., eye tracking, EEG)
Engagement Data

Fig. 6.4 Examples of serious game data available for use in adaptation

a long time to think about something, but still sitting and playing the game, is likely to consider interacting with some part of the environment, and games that present obstacles or hazards for the player to interact with while they play may mitigate their ability to idle.

Finally, the data provided for all the models must be used to make an assessment of the player. One approach to this is to use an unsupervised machine learning algorithm to compare the player to others who have previously completed the game. An unsupervised algorithm is a good choice because there is no way to say with certainty how competent any given player is based on their performance; we can only make estimates at best. An advantage to this approach is the ability to separately assess the existing data set and categorize each player based on their demonstrated competency. This way, the player can be assigned a label (such as low, medium, or high competency) and that label used to make determinations about the best form of adaptation. In the next section, we will discuss what can be adapted based on this information.

In GidgetML, the data collected from each task included the number of failed attempts at a solution as well as the number of “steps” needed to complete the task. Each player was then categorized into different levels of competency in debugging based on a K-means clustering machine learning algorithm. Using this algorithm, each of the three clusters was labeled as low, medium, or high based on the average performance of the players in that cluster, and each player in that cluster was assigned that label. Then, when the model had to account for a new player, it would repeat the K-means clustering algorithm and use the old labels to determine the player’s categorization based on the players with whom they shared a cluster.

6.4.2 Challenge #2: Game Elements

How do you select what game elements to adapt?

With an accurate model of tasks and players, a game can determine when it is necessary to adapt to the learner. However, there remains the question of what to adapt in a task—specifically, which game elements should be changed and how should they be adjusted.

It is first important to distinguish between game elements conducive to learning and game elements relevant only to game play and entertainment. For example, a change in the amount of hints and feedback provided to the player is likely to impact their learning, while increasing the number of obstacles that require manual dexterity to avoid may only serve to make the game more challenging if those obstacles are not related to any of the game’s competencies. It may be tempting to increase the difficulty of a game by adding more obstacles or using stricter restrictions on time. However, the audience for a learning game will include players who are skilled at games but lacking the desired competencies, as well as players who demonstrate high levels of competency but are less familiar with video games. There are different ways to challenge each player, and if the goal of adaptation is to create an experience that finds a balance between challenge and skill, then the way in which players are challenged will vary based on their skill level.

An important purpose of adaptation is to facilitate learners who are struggling with difficult content. This is why one of the most valuable forms of adaptation comes in the form of feedback. This can vary from hints about how to approach a task differently or even demonstrations about how to attempt to solve a problem that the learner may not be familiar with. The administration of feedback must also be carefully timed; presenting instructions at the start of a task might be ignored, but giving a hint to a player immediately after they make an unsuccessful attempt at an action is likely to immediately affect what the player will do next.

GidgetML modified only two game elements using its adaptive settings. The first modification was a change to the sample solution provided to players at the start of each task—players who demonstrated a high level of competency were given obscure code with many errors, while players at a low level were given code that might be mostly correct with only a few mistakes. The second modification changed the restrictions on what would be an acceptable solution to a given task. For high-competence players, only a very efficient or perfect solution would be accepted as correct, while low-competence players would be able to submit less efficient solutions to pass a level.

RoboBUG was developed to have two different types of adaptation. The first was a change to game play obstacles and code, using an approach similar to GidgetML for obscuring the code presented to players. RoboBUG also featured game obstacles that would hinder the player when navigating through the code, which were changed to be faster and more inconvenient for players at higher levels of competence. The second form of adaptation was the introduction of hints and feedback presented to players as they interacted with the code or failed to complete a level. Players at low levels of competence would receive frequent feedback and advice on how to understand the code in a task, while players at higher levels would receive such hints infrequently or not at all.

6.4.3 Challenge #3: Cold Start

How do you solve the cold start problem in adaptive SGs?

The cold start problem occurs when for new users "…the system does not have information about their preferences in order to make recommendations" [7].

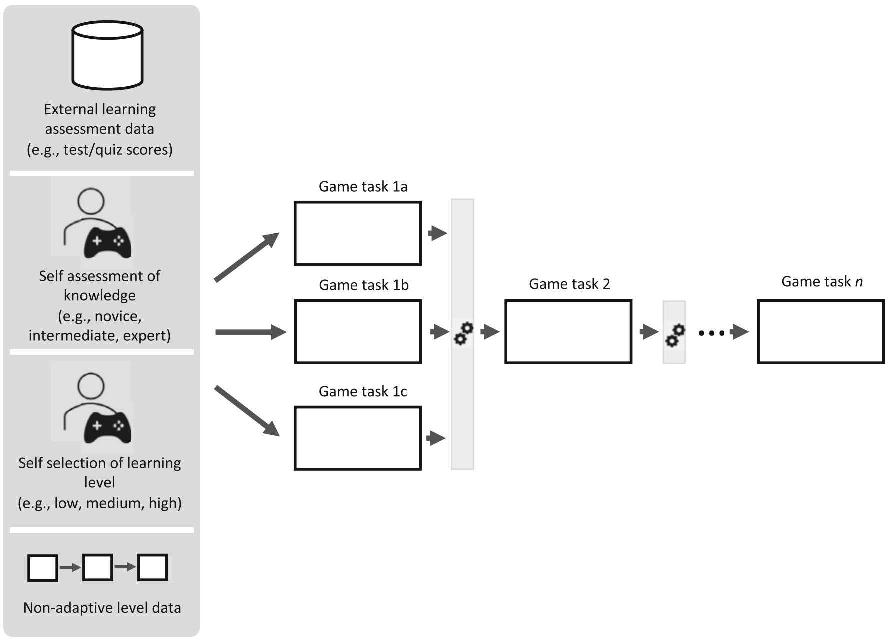

External learning assessment data (e.g., test/quiz scores)
Self assessment of knowledge (e.g., novice, intermediate, expert)
Self selection of learning level (e.g., low, medium, high)
Non-adaptive level data
Game task 1a
Game task 1b
Game task 1c
Game task 2
Game task *n*

Fig. 6.5 Alternative approaches to addressing the cold start problem in adaptive SGs

Specifically, for adaptive SGs, there is insufficient information about a player's learning needs to make a recommendation on the starting state of the SG.

It may be tempting to resolve the cold start problem by providing the game with existing external data about players, such as their grades or results on a pre-game quiz (see Fig. 6.5). However, this has a number of drawbacks. Firstly, there are issues of privacy in tracking the behavior of players while associating their data with grades—this could potentially be used to identify players who would otherwise wish to remain anonymous. There is also the question of whether or not the grades or quiz results are actually reflective of the player's competence. The cold start problem is not solved if the data provided is not accurate, and it could be the case that a player's grades or quiz scores are not reflective of the same competencies associated with the game's tasks.

One approach that can be considered is simply ignoring the cold start problem altogether. With this approach, there will be significant errors in the assessment of players for whom little game play data has been collected, but the model should become more accurate as the size of the data increases over time. A way to mitigate the issue of initial errors in assessment is to limit the degree to which adaptation happens in the earlier parts of the game. For example, the first tasks in a game might have little variation between them in order to account for the potential errors in assessment, while later tasks might have a large range of difficulty levels as there is a greater level of confidence in a player's actual level of competency.

An alternative solution is to use an earlier part of the game as the initial data set for the model, and only begin to adapt game play once the player has reached a certain point. The introductions to many games feature tutorials or guides to demonstrate for players how they interact with the game's environment. It may be the case that these tutorials and guides are suitable for data collection, but unsuitable for any kind of adaptation. In such a scenario, the way in which a player demonstrates the competencies targeted by the tutorial should be used as the initial data provided for machine learning. Thus, it is better for these tutorials to allow for a high degree of player agency, as opposed to tutorials which provide explicit step-by-step instruction on how to proceed.

GidgetML handled the cold start problem by only introducing adaptive game play halfway through the game's tasks. The first half of the game was an extended tutorial to introduce each game play feature one at a time, and it was only after these tasks were completed that GidgetML would begin to adapt to select new tasks. This meant that GidgetML had access to many levels worth of game play data that could be used to predict performance from each player.

6.4.4 Challenge #4: Adaptation Frequency

How frequently should you adapt in an SG?

Aside from the questions of how and what to adapt, there is also the question of when. Frequent adaptation can have the benefit of quickly addressing issues of frustration experienced during game play, which might otherwise lead to players abandoning the game from believing they lack the skill to play. However, more advanced machine learning algorithms may lead to issues of computational performance in the game, particularly as data sets grow and when the number of features is large.

Adaptation frequency can be separated into three categories: between session, between task, and during task (see Fig. 6.6). Between session adaptation occurs

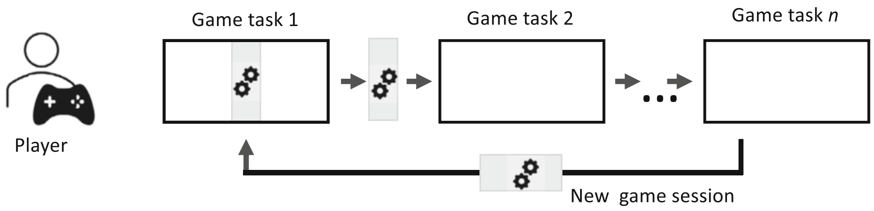


Fig. 6.6 Adaptation frequency—within tasks, between tasks, and between sessions

	Computational cost of adaptation	Time to adaption	Quantity of game data available
Within tasks	●●●	●	●
Between tasks	●●	●●	●●
Between sessions	●	●●●	●●●

Fig. 6.7 Trade-offs between different adaptation frequencies

when the algorithm only runs after the player has finished playing, presumably to pick up the game at a later time. Between task adaptation uses the data from each task completed and, in combination with the existing data set, adjusts the subsequent task based on the results of the game's algorithm. During task adaptation takes into account the player's current behavior and, based upon the task as well as the data sets, determines whether or not (and how) the task should be adjusted while the player is attempting to complete it.

Computationally, it is easiest to adapt between game play sessions, but for games that do not have significant replay value or are unlikely to be played for more than one session, this approach may not be useful. Between task adaptation can serve to adjust tasks based on previous behavior and allow the game to plan out the player's path to help adjust for any shortcomings in desired competencies. However, it does not provide a solution to players struggling with an immediate task, and cannot adjust to compensate unless the player finally completes the task or fails it. Although the approach of adapting during a task offers a solution to this problem, it has its own issues, namely, those of performance as well as the risk of overfitting data depending on the algorithm being used (Fig. 6.7).

GidgetML and RoboBUG made use of between-task adaptations in order to select from three different versions of subsequent tasks in the game. Since the elements of the game that were modified were exclusively restrictions on the task's acceptance criteria or increased vagueness of the sample code or sample solution, there would not be any elements available to modify during the task. In addition, since the games were designed to be completed in a single session, it would not be possible to make use of between session adaptation in an effective way.

6.4.5 *Challenge #5: Evaluating Learning*

How do you evaluate that an adaptive SG benefits learning?

Perhaps the most challenging of all issues facing developers of educational games is the ability to evaluate their efficacy, both for entertainment and engagement, as well as their value for learning. Although engagement and learning may be correlated, it is possible for a game to be significantly engaging while providing little learning value, or vice versa.

Historically, educational games are not frequently given thorough evaluations for their value in learning (see Figs. 6.8 and 6.9) [11]. Many games developed by researchers and gaming companies are only tested for their functionality, and not tested to see if the target audience will gain any significant improvement in their development of any competencies. Part of the reason for this is that it is difficult to

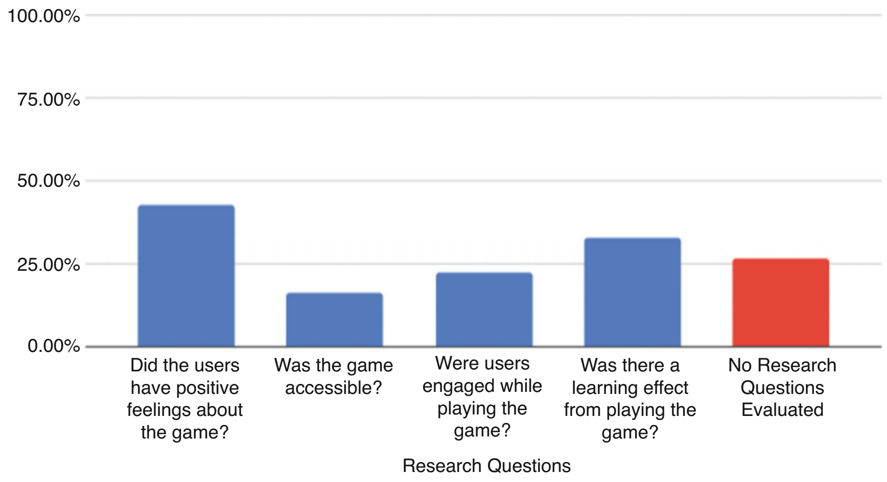


Fig. 6.8 Questions evaluated by studies of serious programming games [11]

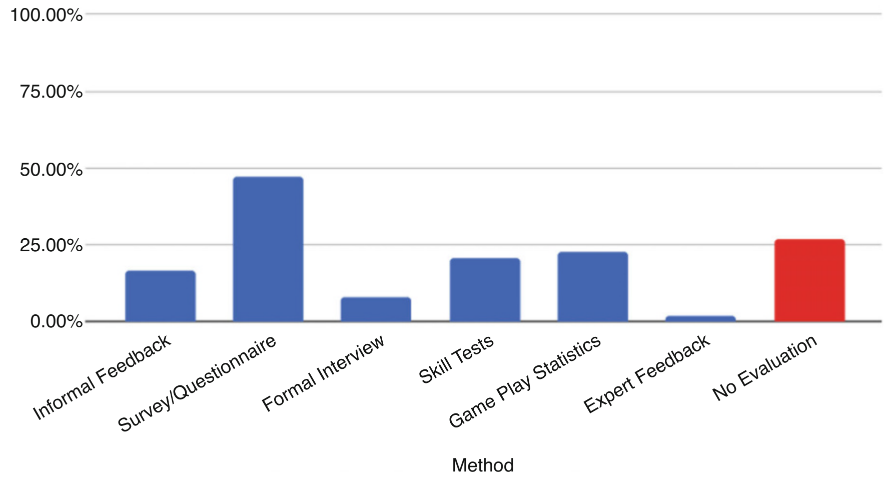


Fig. 6.9 Methods used in the evaluation of serious programming games [11]

ascertain whether or not a game is an effective learning tool. Adaptive and non-adaptive games are equally capable at logging game play data, but there is no established set of best practices for evaluating the learning effect of educational games.

One approach to testing is to consider the use of a post-game evaluation, such as a questionnaire, to determine how much the player understands after playing the game. This was the approach used for RoboBUG—players would complete a pre-game questionnaire that tested their knowledge of the learning competencies, then play the game, and then complete the same test again to see if they would change their answers. However, this has several issues—firstly, players may simply not wish to complete the assessment if there is no incentive to do so. Secondly, a pre-game questionnaire is necessary in order to compare the results of the questionnaires, and such tests are an inconvenience that may encourage players to not bother playing the game in the first place. Finally, and most importantly, it is completely possible that the questionnaires do not actually assess the competencies that the game will teach. A multiple-choice test, or other test that can be automatically evaluated, is ill-suited to determine whether or not a player has achieved the highest levels of Bloom's taxonomy [2]. Although it may be possible to make a test suited to determining whether or not the player can recall, understand, or apply different competencies, it is not as easy to determine whether or not a learner can analyze, evaluate, or create when limited to an automated grading system. It is also challenging to overcome this obstacle by creating a generalizable questionnaire, because any two games are unlikely to cover the exact same content.

6.5 Discussion

The development of new adaptive serious games and the addition of adaptation to existing serious games are both challenging engineering problems that require a combination of expertise in the learning domain, game development, software development, and machine learning. In this chapter, we have presented one approach to engineering adaptive serious games using ML. We have also discussed five of the common challenges faced in the development of adaptive SGs and provided insight and guidance based on our own experience of development adaptive SGs in the field of computer science.

The main limitations of the models and practices presented are as follows:

- All of the models and practices are based on our experience with the development of adaptive SGs for computer science. While we believe these generalize to other domains, we acknowledge that the generalizability has not been fully researched.
- Our methodology for evolving existing SGs to us ML has not been independently utilized by third-party developers. We have successfully applied this methodology to add adaption to a third-party developed SG (Gidget) and to an

in-house developed SG (RoboBUG), but we do not have data on the use of this methodology outside of our research group.

Our experience with the development and deployment of adaptive SGs in computer science has shown that despite the challenges that may be encountered, the use of adaptation in SGs provides an opportunity to enhance the engagement and learning of SG players by personalizing the game play to their specific learning needs.

References

1. Caponetto, I., Earp, J., Ott, M.: Gamification and education: a literature review. In: European Conference on Games Based Learning, vol. 1, p. 50. Academic Conferences International Limited, Reading (2014)
2. Forehand, M.: Bloom's taxonomy. Emer. Perspect. Learn. Teach. Technol. **41**(4), 47–56 (2010)
3. Frutos-Pascual, M., Zapirain, B.G.: Review of the use of ai techniques in serious games: Decision making and machine learning. IEEE Trans. Comput. Intell. AI Games **9**(2), 133–152 (2015)
4. Kopeinik, S., Nussbaumer, A., Bedek, M., Albert, D.: Using CbKST for learning path recommendation in game-based learning. In: 20th International Conference on Computers in Education, pp. 26–30 (2012)
5. Lee, M.J., Ko, A.J.: Personifying programming tool feedback improves novice programmers' learning. In: Proceedings of the Seventh International Workshop on Computing Education Research, pp. 109–116. ACM, New York (2011)
6. Lee, M.J., Ko, A.J.: A demonstration of gidget, a debugging game for computing education. In: 2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 211–212. IEEE, Piscataway (2014)
7. Lika, B., Kolomvatsos, K., Hadjiefthymiades, S.: Facing the cold start problem in recommender systems. Expert Syst. Appl. **41**(4, Part 2), 2065–2073 (2014)
8. Miljanovic, M.A.: Adaptive Serious Games for Computer Science Education. PhD thesis, Ontario Tech University, Oshawa, ON (2020). Supervisor: Jeremy S. Bradbury
9. Miljanovic, M.A., Bradbury, J.S.: Robobug: a serious game for learning debugging techniques. In: Proceedings of the 13th Annual ACM International Computing Education Research Conference (ICER 2017), pp. 93–100 (2017)
10. Miljanovic, M.A., Bradbury, J.S.: Making serious programming games adaptive. In: Proceedings of the 4th Joint Conference on Serious Games (JCSG 2018) (2018)
11. Miljanovic, M.A., Bradbury, J.S.: A review of serious games for programming. In: Proceedings of the 4th Joint Conference on Serious Games (JCSG 2018) (2018)
12. Miljanovic, M.A., Bradbury, J.S.: GidgetML: an adaptive serious game for enhancing first year programming labs. In: Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020) (2020)
13. Shute, V.J.: Stealth assessment in computer-based games to support learning. Comput. Games Instruct. **55**(2), 503–524 (2011)
14. Streicher, A., Smeddinck, J.D.: Personalized and adaptive serious games. In: Entertainment Computing and Serious Games, pp. 332–377. Springer, Berlin (2016)
15. Susi, T., Johannesson, M., Backlund, P.: Serious games: an overview (2007)

Part II
Topics on Experiences with Gameful Systems

Chapter 7
Future Directions in Games for Serious Contexts: A Conversation About Transferability

Vanissa Wanick, James Stallwood, and Guilherme Xavier

Abstract This chapter provides a conversation in the form of an opinion piece about strategies commonly utilized in games that can be “transferred” to Serious Games (SGs) and games for serious contexts. The aim of this chapter is to provide different perspectives and examples that are currently utilized by entertainment games that could be utilized in SG development. SGs are often developed for particular situations, and with that, the development process might be attached to specific stakeholders, becoming, most of the time, a “one-off” product, which may limit the SG life cycle and game repurposing. This chapter brings with three complementary perspectives to address future challenges and opportunities regarding emerging aspects of player agency and SG modification and transferability across different contexts. First, we discuss emergent possibilities, bringing examples from digital entertainment transferability. Second, we take into consideration “modding” strategies to provide insights for SG modification and transferability, discussing the role of the “context” in games development. Third, we demonstrate the importance of AI emotion modelling to inform better game design. To conclude, we respond to these ideas and provide suggestions for SG research and practice.

Keywords Personality vectors · Transferability · Serious games · Position paper · Modding · Emotional modelling

7.1 Introduction

Serious games (SGs) and gamified applications utilized in non-entertainment contexts have the potential to promote positive behavior but also keep the user

V. Wanick (✉) · J. Stallwood
University of Southampton, Southampton, UK
e-mail: vwv1n12@soton.ac.uk; J.E.Stallwood@soton.ac.uk

G. Xavier
Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil
e-mail: guix@puc-rio.br

engaged in a specific activity. Due to their attachment to the "context," these games and applications may become a "one-off" product, being difficult to replicate outside their own environment. But what if the process of creating and developing serious games could be fed by other models, imported both from the entertainment area and from constituent approaches of automatism programming techniques, mediated by the experience of their designers?

This chapter provides a conversation and an outline of topics usually utilized in games that can be "transferred" to serious games and games for serious contexts. This approach we have called as "transferability." The concept of transferability discussed in this chapter can be twofold. From one side, transferability is about the generalization of "solutions" and from another, transferability accounts for the extent to which a solution can be effectively achieved in another context. Thus, considering that serious games are designed based on and for a particular context, we believe that by discussing transferability, we can provide recommendations for improvement and insights to design and develop more effective SGs.

In research, transferability relates to the degree to which that research method or output can be transferred to other contexts. Transferability might be at times combined with generalization, which means that the output can be applied into other contexts. For SG design, it means that these games are designed with a purpose in mind; however, would that mean that a serious game can be only applied to one single context? As [1] mention, the definition of the term SG is related to cultural practices, but as these practices evolve, the range of the term might also change. A SG can be used to communicate a message, enable training/cognitive or physical capabilities, and facilitate data sharing and collection (e.g., crowdsourcing and citizen science applications, such as *Foldit*) [1]. This means that SGs have a clear purpose, though it does not imply—not directly—that the game itself needs to be designed for that single purpose.

Yet, the term "serious games" deals with a conflict between what is admitted as purposeful seriousness and what is expected from a game as an activity aimed at the enjoyment of its participants. If the game exists at play time, this is where, at the "Game," we will drop our anchors to take a look around. In addition to this discussion, therefore, we will first consider the "game" part of the term "serious game," implying in the dialogue that the difficulty for designers in making their projects engaging is not in the content but in the way in which it is presented for interactive participation.

Considering this, this chapter discusses transferability, from the perspective of the authors, in three sections: (1) transferability of competences and skills constituted by gaming practices (e.g., the idea of games as instruments) and transferability via games as tools (e.g., *modding* practices), (2) transferability of "informal" game development practices via Game Jams and games developed as a commentary about a particular "serious" context (e.g., health issues), and (3) personality vectors as an AI mechanism to improve the quality of in-game agents. Each section has a particular treatment given by the authors, with relevant examples. We believe these examples extend debates in the literature around repurposing of SGs [2, 3] and agent-oriented software engineering [4].

Despite the challenges of developing games for serious contexts (e.g., user engagement, costs of production, metrics of the effectiveness of the game/application, etc.), software engineers and designers have been working toward personalized strategies and algorithms that provide users with unique experiences [5]. Artificial intelligence (AI) might be able to solve a few issues with personalization; however, gameplay adaptability might be difficult to achieve when the game is already made.

Games applied in serious contexts might also depend on stakeholders' ability to deploy the game in that context (e.g., by recommending and reading results of a health application or designing experiences for (and with) students). Thus, when designing games for serious contexts and gamified applications, it is important to understand how much control would be given to its "users," how and when, in order to make it relevant.

This chapter aims to address future challenges and opportunities regarding emerging aspects of player agency and SG modification across different contexts through a set of conversations posed by the authors. To conclude, we respond to these ideas and provide suggestions for SG development. For software engineers, we discuss the potential of modular approaches, together with the application of personality vectors to enrich intelligent tutoring mechanisms and potential practices for the early stages of SG development.

7.2 Serious Games Design Transferability: Setting Up the Conversation About Purpose, Tools, and Instruments

In this section, we discuss aspects that blur the line between entertainment and serious contexts, such as games designed as a commentary about a "serious" context, but that are not SGs per se as their main purpose is entertainment. We will use two examples to discuss the concept of transferability in the design of two games: *Before I forget* and *Hellblade: Senua's Sacrifice*.

Before I forget (Fig. 7.1) tells the story of a woman trying to reconnect with her past, but it creates several scenarios that reflect the emotional portrait of dementia. In an article published by *The Guardian* [6], the authors mentioned the idea emerged from a Game Jam that had in fact a serious context to tackle. Game Jams (GJs), from Global Game Jam (GGJ) to many others (particularly hosted in the platform *itch.io*), can support a new landscape for game design and development, since it provides a safe space for people to collaborate and create together in an informal setting. The reason for that is that these platforms and events can offer innovative and interesting new mechanics since it allows quick and rapid prototyping games. GJs are usually thematic and have an initial point of interest. Themes may vary, from indigenous communities to keywords like "roots" (GGJ 2023s theme). Within these themes, questions might emerge; for example, what is the best way to represent the concept of "roots?" What do "roots" represent? How might these ideas become interesting mechanics? These are common questions that emerge in a brainstorming session,

Fig. 7.1 *Before I forget* (Source: Threefold games)

which may emerge from a problem statement [7]. But what if the questions become more technical and more subject specific?

Let's bring back *Before I forget* and discuss the questions suggested by the developers and designers. During the early stages of development, the creators mentioned that one of the questions they had was "What happens when we lose our memories?" [6] Then the creators moved to design questions, such as on how to implement and how to go further into developing the narrative and the other design components. The key aspect for the innovative factor in this case was the transferability from one discipline to the other and having two questions being asked at the same time: one related to research and scientific aspects and the other related to design and development (on how to make the final prototype). *Before I forget* is not a SG but has serious contexts involved in its design and core experience.

The same can be said about *Hellblade: Senua's Sacrifice*, which tackles psychosis. For the design of this particular game, designers worked together with professors and neuroscientists, and the research was fed into game development via the description of hallucinations and other experiences [8]. The game itself portrays a *Viking* environment, but it is the experience that conveys the psychotic state of the main character (*Senua*). *Senua* is a traumatized Celtic warrior on a quest to the *Viking* underworld, Hel. Thus, there is a layer of "fantasy" being added to a "serious" context. It is worth mentioning that the game, however, has not emerged from a GJ, as compared to *Before I forget*, but from a research project.

Both *Before I forget* and *Hellblade: Senua's Sacrifice* had their base on scientific and subject specific knowledge but are entertainment games. However, this does not mean that these games cannot be used in an educational setting. In fact, they might be great tools to teach about particular health conditions.

In this sense, it is important to differ two concepts that are fundamental when considering games, and serious games in particular, with a focus on their purpose and transferability: games as "tools" and games as "instruments." Although they seem like only terminological differences, they evoke different interpretations and thus different perceptions of purpose.

When we consider games as "tools," we are assuming their functional characteristics as something participated to resolve a problem. It should be noted that a game as "problem-solving activity, approached with a playful attitude" [7] is a sufficient ontological definition of game, in the absence of a definitive one, that both the area and the related disciplines have not yet been able to consolidate. That said, when we think of games as "tools," we are assuming that they fulfil a dual purpose: the first within a diegetic dimension and in compliance with rules to reach an objective and second in a sphere that expands in the experience of its own practice.

For example, there is a recent appeal in translating work activities that would otherwise be considered completely devoid of fun, bringing to make-believe the "responsibilities" of a simulated job-like activity. We highlight here the games *Plane Mechanic Simulator* (Disaster Studio/Cobble Games, 2019), *Euro Truck Simulator 2* (SCS Software, 2012), and *PC Building Simulator* (The Irregular Corporation, 2018), which somewhat could be perceived as serious games in convenient contexts (if the player is a mechanic, trucker, or computer technician), beyond those they are "originally" related to.

In the first game, *Plane Mechanic Simulator*, the player is invited to consider that World War II airplanes are like an assembly of puzzle pieces that occupy a specific location at a specific time, which we can summarize as a *space-time-problem*, in the mechanical structure of a complex vehicle. For there to be a repair (pointed out bureaucratically, on a clipboard), the player must navigate a three-dimensional structure of the object to be mended, laboriously removing screws, plates, and exquisitely detailed parts to meet the objective of the demand, later carrying out the reverse process of disassembly to obtain victory. What at first glance may seem like an activity worthy of Sisyphus, it actually finds an echo in a mechanistic and structuralist society, the one that seeks to reduce the distance between *modus faciendi* and *modus operandi* of cultural assets in the virtual world. The industrial complexity has as a summary a kind of *gap* between what is made and what is used, which in part explains the large number of different "maker kits" now on online sales to satisfy hands that are not busy with manual duties but overloaded with intellectual work, as a way of aesthetic "compensation." Thus, if there is a low possibility of actual building nowadays in our lives, at least the experience of *virtual* "building" is possible, seductive, and free from consequences.

In the second game, *Euro Truck Simulator 2*, the player takes the wheel of a truck to transport goods, of all kinds, along the roads of Europe. There is a version located in the United States by the company itself, but this *spin-off* title little or nothing differs in the result of a driving experience, since the community also manages to create other *mods* to update the game to other parts of the world and their respective highways. If the player has no interest in trucks, but in buses or even cars, it is too possible for unofficial installations to adapt the game to the taste of its driver, which is seen as normal behavior by the developer company. This, in part, solves technical issues (in the "eternal pursuit of realism as an end") and in part interferes with the way the game is consumed (beyond the usual player and in more *spectacular* ways). On the *Twitch* video platform, specialized in broadcasting matches from this and many other games, there is vast content of "drivers" getting involved in hypothetical scenarios of dangerousness on unrealistic roads, in a kind of "self-assured skill tests," which in the game is a concern but not the most important. At least, no more than following the traffic rules (whose fines are immediately charged to the driver), exploring kilometers and kilometers of territories based on typical old continent environments without running out of fuel or passing out from sleep deprivation, and delivering packages on time for XP acquisition.

In the third game, *PC Building Simulator*, the player takes over a private shop specializing in repairs and installation of computer parts, retracing a path similar to *Plane Mechanic Simulator* in a genre that we could call *adminpuzzle:* the traditional puzzle of digital games plus financial responsibilities such as those of *Euro Truck Simulator 2* and their administrative schedules, garages, and employees. The metalanguage of this game is too provocative and does not allow us to exempt a comment about a computer game in which computers are based on "real" computers, and even their parts are sponsored by "real" companies, dedicated to computer construction that must be built with performance for digital games. The process is cyclical and, therefore, meditative: tasks need to be carried out on different equipment, seeking with audio-visual assets to "ludify" something that would otherwise be perceived as just a technical work devoid of "epicity," common in other interactive digital works.

As well as these three games, many others seek, in the rigor (or lack of) of the simulations, to bring their players the feeling of a metrical tooling operation. There are learning curves and, therefore, results that demand dedication for a certain number of hours. The game is thus a "tool" to be used to satisfy an estimated result, generally far from the reality of its player. As games are opportunities for other experiences, there is meaning in challenges that deal with this alternative and, therefore, experimental prerogative.

Minecraft, for instance, can be said to be a "simulator" that allows players to experiment with. The "game" has playful and gameful components, allowing players to explore an open world but to also play with rules (surviving mode). *Minecraft* has many versions including an educational version, which then becomes a "serious toy" (using [1]'s terminology). This "version" and its *mods* are accompanied by teaching resources. In this case, the "real" context (e.g., classroom) is still imperative for the learner to achieve the learning objectives. What makes *Minecraft* an interesting example about transferability is that the game can be applied into other contexts, which comes together with the idea about repurposing a game for a particular context. Can a SG designed for that purpose be repurposed and have "transferable" components? So, could we do the reverse? *Foldit* is potentially a good example (see Fig. 7.2). When *Foldit* was launched in 2008, its first challenge was to decode proteins, which then were applied to solve issues related to the COVID-19 global pandemic (see Fig. 7.2). *Foldit* did not change its core and still remained

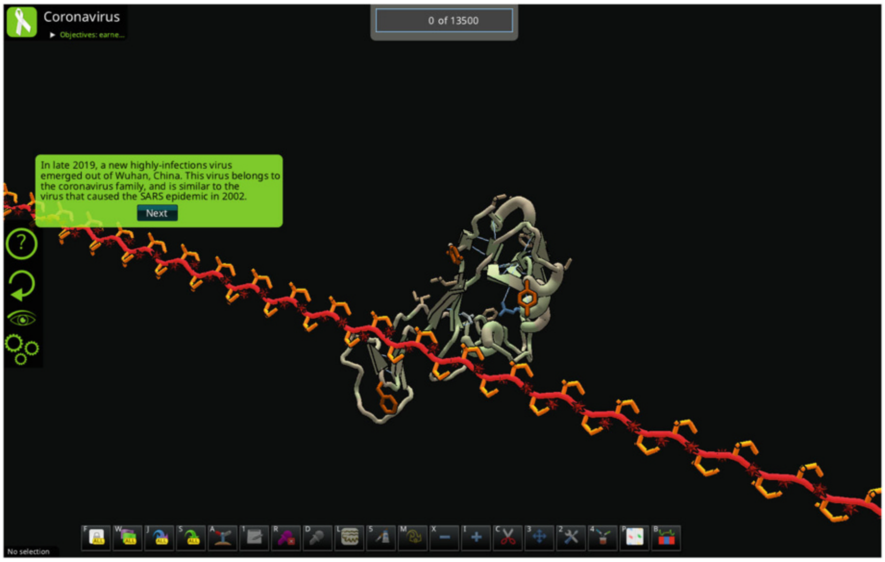

Coronavirus
Objectives: earne...
0 of 13500
In late 2019, a new highly-infections virus emerged out of Wuhan, China. This virus belongs to the coronavirus family, and is similar to the virus that caused the SARS epidemic in 2002.
Next
No selection

Fig. 7.2 *Foldit*: COVID-19 version (Source: Screenshot from authors based on http://www.fold.it)

with the same mechanics, but it changed its challenge and potentially the way these were presented to players. Therefore, would "transferability" mean that the core mechanics stay intact? Would that not be the same as reskinning an existing game?

Jenkins et al. [9] proposed that educational game design requires negotiation of identities, assumptions and meaning, as games it might not be clear if the games are seen as just entertainment or just educational pieces. This might be a matter of perspective and purpose, being the goal of the game to support learning objectives or particular goals, and the "fun" is the way to achieve that. Nearly 20 years later, there is still a need to negotiate "identities," which [9] represent by teachers, designers, students, and so on. Nearly 20 years later, these roles and identities are revisited with the rise of the *modding* culture, like those common in platform/gamehub *Roblox* (Roblox Corporation, 2006). Modifications have been used in design and in games for serious contexts, as mentioned before. However, these were still modifications mostly made by teachers/designers and less from the perspective of players/students. In fact, one of the most challenging aspects is to be able to identify which games are "moddable" [10].

We talk now about games as "instruments," that is, as an activity that demands time invested in skill, constituted by dedication and some *tuning*: the operational adjustments that are necessary so that the result is in line with what is expected for a driven sensation to fulfil rule-imposed objectives. In these games, victory is not enough but the certainty of "being one with the game" in the fullness of knowing how to know. Thus, the more used the instrument, the better the apparent and internalized result, which we can understand as a learning process.

Csikszentmihalyi's [11] *Flow* is immediately expected to appear right here: by becoming one with the game, the player awaits the trance that allows him to advance and advance, seeking in virtuosity, both personal and social satisfaction.

The transferability of competences and skills constituted by gaming practices has been the core of serious games since their first applications for school renewal and training in the 1970s of one or more proficiencies; it becomes common with the expansion of digital electronic accessibility in the 1990s and with the multi-mediatization of cybernetic microcomputing in the 2000s. Although they belong to another dimension of interactivity with a purpose (which would go against the usual expectation of a game for its "promising empty" condition precisely because it belongs to the sphere of interstitial activities of modern and productive societies). While "instruments", games require their participants to understand all the resources involved in order to take better advantage of them. In addition to their mechanics, dynamics, and aesthetics [12], games as an activity exist only ongoing, but as an experience, exist both ongoing and after, in the form of affective memory and tacit knowledge.

Whether "tools" or "instruments," games can be "recommissioned" for other practical purposes as long as they are not deprived of their main conditional characteristics: fantasy and control. This means that entertainment games can be appropriated, via an exercise of "metaphorization"; that is, the game has a symbolic meaning that could function as a metaphor or analogy. After all, as a means of communication and expression, games assume themselves as promoters of a purpose. And being recognized as such, they facilitate the migration of use knowledge from one context to another. As a subversive sample of this kind of entertainment gaming transferability we are talking about, let's discuss the *modus operandi*, the use of the keyboard for first-person shooter games.

Back in the year 1997, *Quake* became an outstanding commercial success by iD Software. Until the emergence of complex games that require the simultaneous use of mouse and keyboard, video games had a certain amount of possible and allowed inputs, establishing two paradigms sufficient for interaction, the "lever" (generally pushed in the direction of the command) and the "button pressed" (to activate or deactivate an action). The use of the keyboard had in the function keys F1 to F12, Enter, Space Bar, and directional arrows an industrial agreement of use; therefore, they were foreseen in function of the programs that made use of their positions and functions. When that year Dennis "Thresh" Fong beat Tom "Entropy" Kimzey in the first national *Quake* tournament, he went on to popularize an inadvertent use of W, A, S, and D keys as the new "directional" keys. Anyway, the next first-person shooter games like *Half-Life*, *Counter-Strike*, and *Unreal Tournament*, to name a few popular examples, gave up the directional arrows due to this new "imported" adaptation scheme, which is still considered *default* even for games that are not first-person shooters.

Speaking of the present, in search of a future in which SG invites the versatility of its samples to acquire welcome *flow* states, it is necessary to investigate opportunities in the production of game assets and its relation with its players. Transferability, then, becomes a token both serious and entertainment games must rely on, focusing on engagement behavior, directly associated with emotional responses. Next, we bring some technical suggestions about that.

7.3 Going Technical: Personality Vectors as a Strategy Toward Emotional AI Modelling

The aim in this section is to first set a premise for a concept of personality vectors and then show with an extended example how these personality vectors might be used in a more complex example. This section deliberately discusses examples outside the SG field, particularly looking at a potential strategy to evoke emotional behaviors of non-player characters (NPCs). The application of these examples is addressed in the conclusion.

To begin, let us think of a guard patrolling an environment. This guard will patrol until an enemy, the player, is spotted and will then engage the enemy in combat or flee from the enemy dependent on if the guard is sufficiently healthy and armed to do so. We will assume that the guard begins their duty fully in good health and well-armed for the purpose.

Regardless of whether we might use a state-machine or a behavior tree or any other kind of model for this NPC, the design is similar for all of them. The guard receives information from the game-world and its own current situation and will decide upon certain actions if pre-conditional statements are met. For example, suppose that it is decided that in order to engage an enemy, the guard must first be able to see the enemy, be reasonably healthy (e.g., health > 50%), and have an abundance of ammunition for a number of shots to be made (e.g., 10 shots). If those conditions are met, then the reasoning framework for our guard will make them engage their foe. This collection of preconditional statements and actions can be called the NPC's "strategy" or "strategy framework." One could think of it as the training manual that the guard received before they took the job.

This strategy framework may be a sufficient basis for all guards, but the way it works opens an obvious in-game question: how does the guard know how healthy they are? So, we might abstract this further and ask what does this guard have the right to know accurately about themselves from moment to moment? As the preconditional statements in the strategy framework rely on the data collected by the guard, why should that data be processed directly rather than being put through some data judgement protocol first?

If the guard is wrong about spotting the player, then the guard may open fire into a harmless jacket hanging on a door. If the guard is wrong about being sufficiently healthy to take on the threat, they will find themselves at quite a disadvantage. If the guard is wrong about having enough ammunition, they will quickly find themselves in an action movie cliché.

For simplicity's sake, we could imagine the data judgement protocol to look something like this and using a simple probability:

```
NPC_Judgment():
r = random_number(0, 1)
IF r <= 0.05
   THEN RETURN random_number(-10, 10)
ELSE RETURN 0
```

For the guard's preconditional variables (health, ammunition, enemy_spotted), there would be two distinct variables used: the actual, or true, value variables and the perceived variables where the returned value of the NPC_Judgment protocol is added to the actual value. The result is that instead of using the actual values in the strategic framework's conditional reasoning, the perceived values are used instead.

This would provide greater nuance and difference to the guard's behavior at relatively little cost and, more importantly, without having to alter the basic strategy framework at all. However, we could go further. Though we have chosen arbitrary numbers for the probability and the ranges to add for the perceived variables, these could instead be linked to a predesigned personality for the guard or change over time according to circumstances. We can rewrite parts of our NPC_Judgment protocol to account for these options.

```
misjudge_rate = 0.05
misjudge_extreme = 10
NPC_Judgment():
r = random_number(0, 1)
IF r <= misjudge_rate
 THEN RETURN random_number(-misjudge_extreme, misjudge_extreme)
```

In predesigning the personality for the guard, we might decide to make a guard that is more prone to making misjudgments, and so we would increase the misjudge_rate to be a higher value and more likely to occur. Or we might decide that our guard character is even more inaccurate with their perceptions than an ordinary guard and increase the misjudge_extreme value. Or, indeed, we could decrease those values for more seasoned guards and any mixture in between. In taking this approach, we allow ourselves to make personality and experience changes to our guard agents without changing the strategy framework for all guards. Divorcing this process enables us to safely experiment with these parameters without the need for tedious minute work in the strategy framework itself.

We might decide instead to fix these values to some external *stimuli* as well. For example, consider the type of guard who on seeing half of their comrades. If instead we took something like the number of times the guard has seen a fallen comrade and either mapped that or used it as a factor for the guard's misjudge_rate and misjudge_extreme values, then we begin to get something approaching a basic fear index. When people are scared, they make mistakes. Our NPC_Judgment variables might instead look a little like this:

```
misjudge_rate = 0.05 * fear_value
misjudge_extreme = map(10, 30, fear_min, fear_max, fear_value)
```

where in the first instance the fear index acts as a factor for the misjudgment probability and in the second instance the misjudgment extreme is mapped between 10 and 30 depending on the current value of the fear index compared with some maximum.

If we were to have multiple personality indices, we might weight their effects on these judgment variables differently according to how we thought they might apply or by personality design. Suppose our guard has two personality indices: fear and anger. We could design a guard that is affected more by one than another or by both equally. So, a guard who is more cowardly might be made like this, misjudge_rate = (0.1 * fear_value) + (0.02 * anger_value), whereas a guard that is more easily caught in the red mists of rage might be made like this: misjudge_rate = (0.01 * fear_value) + (0.2 * anger_value).

In either case, we have a liberty to model our guards according to personality in whichever way we choose. A fearless guard might take no influence from fear at all.

In the event we have multiple personality indices to track and to factor into our NPC_Judgement protocol, we could create a personality matrix where one row, say the first, is the effect vector for the guard's personality and the second row are the personality indices generated from external events in the game. Treating the rows as single vectors, our simple model for the misjudge_rate variable can be calculated with the dot product of these two rows.

Of course, we don't have to use the dot product approach or the mapping approach described above; these are simple implementations. However, going forward with this idea, it is worth remembering our two key ideas: a strategy framework which does not alter at all between different agents and a personality matrix or vector of indices which is used to add a variation to the behaviors described in the strategy framework.

Let us now move on to a different kind of problem: NPC poker players or poker agents. Poker, and in this case referred specifically to Texas Hold 'Em, as with many other card games, has standard conventions that accompany its rules, and because lying is an intrinsic part of playing poker, those conventions make the game both stable and unstable in trying to determine the actions of a player.

For example, it is a common convention that the closer a player is to the dealer chip from a clockwise position at the table, the stronger their hand should be if they choose to call or raise a bid. The reasoning for this strategy is simple; players closer to the dealer chip must act before those further away, and it is better to be in a position of strength. Similarly, if a player has the dealer chip or is close to the dealer chip on the right, they can play weaker hands knowing that they do not have to act before anyone else.

If we were to design a poker agent to use this strategy alone in a limited experiment of opening plays, we would be missing a fundamental part of the game. Our agent would consider its position, analyze the strength of its hand, and make some judgment, probably probabilistic on whether it should call, raise, or fold. However, *what we are missing in this implementation is the ability to lie*. Because of the convention that is held due to the collective experience and knowledge of poker players down through the ages, a big blind who raises represents a strong or monster hand. At this point, we should carefully not the poker terminology “represents.” We do not say the big blind player has a strong hand, only that they represent one and act as if they do.

The curiousness of poker is that it is a game seeped in personality. As a player, we must assume our competitors are trying to be truthful about their hand and intentions even though there is an excellent chance that they are lying to us.

Therefore, when we design such agents, we must be mindful of many more factors. We could, for example, begin with a simple protocol for analyzing the strength of a hand. There is, for example, an approximately 6% ($\frac{\binom{13}{1}\binom{4}{2}}{\binom{52}{2}}$) chance of being dealt a pocket pair. In the pre-flop portion of the game, this is a major strength. Afterall, there are only 13 values of pairs, and their relative unlikelihood means that at a table of five people, if you have a pocket pair at this stage of the game and cards were turned, you’d have a very good chance of winning. By the end of the game, five cards, the likelihood of other players also having a pair increases significantly as do the chances of your pair being beaten.

What is useful for designers is that poker is a game which illustrates the folly of holding onto a strategy framework only model. Our goal is not in creating poker agents that win optimally wherever they can but ones that *act like poker players to increase the verisimilitude of the game experience.*

This then leads us to the question, what of the personality vectors/matrices? How do we use them? We must first ask ourselves what is a reasonable area in which a poker player could make a mistake in their reasoning? What is the equivalent of misjudging one’s ammunition or health status in poker? Finally, because of the nature of the game itself, how can we abstract misjudgment protocols to include intentional misrepresentation of our hand, bluffing?

Let’s start with some assumptions about our poker agent. Firstly, we will assume that our agent will not forget the cards they have. While this does occasionally happen, the agent is technically free to look at their cards at any time so we will not consider this to be a valid misjudgment opportunity. Secondly, we will assume that our agent knows the rules of the game fully and isn’t playing different variants unknowingly. Thirdly, we will assume our agent can perform simple arithmetic operations like percentages of the pot, cards seen versus cards remaining, etc.

As there are many factors that could be in play in a poker game, we will only use one of them: hand potential. Using only the values for the two cards in the agent’s hand and knowing if they are suited (sharing the same suit) or not (“off suit” or “off”), there are 169 possible hands an agent might have. We can value our hands numerically from the best possible two-card combination (two aces) to the worst possible two-card combination (7 – 2 off). As with our earlier guard example, it should be possible to assign a misjudgment for card strength in the same way we discussed the idea of health or ammunition earlier.

Card strength is not only relative to other cards but also indicates an idea of playable freedom depending on where the agent is sat relative to the dealer chip. In other words, a hand’s strength is not dependent only on the values of the cards but also on where the player is sat in combination. If there were six people at a table playing the game, then the best position would generally be in the sixth chair. If we label each seat, beginning with the small blind at 1, then the seat position for the player can be used as a starting point. If our agent were sat next to the button, then they would have a position score of 5.

To begin to formulate a score, let's call this variable hand_potential. Thus, so far:

```
hand_potential = position
```

To keep things simple, we could assign each card combination a score using 1 (worst) to 169 (best). Combining these terms into our hand_potential variable, we get:

```
hand_potential = position(hand_strength)
```

If we use the extremes for our range of possibilities, this means that we have two cases of comparison to judge our combination. The first is that at our six-person poker table, the worst possible hand in the best possible position is comparable to the sixth worst hand in the worst position. The second is that our 28th worst hand in the best possible position is comparable to a pair of aces in the small blind position. Clearly this simple combination doesn't really account for hand strength appropriately. We can fix this by either reducing the effect of the position on the final score or increasing the effect of the hand strength.

If we square the hand strength instead, we get a more satisfactory result. The worst hand in the best position is only better than the second worst hand in the small blind position. Whereas with this model, the best hand in the worst position has a worst score than the 70th worst hand in the best position, a little shy of half all-possible hands. If this is not satisfactory still, and this is a very simple modelling for the purposes of illustration, we could add some factor before we square the result. We might add ten to the pre-squared hand strength if it is a pocket pair and five to the suited cards to add a reasonable distinction. A small pair (generally considered to be less than ten) has more chance of becoming a three of a kind than suited cards have of becoming a flush or a straight.

hand_potential = position((hand_strength + (suited? + pair?))^2)

This model for hand potential may not be perfectly nuanced, but it is hopefully sufficient for a basic agent to help make its judgments about play decisions and basic enough to implement simply. Indeed, we might calculate hand strength according to the number of over cards as well, but we'll leave that distinction out for the moment. As we now have a judgment variable, we can apply similar principles of alteration as with the guard assessing their health and ammunition.

In this instance, we would see agents making raises when they should call or folding when they should call, etc. These changes would form a slight variance to the accepted strategies replicating what we see real poker players do, often to the chagrin of championship bracelet winners around the globe who take a dim view to such plays.

This leads us to the first of our potential indices for the personality vector: *experience*. Experienced players will more readily compute the conventions of play because it helps them size up their opposition. Experience will lead a player to better evaluate their hand and position in the game and cause them to act more accordingly with the conventions as they are tried and true. Inexperienced players, who maybe are less aware of their position and the conventions are more likely to make risky and daring moves if they move at all.

Likewise, with experience, a player might be described as a tight or loose player (which links to the concept of bluffing). A tight player is someone who works more with the value of their hand and position and acts strictly accordingly (the kind of player that will deviate less from the conventions of the strategy framework). A loose player, conversely, is the opposite. A simple implementation of these kinds of personality on a factor such as hand potential would be to observe if a tight player does not play a hand because one or two factors are not as optimal. This being the case, a player's style as index in the personality vector would act as a limiting or gaining factor for hand potential leading loose players to make riskier plays (in effect over valuing their cards) and tight players to make much more conservative plays (in effect under valuing their cards).

Therefore, based on the two examples presented in this section (patrolling agent and poker player agent), we can expect that the strategy framework can be formulated separately from the agent's interpretation, which would be then grounded on the personality vectors. These personality vectors can vary in many ways as described, from reactions to particular *stimuli* to the ability of being able to "lie."

In the next section, we discuss the applicability of these ideas into SG development. We hope to see the adoption of more personality-based reasoning for game agents, so it will lead to a better experience all round for the players of those games. We hope also that a general paradigm model for this can be developed in much the same way other reasoning models have been created.

7.4 Conclusions

This chapter proposed a conversation about aspects that can be "transferred" from games and then can be potentially utilized in SG development. To conclude this conversation, we would like to propose several ideas on how to take these concepts into practice.

7.4.1 Games as "Tools" and Games as "Instruments"

In Sect. 7.2, we have mentioned about the transferability of competences and skills constituted by gaming practices, particularly from the perspective of games as tools and games as instruments. In this case and in SG development, games could be repurposed if seen as a “serious toy,” meaning that the game itself would have multiple facets. This could be explained via a modular approach (similar to LEGO bricks), in which the game now seen as a “toy” could evoke multiple playing settings. In this particular case, *modding* can be a methodology to be applied and investigated, particularly the aspects that make a game “moddable.” Thus, by incorporating “moddable” affordances to the game, it might be that designers and developers could provide more modular approaches to development. Yet, questions still remain: How can a player learn how to modify the game? Can this be accessible to all players, or is it still for a minority of players who are familiar with development? The same can be applied for the discussion and title from [9]’s paper *You can’t bring that game to school* (2003); we may, as developers and designers, actually allow students to bring the games to school and change them. Since games become even more part of today’s society’s culture, the choice of which game to use for pedagogical reasons could be negotiated.

Another aspect mentioned in Sect. 7.2 was games as “instruments.” The transferability aspect of this category is competences and skills, which aligns with flow mechanisms, game balancing, and alignment of player-game knowledge (“know-how”). Thus, for SG development, this aspect might inform SG adaptability and controlling conventions and heuristics.

7.4.2 Early Development, Purpose, and Contextualization

In this chapter, we mentioned the benefit of GJs, particularly when developing games quickly and dynamically in teams. Serious contexts are common (e.g., sustainability, health, politics, etc.); thus, SG developers and designers might want to engage in defining small but effective game mechanics and processes in order to increase SG development quality in early stages. As noted by [13], SG effectiveness and playability tend to be evaluated in the end of product development, showing that there is still a gap for quality assurance during the early stages. We expect that perhaps via GJs (or similar participatory/co-creative events) and alignment of both research and design questions designers, developers and stakeholders can ensure SG quality but also develop innovative mechanics and design solutions together. In this case, player control and fantasy should be balanced with the learning (or behavioral) objectives, in order to inform better practices in SG development. Since playability aspects influence SG effectiveness [13], it might be that the mechanics need to be explored before aligning it with the pedagogical needs. Yet, design and research questions need to be equally balanced and addressed. Our examples in Sect. 7.2 showed how these can be addressed and perhaps this approach might help SG development in the early stages.

7.4.3 Personality Vectors, Emotional Modelling, and SG Development

Introducing personality vectors as part of the reasoning structure can add nuance and variance to the expected behaviors of non-player character agents in games in a way that increases the opportunity for unique and more lifelike behavior from those agents. What is more, the two-structure approach of keeping the personality vector separate from the strategy framework allows us to influence changes in the agent easily, cleanly, and with an overall simplicity that maintaining one structure does not allow and that does not require us to design multiple solutions for multiple agents. In essence, the strategy framework is the training manual, and the personality vector is the agent's interpretation. These ideas can be implemented toward intelligent tutors, for example, in order to generate more believability and perhaps even trust (even with an NPC being able to lie). In health, personality vectors could enhance the emotional response NPCs might have toward a sensitive topic or provide more human-like responses.

Yet, as a result of a conversation on "transferability" from general games to SG development, this chapter brings back into discussion particular questions that might be of interest and might enrich SG development and research, such as: How much of the game stems from the context? Can this be a multiplatform experience? What does it mean to design a "transferable" serious game? What can be learnt from current design processes to make these games more transferable? Can this be applied in all serious games?

We hope that there will be greater flexibility in what is understood as academic terms, since the opening is beneficial not only for the diverse realization of serious games but as an invitation to new looks at applied entertainment.

References

1. Alvarez, J., Djaouti, D., Louchart, S., Lebrun, Y., Zary, N., Lepreux, S., Kolski, C.: A formal approach to distinguish games, toys, serious games & toys, serious re-purposing & modding and simulators. IEEE Trans. Games. (2022).
2. Protopsaltis, A., Panzoli, D., Dunwell, I., de Freitas, S.: Repurposing serious games in health care education. IFMBE Proc. **29**, 963–966 (2010)
3. Chettoor Jayakrishnan, G., Banahatti, V., Lodha, S.: GOVID: repurposing serious game for enterprise COVID-19 awareness. ACM Int. Conf. Proc. Ser., 11–18 (2021)
4. Gómez-Rodríguez, A., González-Moreno, J.C., Ramos-Valcárcel, D., Vázquez-López, L.: Modeling serious games using AOSE methodologies. In: International Conference on Intelligent Systems Design and Applications, ISDA, pp. 53–58 (2011)
5. Ninaus, M., Nebel, S.: A systematic literature review of analytics for adaptivity within educational video games. Front. Educ. (Lausanne). **5**, 308 (2021)
6. Wen, A.: Before I Forget: the video game that tackles dementia | Games | The Guardian. https://www.theguardian.com/games/2018/jun/06/before-i-forget-early-onset-dementia-video-game. Accessed 17 Nov 2022
7. Schell, J.: The Art of Game Design, a Book of Lenses. (2008).

8. Lloyd, J.: How Hellblade: Senua's Sacrifice deals with psychosis | BBC Science Focus Magazine. https://www.sciencefocus.com/the-human-body/how-hellblade-senuas-sacrifice-deals-with-psychosis/. Accessed 17 Nov 2022
9. Jenkins, H., Squire, K., Tan, P.: You can't bring that game to school! In: Laurel, B. (ed.) Design Research: Methods and Perspectives, p. 334. MIT Press, Cambridge, MA (2003)
10. Abbott, D.: Modding tabletop games for education. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2019)
11. Csikszentmihalyi, M.: Finding Flow: The Psychology of Engagement with Everyday Life. Basic Books (1998)
12. Hunicke, R., Leblanc, M., Zubek, R.: MDA: a formal approach to game design and game research. In: Workshop on Challenges in Game AI, pp. 1–4 (2004)
13. Vargas, J.A., García-Mundo, L., Genero, M., Piattini, M.: A systematic mapping study on Serious Game quality. ACM Int. Conf. Proc. Ser. (2014)

Chapter 8
Code-Venture: A Mobile Serious Game for Introductory Programming

Leckraj Nagowah and Diksha Cuniah

Abstract In the last decade, there have been tremendous improvements in the IT field, and the demand for skilled professionals has ever since grown rapidly. For a better economic development, it is thus of primary importance for schools and universities to uncover and train new talents who will help propel our society's upward trend in IT and meet the increasing demand. On the other hand, there is a misconception among youngsters that programming is complex and not designed for everyone. Using the fact that nowadays games are becoming increasingly popular especially among the younger generation, a mobile serious programming game, *Code-Venture*, is being proposed in this chapter. Other than being fun and entertaining, the aim of *Code-Venture* is to help the players understand the basics of coding and sharpen their skills in programming. *Code-Venture* is based on the fundamental programming principles as recommended in the ACM/IEEE Computer Science Curricula 2013. Moreover, through the implementation of the teacher's application, which stores scoring information about the players of the game, a constant monitoring, assessment, and evaluation of the player's performance can be performed. Pre-game and post-game surveys have been conducted to evaluate the mobile serious game *Code-Venture*. Most of the 35 respondents found the game useful and engaging with a considerable increase in the number of students who are willing to join a career in programming and another increase in the number of respondents who now found programming easy.

Keywords Mobile serious game · Game-based learning · Educational games · Introductory programming · Serious programming game

L. Nagowah (✉) · D. Cuniah
Faculty of Information, Communication and Digital Technologies, Department of Software and Information Systems, University of Mauritius, Réduit, Mauritius
e-mail: l.nagowah@uom.ac.mu; diksha.cuniah2@umail.uom.ac.mu

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_8

8.1 Introduction

Technological innovations hold a key place in most sectors worldwide. It has become a compulsory tool to most people in the workplace and life in general. Nowadays, almost every service, such as online shopping, reservations, or even learning, can be accessed online or through mobile applications. What lies behind those applications is programming. While coding can be fun and exciting, to some it comes out as boring and tough or even stressful when traditional learning methods are used [1, 2]. Additionally, a high rate of failure coupled with students' lack of interest in programing have been reported in a number of studies [3].

It is well known in the computer science education (CSE) community that students struggle in programming classes, which can lead to high dropout and failure rates [4]. In 2007, Bennedsen and Caspersen [5] estimated through a study that out of 2,000,000 IT students worldwide, about 650,000 failed every year. They then replicated their study in 2019 and observed that the failure rate was still high at 28% [6]. Fundamental programming and computational thinking skills are crucial in the field of computer science whereby one might find it challenging to enter the world of IT without these core skills [7].

On the other hand, Halbrook et al. found that 95% of homes with children of under 18 years of age own some form of video-game platform [8]. As a matter of fact, researchers are investigating game-based computer science learning [9, 10] where numerous games are being developed that mainly focus on computer programming and aim at helping students grasp the fundamental programming concepts. Mathrani et al. [11] confirmed the effectiveness of serious games as a means of teaching and learning. Students who participated in the survey proved that game-based learning made them more engaged in the use of programming principles through gaming steps, as they were able to visualize the programming constructs and thus get a better understanding of how it works. Boeker et al. [12] noted that a conventional approach was not very effective when learning programming. Thus, using serious games as a means of teaching made the normally boring classroom environment more fun and enhanced the player's ability to grasp difficult concepts. Ding et al. also confirmed through their study that game-based learning is more effective, easier to grasp, and more preferred by students than traditional learning methods [13].

The primary aim of this work is to develop a mobile serious game that aims at enhancing the programming skills of beginners, together with a mobile application for teachers who will assess, monitor, and evaluate the performance of players. The chapter also discusses the results obtained from a pilot study with 35 students aged between 16 and 21. The authors also plan to conduct a more structured experimentation in the future to assess the perceived improvements in the programming skills of the users of the mobile application. A structured approach was used to implement our mobile serious game and is described in the subsequent sections. The remainder of this chapter is organized as follows. In Sect. 8.2, a background on the related works has been given, and an analysis of the existing works has been provided in Sect. 8.3. Section 8.4 highlights the architecture and the main components of *Code-Venture*. The implementation and testing of the application are presented in Sect. 8.5. A discussion is presented in Sect. 8.6, and finally, Sect. 8.7 concludes the chapter.

8.2 Background Study

A game refers to an organized play with rules, goals, and challenges for the purpose of entertainment [14]. The term gamification was first coined in 2008, and in contrast to normal games, it is characterized by its serious purpose. Researchers agree that gamification usually focuses on game elements and mechanics in serious contexts. Gamification therefore includes the use of game elements in non-game contexts. Game elements include levels, points, badges, leader boards, avatars, quests, social graphs, or certificates [15, 16].

Gamification is closely linked to two related concepts, namely, game-based learning and serious games. Game-based learning is the achievement of defined learning outcomes through game content and play, as well as boosting learning by involving problem-solving areas and challenges that offer learners, who are also the players, with a sense of accomplishment [17]. As the name says, game-based learning aims at educating the players. It however relies on a fully fledged game, commonly known as a serious game. A serious game refers to playful interactive applications whose primary purpose other than being fun and entertaining includes education, training, analysis, visualization, simulation, health, and therapy [18]. Serious games and game-based learning therefore differ from gamification due to the fact that they are fully functional games. They all share the idea of employing pleasant gameful experiences for the benefit of a serious goal, such as education or behavior modification, rather than focusing on enjoyment. Gamification as a broader concept merely takes components of games and applies them to the real environment [15].

8.2.1 Related Works

To review existing serious games on programming, two searches have been carried out: one on Google Scholar to find some related works and another one on common app stores to look for existing commercial applications. This section presents some of the findings.

Jemmali and Yang [19] developed a serious game called “May’s Journey,” which targets middle school and high school students. This game was primarily designed to encourage girls to choose programming as a field of study. It was a 3D puzzle game where the player had to interact with the environment to solve mazes involving basics of programming. The puzzles focused on concepts and logics but still allowed the player to type programming instructions to bridge the gap between real programming and coding in a game. The game was tested with ten teenagers aged from 14 to 17 years for educational content, and the authors reported that the teenagers were engaged with the game.

Du et al. [20] made a study about the Hour of Code by Code.org, which is a 1-h introductory tutorial to programming, making use of visual programming. The game uses blocks to program a solution for different puzzles. After solving each maze, the player received a positive feedback and advanced to mazes that are more complex. One hundred and sixteen students from two universities participated in the study, and the game proved to have a positive impact on the students' attitude toward programming. However, the game also turned out not to have a significant effect on the player's actual coding skills.

Miljanovic and Bradbury [21] developed a serious game with a systematic approach to focus on programming comprehension rather than writing codes. Thus, novices without knowledge of programming could play this game and gain some basic understanding. The player had to accomplish several comprehension tasks in order to activate a Mech Suit system. While advancing in the game, the player was able to develop a concise understanding of variable values, data types, program statements, and control flow, all of which were repeatedly tested throughout the gameplay.

Law [22] investigated on how to enhance the iteration, selection, and building of command blocks in programming through the use of video games. The skills to be developed were problem-solving skills and computational thinking skills in order to get the required result. The author used the freely available "Program your robot" game that targeted these skills and allowed the player to visualize the abstract concepts in programming. Despite the fact that the pilot study was carried out with only 42 students, the findings and the student comments showed positive results and indicated that it would be worthwhile to expand the study to a broader cohort pursuing a wider range of computing programs.

Junaeti et al. [23] conducted research about teaching basic programming concepts using the genius learning strategy. "Array Adventure" was a serious adventure game designed to target the principles behind arrays. The first level involved one-dimensional arrays and the second level catered to multi-dimensional arrays. The game was set in a 2D environment whereby the player had to complete missions in an adventure style gameplay. The game was evaluated by 30 students and 2 experts and positive results were obtained.

Jordaan explored the likelihood of making use of board games to improve the learning experience of computer science students. The findings of this study demonstrated that students appreciated the dynamic learning atmosphere provided by board games and that they accepted them as a fun and enjoyable method of instruction [24].

Lotfi and Mohammed [25] presented a mobile serious game that taught object-oriented programming concepts for beginners. "OOP Serious Game" was set in a zoo environment where the player had to create animal classes and understand the methods that were behaviors, actions, and voices. The elements of class, object, and complex paradigms like inheritance and polymorphism were taught through gamification techniques in order to facilitate the learning of these concepts. The game used an in-game assessment mechanism to gauge the player's knowledge of four OOP concepts, namely, class, object, inherence, and polymorphism. However, the results were inconclusive as to the game's efficacy in teaching the programming concepts to the players.

Yallihep and Kutlu [26] analyzed and evaluated the effect of a mobile serious game called "LightBot" for learning programming. A 5-week study was carried out in a primary school in Turkey with 36 fifth-grade students. According to the research, the game positively influenced the students' achievements. Complex concepts like recursions and procedure were taught at an early stage to students in a gamified learning approach.

Zhao et al. [27] proposed a serious game that focuses on the structure of the C programming language. "Restaurant game" was a 2D game that incorporated programming concepts like data types, variables, and structures. The player was required to engage with various game objects, which were data types' representations in a restaurant, and, as a result, gain a deeper knowledge about application of these programming concepts. Ninety first-year students tested the serious game, and the results showed that the improvements in learning outcomes were statistically significant. More than half of the participants were of the opinion that the game may help them get better grades in the programming course, and more than 60% of the participants said it improved their understanding of programming topics.

Karram analyzed "Code Combat," which is a popular game that targets object-oriented programming concepts in a game-based format [28]. The game offers an engaging and fun environment whereby the player has to complete tasks and challenges to earn points and level up. Rewards such as badges and rankings also make the player more motivated to complete the levels and thus learn the concepts along the way. The game guides the player to type the appropriate code lines in order to assign tasks to the virtual characters and thus complete the puzzles. Code Combat makes difficult concepts like inheritance, nested loops, and recursion simpler through a gaming approach.

Toukiloglou and Xinogalos [29] developed "NanoDoc" that taught programming concepts through a first-person shooter game. The players had to acquire a key by solving programming puzzles to navigate through the different rooms. The programming environment featured a hybrid 2D/3D mode where the player created a program to control the avatar's movement in a 3D grid. The solution algorithm was constructed with a 2D block-based programming environment using colored blocks that could be connected and manipulated through drag and drop actions. The game proved to be effective in improving students' motivation, engagement, and learning outcomes, as well as in reducing their anxiety toward programming.

Akkaya and Akpinar [30] designed a game aimed at teaching the fundamental concepts of object-oriented programming and computational thinking skills to students. The game adopted a constructivist learning approach set in a fantasy environment with metaphorical machines to make the abstract concepts concrete. It included interactive tools such as class and method definer machines for students to program robots. A pedagogical agent provided instructions, support, and feedback, and the game had a visual and textual feedback mechanism to understand the code execution. The game aimed at teaching students the importance and applications of object-oriented programming and computational thinking and also eased their introduction to algorithmic thinking. The game was tested by 61 students with and without prior programming knowledge, and the results showed that the students improved their understanding of the fundamental concepts of OOP.

8.2.2 Commercial Apps

Popular games about coding were also searched for on common app stores with search terms like “serious game programming,” “programming game,” and “coding game” to search for existing serious games about programming. Some of the main findings are listed below.

- Hacked

In Hacked, the player impersonated a hacker who needed to solve some problems with codes and save the world [31]. It had a progressive difficulty and offered a wide variety of options ultimately guiding the player to develop his personalized game. The player was required to have some prior knowledge on programming before attempting the game. The features included were performance tracker, assistance in writing of codes, level system, problem-solving skills, reward system, and competition with other players.

- Coding Planets

Coding planets required the user to solve puzzles through commands issued to a robot [32]. All age groups are targeted allowing them to sharpen their programming skills and gain fair knowledge of coding. The players needed to use their logic to advance through the different levels while developing their problem-solving skills. The main features included were as follows: improvement of problem-solving skill, development of logical thinking skill, sequencing, looping, functions, use of command icons to issue instructions, beginner and advanced difficulty, and reward system.

- LightBot: Code Hour

This game introduced programming concepts for beginners [33]. It consisted of commanding a robot to light up tiles by giving it instructions. The skills targeted were basic concepts like sequencing, loops, and procedures. The game had good reviews whereby players affirmed that they were able to learn about programming concepts in a fun and interactive way. The features included were learning of programming practices like planning, programming, testing, and debugging; development of problem-solving skills; learning about control flow concepts like functions, sequencing, and loops; and programming through commands.

- SpriteBox: Code Hour

This was a puzzle-platformer and adventure game allowing the player to venture through different worlds and using code to complete the objectives [34]. It targeted all players regardless of their programming knowledge and consisted of 20 levels of challenging puzzles. The features included were icon-based programming, sequencing, parameters, debugging, loops and problem-solving, exploration while learning, and beginner and advanced levels.

- Meoweb

Meoweb used fancy displays to make the process of learning programming more fun and approachable [35]. The puzzle games consisted of the manipulation of codes in order to solve the set problems. Logical thinking was also required from the players to advance through different obstacles and levels to complete goals and ultimately reach the final destination. The basic concepts of CSS programming could be grasped by completing the levels. The main features included were as follows: development of problem-solving skill, reward system, leveling system, and logical thinking.

- BeBlocky: Kids Code Easy

BeBlocky was an engaging game that taught basic programming [36]. Target players were mainly children and aspiring novices. The player encountered several robots that needed to be programmed by dragging and dropping programming blocks in a sequential way. The features included were memory boosting; developing aptitude in sequence, loops, and commands; improvement of problem-solving skills; development of logical reasoning; and leveling system.

- Coding Galaxy

The game provided an interactive and user-friendly interface for learning about basic programming concepts [37]. It was designed and reviewed by skilled teachers and specialists who incorporated core methods traditionally used for teaching programming in the system. The targeted players were students aged 5 and above. The game consisted of more than 200 levels whereby the player was expected to complete missions and objectives and solve programming puzzles. The features included were development of computational thinking, problem-solving, critical thinking, communication and leadership skills, development of creativity and teamwork, learning through adventure and quest system, monitoring of user performance, learning report, sequence, looping, conditional logic, function, and parallelism.

- Grasshopper: Learn to Code for Free

Grasshopper consisted of several mini-games guiding players toward all the basic concepts needed in the JavaScript programming language [38]. The player needed to use codes to solve puzzles. Upon completion of all the game levels, the player should be able to write basic JavaScript codes. The features included were calling functions, variables, strings, for loops, arrays, conditionals, operators, objects, arrays, recursions, and HTML.

8.2.3 Skills Required for Introductory Programming

After thorough analysis of the selected articles, the important skills deduced to be imperative when it comes to learning introductory programming have been highlighted in Table 8.1. The logical skills have been devised from the literature, while the technical skills were based on the ACM/IEEE-CS Joint Task Force on Computing Curricula, 2013 [45].

8.3 Analysis

The previous section gave an overview of some of the related works on serious programming games. This section provides an analysis of those works with respect to the ACM/IEEE guidelines for the programming module. The comparison table visually highlights the main features of the serious games surveyed from the literature and those commercially available.

8.3.1 Comparative Analysis of the Related Works

The related works have been analyzed with respect to the ACM/IEEE-CS Joint Task Force on Computing Curricula 2013 [45]. More specifically, Table 8.2 shows how the related works try to address the skills highlighted by ACM/IEEE in terms of algorithms and design, fundamental programming concepts, and development methods. As it can be observed from the table, *Code Combat* is by far the game that provides for most of the features as recommended by ACM/IEEE followed by *Hour of Code* and *May's Journey*.

8.3.2 Comparative Analysis of the Commercial Games

Table 8.3 shows an analysis of the existing commercial programming games with respect to their gaming features and the logical and technical skills targeted for the commercial games. As it can be observed from Table 8.3, *Hacked* is the game that has the highest number of features followed by *SpriteBox* and *Coding Galaxy*. These three games attempt to cater to a high number of technical skills and at the

Table 8.1 Programming skills table

Logical skills	
Problem-solving	One of the factors that affect students' academic performance when it comes to learning introductory programming is low problem-solving skills. According to a study, a high number of novice students failing programming courses was due to a lack of problem-solving skills. Training through activities that help improve this skill can be highly beneficial to students' performance in computing studies [39]
Debugging	Debugging can be considered as a form of "learning from mistakes" strategy. It is well known that students can effectively learn through their mistakes, and in this case, debugging skills are developed when the students have to browse their own code to identify the source of the problem. This is yet another essential skill required to become a good programmer [40]
Testing	Improving testing skill is crucial to increase productivity. Without this skill, a beginner might have difficulty in making a correct working program which has all the required functionality. This skill should be developed from the start and worked on to have a positive impact on the aspiring programmer [41]
Algorithmic thinking	A study proved most students learning introductory programming had underdeveloped algorithmic and computational skills [42]. This skill is crucial for beginners in programming as this is what will enable them to define clear, concise steps to solve any problem, which is basically what the basis of programming is about. Basically, algorithmic thinking consists of the following: • The ability to analyze given problems • The ability to specify a problem precisely • The ability to find the basic actions that are adequate to the given problem • The ability to construct a correct algorithm to a given problem using the basic actions • The ability to think about all possible special and normal cases of a problem • The ability to improve the efficiency of an algorithm Improving this essential skill which has a strong creative aspect includes solving a maximum of problems. By providing the student with simple problems with gradual increasing difficulty, the latter can effectively work on this skill [43]
Sequencing	This is a common process for writing codes. Instructions are given in a specific order, and the computer processes and executes them accordingly. The development of this skill allows the programmer to think like the computer and hence solve programming problems more efficiently. This is yet another core skill required for novice programmers [44]

(continued)

Table 8.1 (continued)

Technical skills	
Variables and datatypes	These are the basics of computer programming. Every code makes use of variables and data types to solve problems. This knowledge is vital to be able to write codes effectively
Functions	Functions also form part of the basics of programming language. Use of functions allows code to be modular, clear, and concise to be able to write good-quality codes
Loops	The ACM Computer Science curricula 2013 also includes the concept of loops as the programming fundamentals. Beginners in programming must understand how loops work and how to include them in their code in order to effectively solve problems
Arrays	Fundamental data structures also include arrays that are another important factor in writing codes. The use of arrays is very common, and many problems require this concept to be able to tackle problems accordingly
Decisions/conditions	Fundamental programming concepts include decisions/conditions that is yet another core element in programming. ELSE and SWITCH statements are very common in programming and crucial for solving problems

same time aim at developing the logical skills of the player. They also have several gaming features, as one would expect from a normal game.

8.3.3 Summary of Findings

Most related works concluded that a serious game for learning programming positively influenced students' academic performance. For instance, Yallihep and Kutlu [26] conducted a research using a popular mobile serious game "LightBot," and Du et al. [20] also conducted an evaluation of a programming game, which showed a great increase in motivation as well as performance in students. Several games were developed that aimed at programming comprehension by combining programming concepts with gaming mechanisms [21, 22, 25, 27]. This method proved highly beneficial as it made learning process more fun compared to traditional ways. Learning programming through gaming mechanisms proved to reduce anxiety and significantly improve the learning curve in students [29, 30]. Karram [28] investigated *Code Combat*, which proved to accelerate and improve the learning process of students. The game targeted several important technical skills as per the guidelines of the ACM/IEEE. However, the gameplay of *Code Combat* seems to follow the same format for all the different levels. It can eventually be deduced that gamified learning has proved to be a great way to learn the basics of programming. It was also observed that a series of logical skills were also imperative when it came to learning programming. Computational thinking skills like problem-solving, sequencing, debugging, and algorithmic thinking to facilitate

Table 8.2 Comparative analysis of games from the literature

	Algorithms and design			Fundamental programming concepts									Development methods	
Serious game	Problem solving	Algorithmic thinking	Sequencing	Non-specific programming concepts	Syntax and semantics	Variables and data types	Expressions and assignments	Input and output	Conditionals and iteratives	Functions and parame-ters	Recursion	Arrays	Debugging	Testing
May's Journey [19]	✓		✓	✓	✓	✓	✓					✓	✓	
Hour of Code [20]	✓	✓	✓	✓				✓	✓		✓		✓	
Robot ON! [21]	✓		✓	✓		✓	✓							✓
Program your robot [22]	✓	✓	✓						✓		✓			
Array Adventure [23]	✓			✓		✓			✓			✓		
Board Game [24]	✓	✓												
OOP Serious Game [25]	✓			✓						✓				
LightBot [26]	✓	✓	✓						✓		✓		✓	
Restaurant game [27]			✓			✓								
Code Combat [28]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NanoDoc [29]	✓		✓	✓										✓
Curious Robots: Operation Asgard [30]	✓		✓	✓										✓

Table 8.3 Comparative analysis of existing features of commercial games

Category	Features	Hacked [31]	Coding Planet [32]	Light-Bot [33]	Sprite-Box [34]	Meo-web [35]	Be-Blocky [36]	Coding Galaxy [37]	Grasshopper [38]
Gaming features	Adventure/exploration type				✓	✓		✓	
	Multiple difficulty level		✓		✓				
	Tracking of player performance	✓						✓	✓
	Report on performance						✓	✓	
	Reward system	✓			✓	✓	✓		✓
	Tutorial/in-game assistance	✓			✓				
	Leveling system	✓	✓		✓	✓	✓	✓	
	Challenge other players/interaction	✓						✓	
Skills developed	Develop planning skill	✓		✓	✓	✓			
	Improve testing skill			✓					
	Debugging skill			✓	✓				
	Problem-solving skill	✓	✓	✓	✓	✓	✓	✓	✓
	Memory boost		✓				✓		
	Logical thinking	✓	✓	✓	✓	✓	✓	✓	
	Critical thinking							✓	
	Teamwork	✓						✓	
Technical concepts	Use of real codes	✓			✓	✓			✓
	Icon-based programming	✓	✓	✓	✓		✓	✓	
	Functions	✓	✓	✓	✓			✓	✓
	Loops	✓	✓	✓	✓		✓	✓	✓
	Sequencing	✓	✓	✓	✓		✓	✓	✓
	Variables and datatypes	✓				✓			✓
	Parameters	✓			✓				

learning process were brought to attention. Finally, a gamified learning approach to learning programming showed a very positive effect in teaching programming classes. Therefore, games targeting the skills required for programming can greatly improve the academic performance of students studying the subject or even teach beginners about how programming works and introduce them to the world of algorithms and computational thinking. The main game elements used in most of the games were levels, points, leader boards, avatars, and quests. Varying the types of games is vital in order to ensure that the players are motivated and engaged. It is also important for the players to have regular and personalized feedback. Most of the games analyzed do not provide for these two important factors, hence the aim behind our system, *Code-Venture*.

8.4 High-Level Architecture

The high-level architecture is a way of representing how the whole system works and shows the interactions between the different modules. Figure 8.1 shows the high-level architecture of *Code-Venture* consisting of the mobile device, an online database, and a mobile application for teachers.

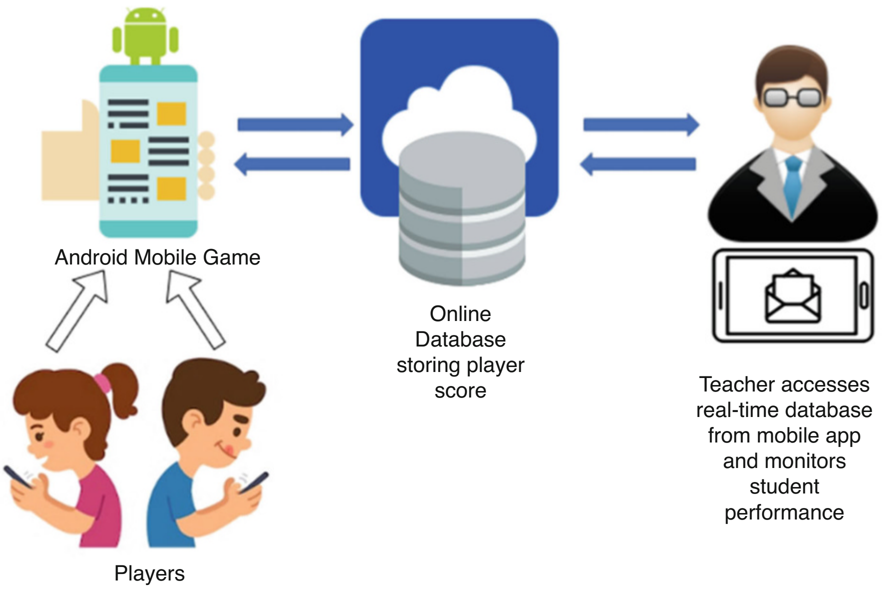

Android Mobile Game
Online Database storing player score
Teacher accesses real-time database from mobile app and monitors student performance
Players

Fig. 8.1 High-level architecture

The main components of *Code-Venture* are as follows:

Players: The players targeted are aspiring programming students, novice programmers, or teenagers with little or no experience in the field. Players will record their personal details and attempt the games through their mobile phones.

Mobile device: The mobile device is where the *Code-Venture* mobile app that consists of different gameplay will be downloaded, installed, and played. Storing of scores is made possible and pushed to the online database.

Online database: The information collected from the game, that is, the player details, scores, and areas of weaknesses, will be saved on the online database for future access by the tutor.

Teacher's mobile application: The students' information retrieved from an online database is displayed on a mobile application designed especially for the teacher. The latter can review player's score, weaknesses, and overall performance of the students on the different games through this application. This app will also allow the tutor to send personalized messages to the students.

8.4.1 The Mini-games of Code-Venture

As compared to existing serious programming games, *Code-Venture* will consist of different mini-games, namely, the *Adventure Mode*, the *Variable Runner*, the *Algorithm Puzzle*, and the *Quiz*, where each of the mini-game targets different skills. Table 8.4 shows the mini-games of *Code-Venture* and the skills that each game targets.

Table 8.4 *Code-Venture* mini-games and the skills targeted

		Code-Venture mini-games			
Category	Skills	Adventure Mode	Variable Runner	Algorithm Puzzle	Quiz
Logical skill	Problem-solving	✓		✓	✓
	Debugging	✓			✓
	Testing				✓
	Algorithmic thinking	✓		✓	
	Sequencing			✓	
Technical skill	Variables and data types	✓	✓		✓
	Functions	✓		✓	
	Decisions	✓			✓
	Loops	✓		✓	✓
	Arrays	✓			

8.4.2 Justifications for Code-Venture's Mini-games

Code-Venture includes several mini-games to differentiate it from existing serious programming games. Giving the player's the opportunity to play a variety of games ensures that the player is exposed to several gameplays and hence remains engaged and motivated. The more engaged and motivated a player is, the more the benefits that can be obtained from playing the serious game. The different mini-games of *Code-Venture* have been carefully planned and are backed by evidence from the literature whereby similar implementations have been successful in educating the players. Table 8.5 describes *Code-Venture* mini-games in more detail and gives justifications as to why these games have been chosen.

8.5 *Code-Venture*'s Implementation and Testing

Code-Venture was implemented with Unity 2019.2.15f1 personal edition and Visual Studio 2017. The C# programming language over .NET2.0 framework was used to program the different mini-games. The computer used consisted of a fourth-generation Intel core processor, a RAM of 8GB, and a 250GB SSD together with an NVidia graphic card GTX 1050ti. Two mobile phones were used to test the application, namely, a Samsung A20 (Android 9.0) and an HTC Desire 828 (Android 5.1.1) having a RAM of 3 and 2 GB, respectively. Different tests were carried out on both emulators and the actual mobile devices including a user acceptance test with 35 students most of whom were to embark on undergraduate studies.

8.5.1 Code-Venture's Main Functionalities

The main functionalities of *Code-Venture* are described and illustrated below.

- Register or Login and Main Menu

The player should enter login details and select “Play Now” on opening the game if he/she is an existing user, as shown in Fig. 8.2. In case of a new user, the “Register” button registers the necessary user details. After signing in, the user is then presented with the menu screen where the different game modes can be accessed as shown in Fig. 8.3.

Table 8.5 Justifications for *Code-Venture* mini-games

Proposed game	Game description and targeted skills	Evidence from literature
Adventure Mode (Coding Adventure)	This is the main game that consists of a 3D environment whereby the player is able to explore freely and solve programming puzzles at the same time. The player is able to trigger dialogues with non-playable characters in the game and accept quests from them. Each of the quests targets different types of mini-games that address different programming problems • Code-rearrange—rearrange bits and pieces of a code in the correct order • Find-error—choose the line of code which is causing an error in the code • Find-output—determine the output of the code when run • Fill-blanks—write the correct code in the empty boxes provided in a piece of code The game also has a "hard mode," which is the same story and environment but with more complex and challenging programming puzzles	Junaeti et al. [23] concluded through research that learning in the form of adventure games had a positive effect on the players' learning process. Players proved to be more motivated and engaged while playing this type of game Jemmali and Yang [19] made a 3D fully immersive adventure game with an interesting storyline to draw the players' attention. The implementation of programming concepts in a visual and interactive way proved to enhance the players' willingness to keep playing. Implementing programming concepts in this environment made the player have a fun gaming experience instead of the feeling of being forced to learn something that may demotivate the player quickly
Variable Runner (game for learning concept of variables)	The game is set in a platform where the player can move the hero left or right as the latter keeps running forward. While running, the hero encounters balls with different values, which are the possible answers to the question being displayed. The player is first presented with a question text and is expected to select the appropriate answer by colliding with it. A scoring system is also included whereby each correct answer adds to the score and vice versa	Miljanovic and Bradbury [21] made use of a gaming mechanism to tackle concepts like variables and data types. A puzzle-type game proved to be effective in helping to learn as well as improving skills like critical thinking and problem-solving Zhao et al. [27] investigated and came up with a game that teaches concepts of variables through foods set in a restaurant. The player is expected to engage with the game objects that represent data types in order to gain a deeper knowledge about this programming concept

Algorithm Puzzle (game teaching basic programming concepts through commands)	This game aims at teaching the player the basic programming concepts. The player has to issue commands to the hero, and the latter will execute the instructions accordingly. Icons are used to give instructions that are then applied to the hero upon execution. The skills targeted are as follows: • Computational thinking skill • Problem-solving • Planning • Testing • Debugging • Logical and critical thinking skill Furthermore, the following basic high-level programming concepts are also involved: functions, loops, and sequencing	Du et al. [20] concluded that visual programming with the use of blocks was effective in making players better understand programming concepts. Base skills like computational thinking skills, problem-solving, and debugging were developed, which further helps master programming Karram [28] deduced from the popular game *Code Combat* that learning programming through the use of commands and instructions proved to be beneficial in teaching players the basics of programming
Quiz (game consists of a series of questions to test the players' current knowledge)	This game tests the players' knowledge about programming. It also acts as a learning game since it consists of several multiple-choice questions that have to be attempted by the player. The progress and correct or wrong answers are recorded for future evaluation of the performance of the player. The incentive provided to play this quiz game and excel in it is that there are rewards that are obtained if the questions are correctly answered. This motivates the user to attempt the questions seriously and as a result gain further knowledge on programming. The game is adaptive, that is, the complexity of the questions increases as the player progresses in the game	Junaeti et al. [23] made use of incentive and competition mechanisms in order to assess the performance of the player through various quizzes after each game level. The results proved to be successful since players revealed that they gained knowledge on programming by answering the questions Lotfi and Mohammed [25] made use of in-game assessments for each game level in order to evaluate the player about four OOP concepts. The learner's performance was monitored through a scoring and timing system

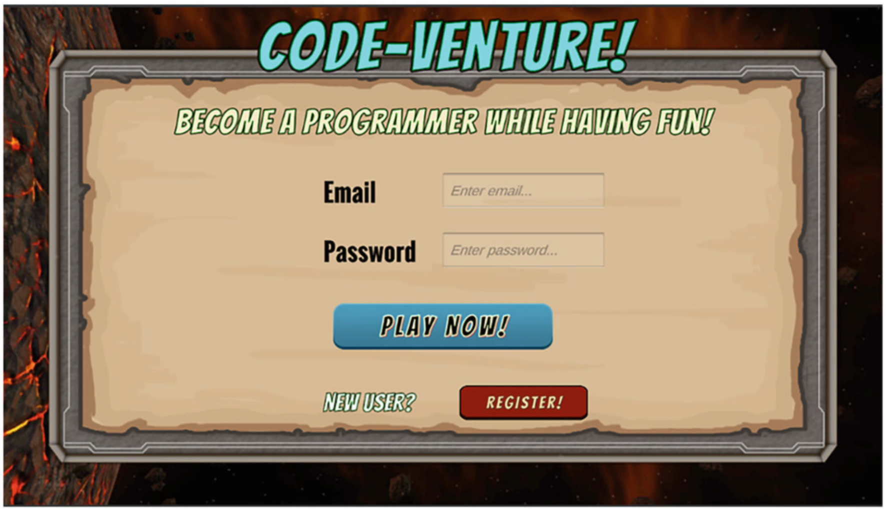


Fig. 8.2 Main menu—login or register

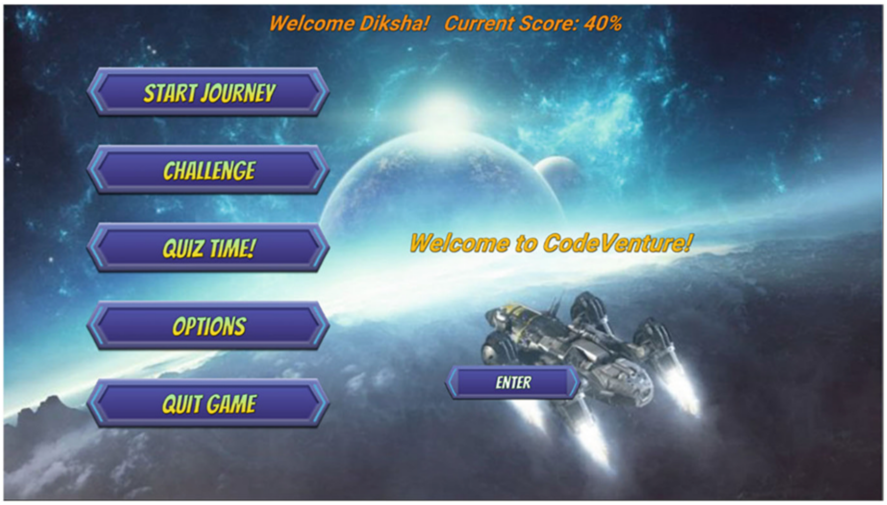


Fig. 8.3 Access main menu with player details

- Adventure Mode and Different Mini-Games

Upon choosing the Adventure mode, the player is able to move the character around the 3D environment and rotate the camera (Fig. 8.4). The player can gather collectibles, trigger dialogues, accept quests, and view the current progress as illustrated in Fig. 8.5.

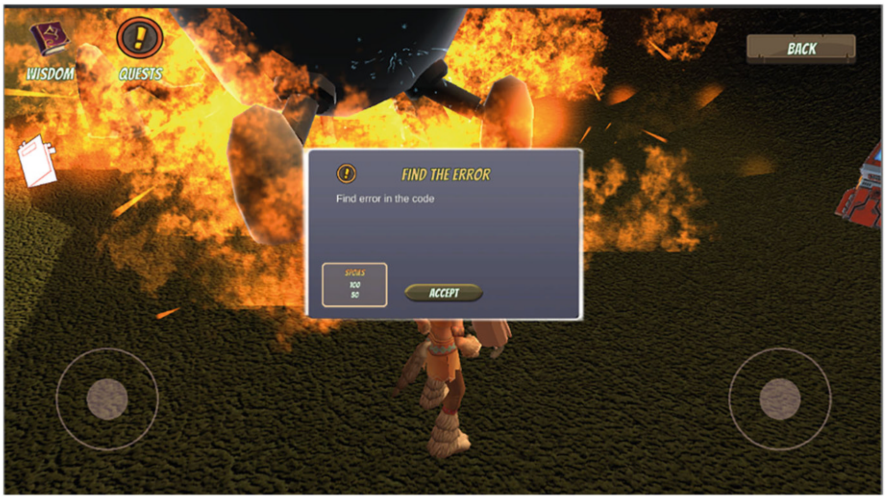


Fig. 8.4 Adventure mode—accept quest

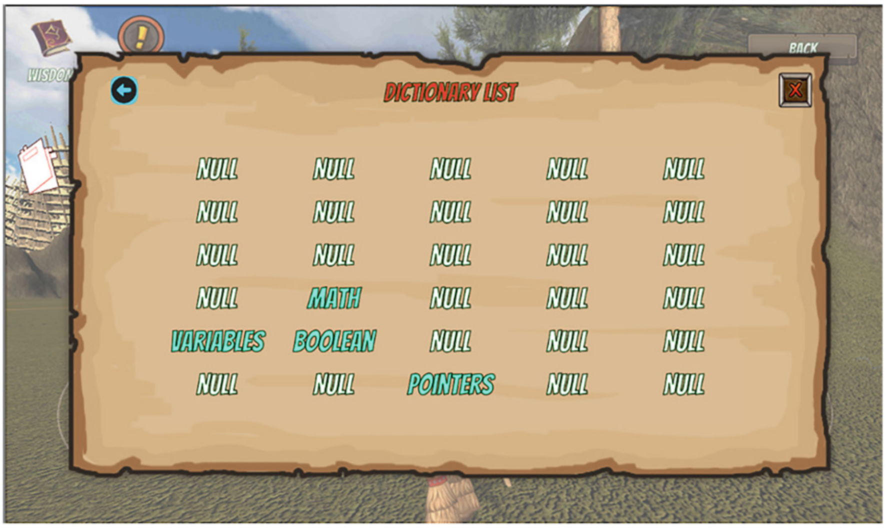


Fig. 8.5 Adventure mode—view progress

- Challenge Mode, Variable Runner, and View Score

The player can select the Challenge mode game option and choose to play "Variable Runner" whereby he is able to move the character left or right to choose the ball which corresponds to the correct answer to the question provided as shown in Fig. 8.6. The player is presented with final score (Fig. 8.7) when the timer is over.

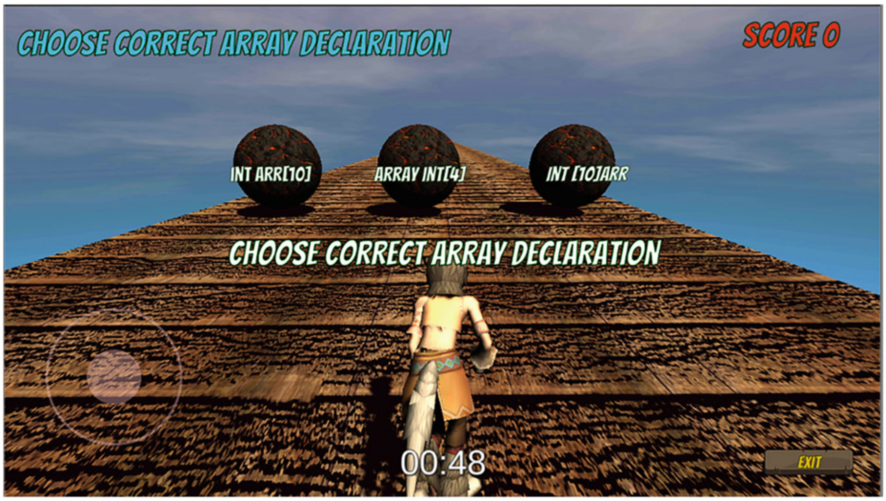
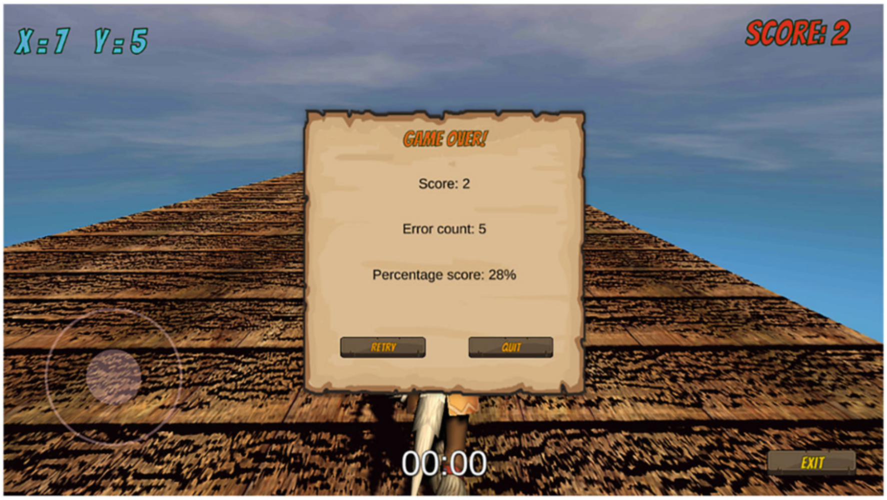

CHOOSE CORRECT ARRAY DECLARATION
SCORE 0
INT ARR[10]
ARRAY INT[4]
INT [10]ARR
CHOOSE CORRECT ARRAY DECLARATION
00:48
EXIT

Fig. 8.6 Variable runner game



Score: 2

Error count: 5

Percentage score: 28%

Fig. 8.7 Variable runner score display

- Challenge Mode, Algorithm Puzzle, and View Score

Upon choosing the "Challenge Mode" game option and entering the "Algorithm Puzzle" game, the player is presented with a scene with interface commands as shown in Fig. 8.8. The player can make use of different functions and loops in order to complete a level. The character can be moved by choosing different commands, and the score is displayed after each level is completed (Fig. 8.9).

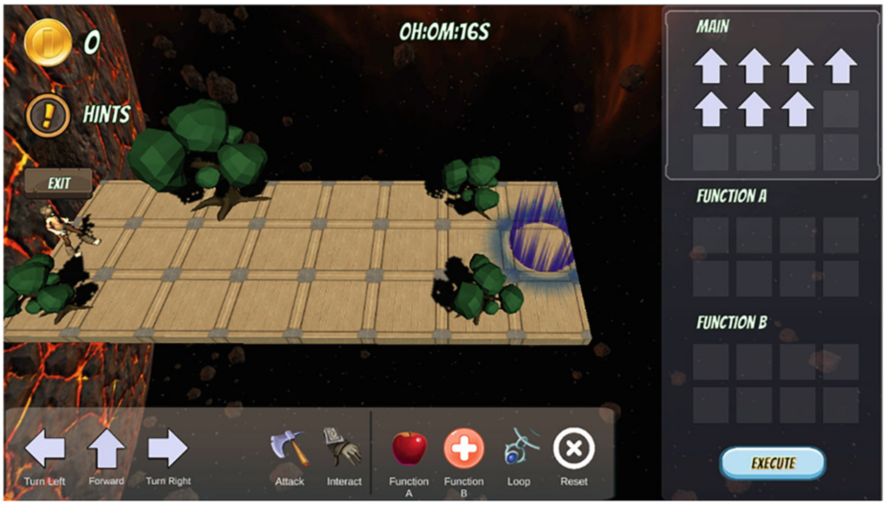


Fig. 8.8 Algorithm puzzle game

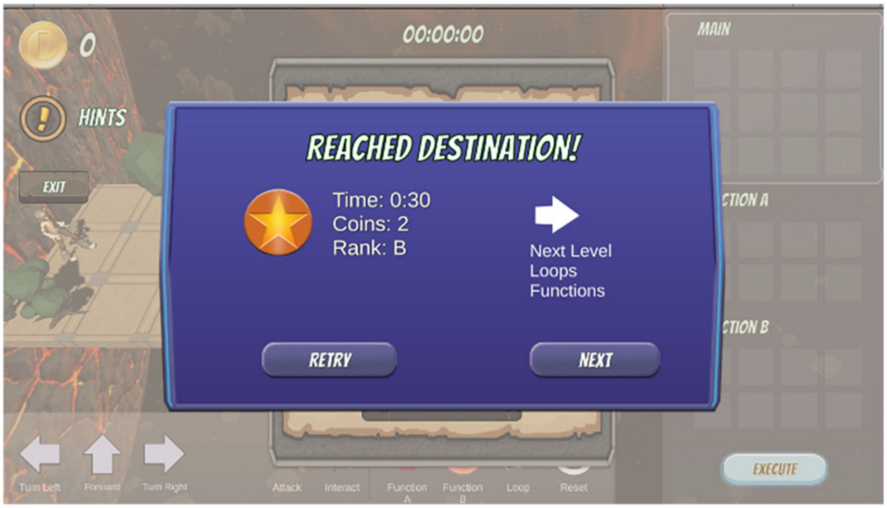
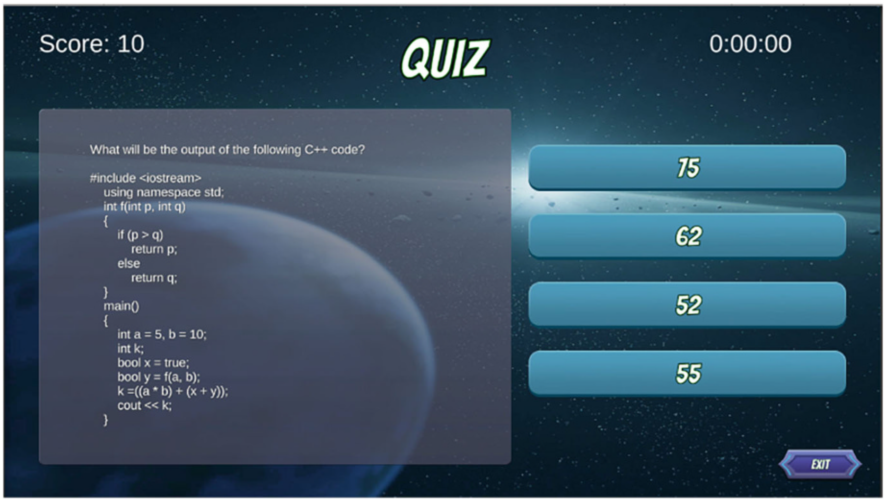


Fig. 8.9 Algorithm puzzle score

- Quiz Game and Score

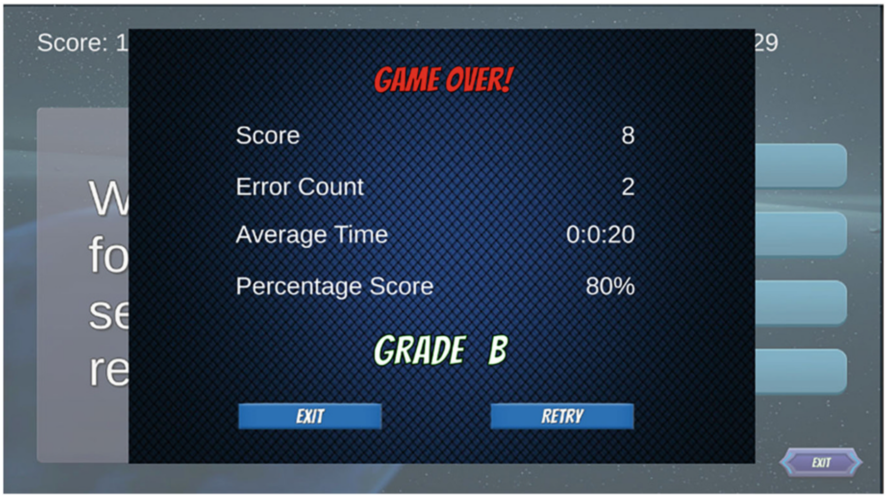
After selecting the quiz game, the player is presented with a question together with four possible answers, as shown in Fig. 8.10. The player must choose an answer whereby the correct answer is highlighted in green, while a red color is used for selection of an incorrect answer. After attempting ten questions, the player is given his final score together with the grade, error count, and time taken to complete the questions (Fig. 8.11). The quiz consists of several levels where the difficulty increases as the player progresses in the game.



What will be the output of the following C++ code?

```
#include <iostream>
  using namespace std;
  int f(int p, int q)
  {
    if (p > q)
      return p;
    else
      return q;
  }
  main()
  {
    int a = 5, b = 10;
    int k;
    bool x = true;
    bool y = f(a, b);
    k =((a * b) + (x + y));
    cout << k;
  }
```

Fig. 8.10 Quiz game

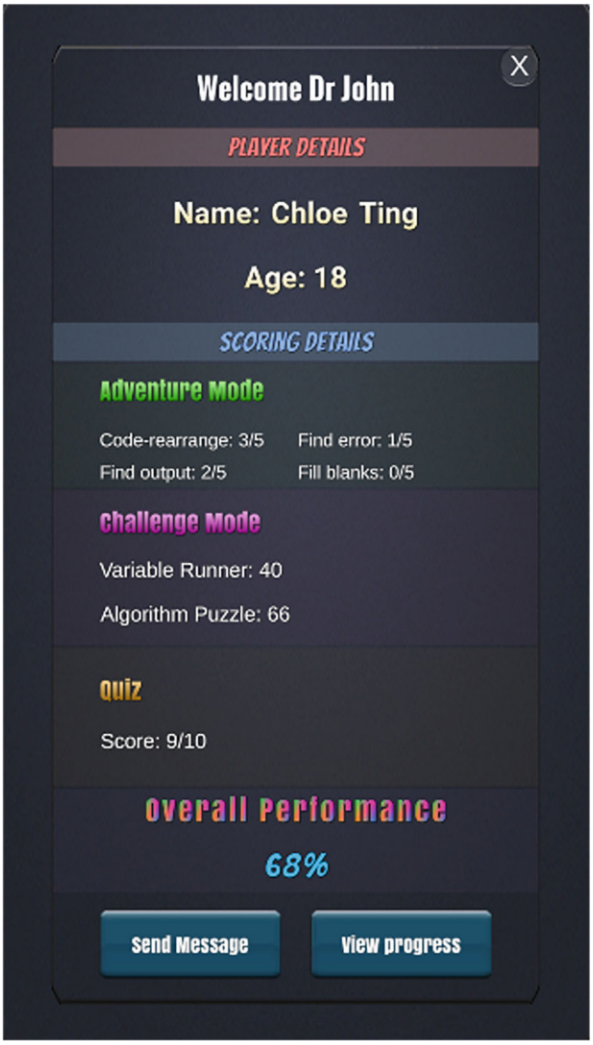


```
Score                     8
Error Count               2
Average Time         0:0:20
Percentage Score        80%
```

Fig. 8.11 Quiz game score

- Teacher Application

The teacher can sign in by entering his/her login credentials and then get access to the list of players of *Code-Venture* for a specific class. After logging in, the teacher can select any player to view his/her performance details on the different games (Fig. 8.12) as well as send a private message to the player as shown in Fig. 8.13.


Welcome Dr John

PLAYER DETAILS

Name: Chloe Ting

Age: 18

SCORING DETAILS

Adventure Mode

Code-rearrange: 3/5 Find error: 1/5
Find output: 2/5 Fill blanks: 0/5

Challenge Mode

Variable Runner: 40

Algorithm Puzzle: 66

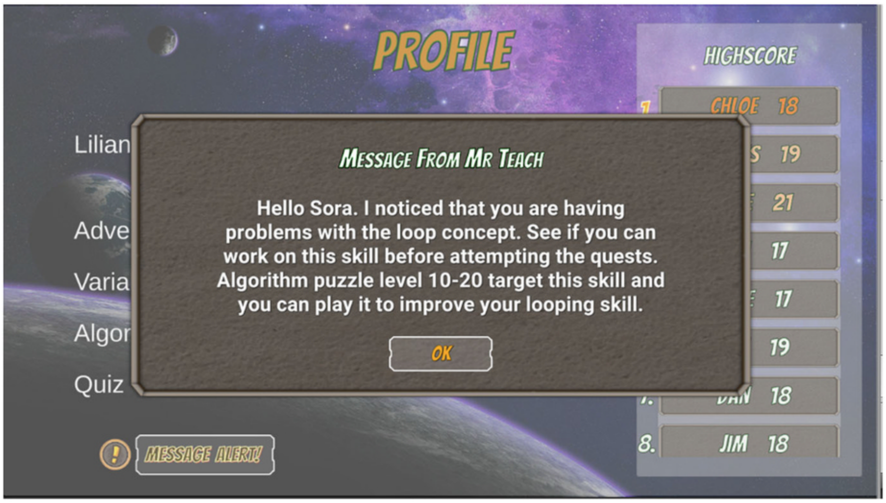
Quiz

Score: 9/10

Overall Performance

68%

Fig. 8.12 Player performance details



MESSAGE FROM MR TEACH

Hello Sora. I noticed that you are having problems with the loop concept. See if you can work on this skill before attempting the quests. Algorithm puzzle level 10-20 target this skill and you can play it to improve your looping skill.

Fig. 8.13 Personalized message sent to student

8.5.2 User Acceptance Testing

A User Acceptance Testing (UAT) was the last phase of the testing process that was performed by end users. This was to validate the software among the targeted audience, to get their precious feedback, and ensure the application met its intended purpose. Pre-game and post-game surveys have been conducted with 35 students who have very little to no perception about programming and were about to embark on undergraduate studies. After inquiring about certain basic information through an online form, they were made to play all games in *Code-Venture* for a period of 1 week. A feedback of their gaming experience was then taken by the means of a second survey.

8.5.2.1 Pre-game Results

The 35 students who tested the application were aged between 16 and 21 years inclusive, 57.1% were female, and the remaining 42.9% were male. Among others, the participants were asked on their background knowledge about programming, and the results, illustrated in Fig. 8.14, showed that 42.9% of them had only a vague notion of programming and 45.7% had none.

They were also asked about their feelings on programming, and the results are shown in Fig. 8.15. 68.6% of the participants thought that programming was hard, 22.9% were of the opinion that programming is medium difficult, and the remaining 8.6% thought programming was easy.

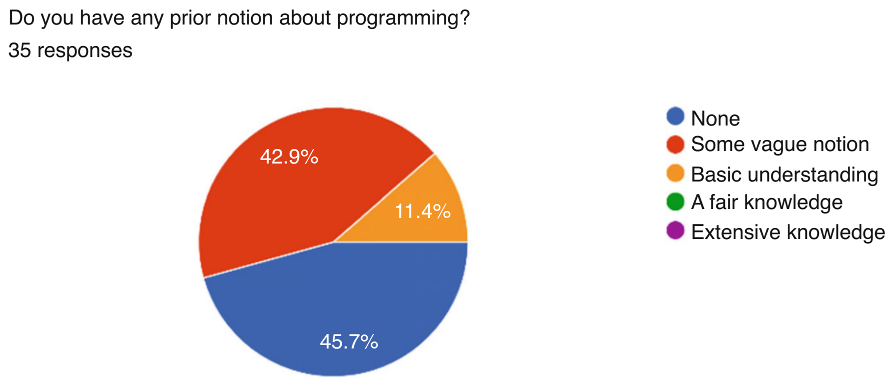


Fig. 8.14 Prior knowledge of programming

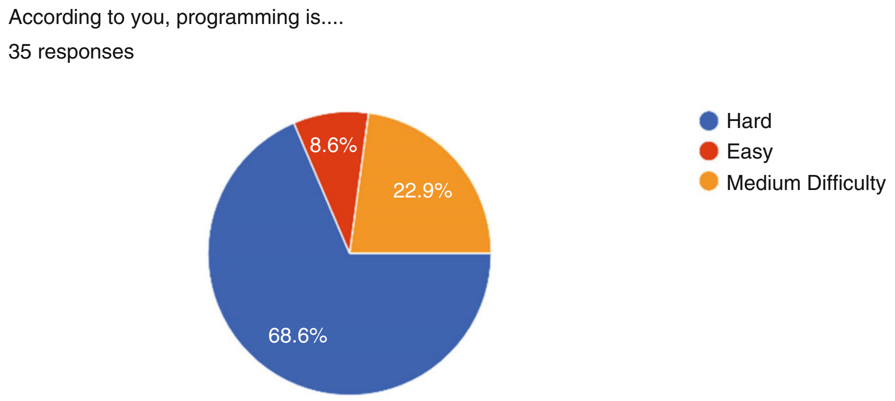


Fig. 8.15 Programming difficulty

The respondents were also asked about intention of pursuing further studies in programming, and the results are shown in Fig. 8.16. 65.7% of the participants mentioned that they did not intend to pursue their studies in programming, 11.4% intended to embrace the field, and the remaining 22.9% were unsure.

8.5.2.2 Post-game Results

The 35 students were given *Code-Venture* to play for 1 week and were required to answer a post-game survey. As illustrated in Fig. 8.17, 42.9% of the participants liked the game very much, while 31.4% liked the game. Twenty percent were neutral about the game, while 5.7% did not like *Code-Venture*. When asked whether they wanted to play *Code-Venture* more in the future, 91.4% of the respondents answered favorably.

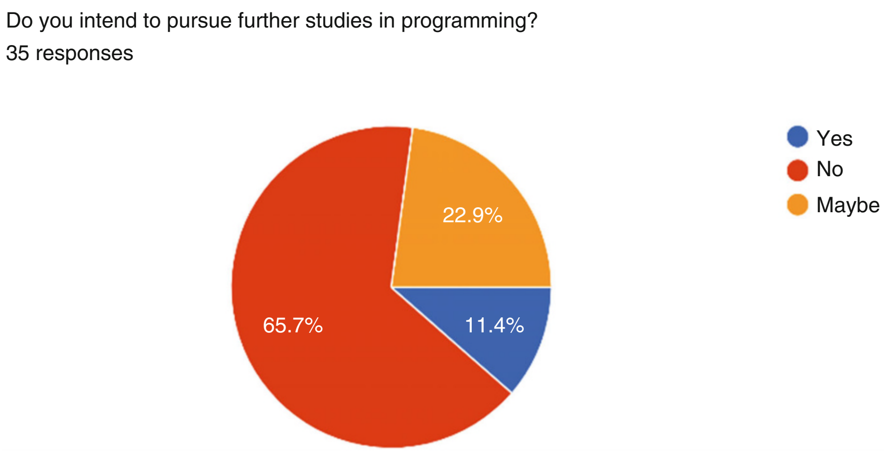


Fig. 8.16 Further studies in programming

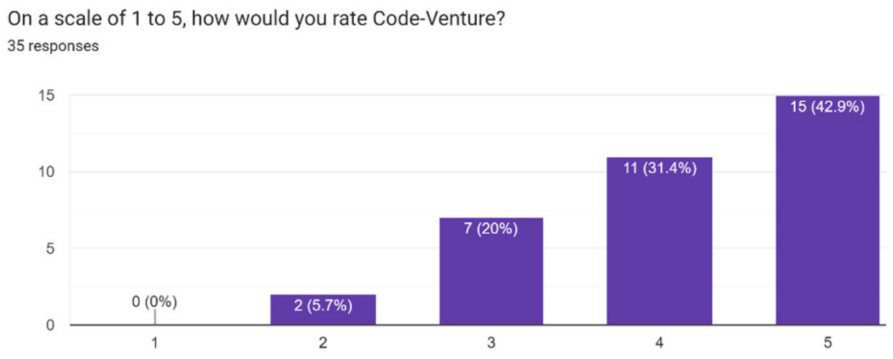


Fig. 8.17 *Code-Venture* ratings

The participants were also asked to rate *Code-Venture* as a means of teaching programming. As shown in Fig. 8.18, 51.4% found the game to be useful, and 3.14% enjoyed the game and received some information on programming. 11.4% were not able to decide whether the game was useful in helping them better understand programming, while 5.7% found the game to be minimally useful. It is also worth noting that none of the respondents indicated that the game was not useful at all in helping them in programming. When prompted about game-based learning, 77.1% of the respondents were of the opinion that game-based learning is a good approach to learn programing, 20% were unsure, and one participant mentioned that it is not suitable.

After playing *Code-Venture*, the participants were again asked about their feelings on programming. As shown in Fig. 8.19, this time, 40% of the respondents found it to be easy, compared to the 8.6% obtained prior to playing *Code-Venture*. Only 5.7% found programming to be hard post *Code-Venture*, while previously a

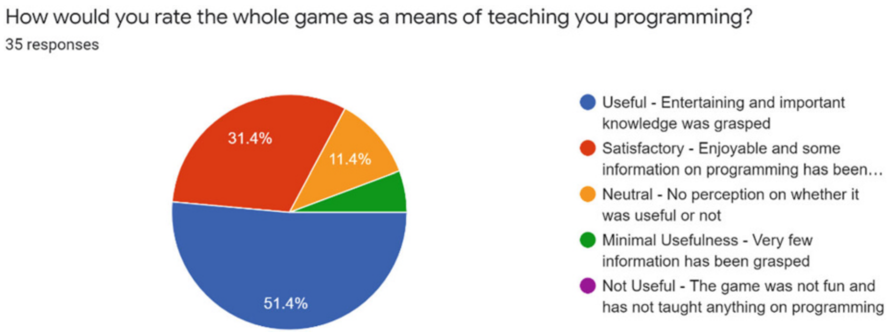


Fig. 8.18 Usefulness of *Code-Venture*

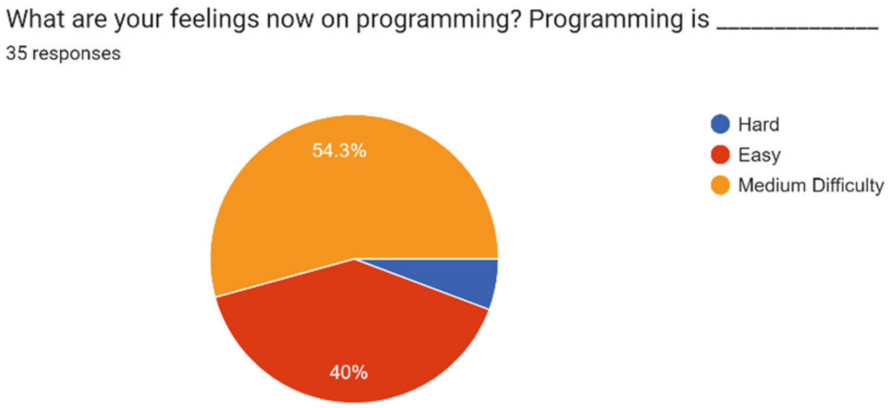


Fig. 8.19 Feelings on programming after playing *Code-Venture*

huge proportion of 68.6% found it to be difficult. The percentage of students who found programming to be slightly difficult rose from 22.9% to 54.3% with many respondents moving their feelings from *Hard* to *Medium difficult*.

When the participants were again asked about their intention to pursue further studies in programming, 32.4% responded positively, compared to the 11.4% previously. Only 5.8% (two participants) mentioned that they did not want to have their further studies in programming compared to a whopping 65.7% previously. The percentage of students who were unsure about their further studies in programming rose from 22.9% to 61.8% with many respondents moving their opinions from *No* to *May be* (Fig. 8.20).

8.5.2.3 Overall Feedback from Students

The following feedback were compiled from the post-game surveys.

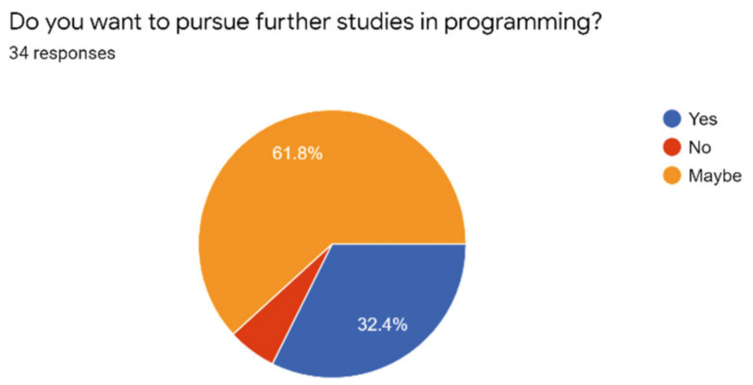


Fig. 8.20 Further studies on programming after playing *Code-Venture*

- Adventure Game—Positive

The game proved to bring a sense of motivation and eagerness to learn. The element of game exploration and quests further enhanced the player's determination to progress through the game and complete all the achievements. By doing so, the students had to go through programming concepts and rules that made them more familiar with the world of programming.

- Variable Runner—Satisfactory

This game proved to be informative and did have a good response from players. However, the players expressed that it was too basic and systematic. The concepts were only being introduced without proper explanation on how these concepts actually work. Hence, comprehension of variables and data types was only partially achieved.

- Algorithm Puzzle—Positive

This game proved to be useful to most of the participants, as they understood the game easily and managed to complete several levels with a good score. Upon questioning, they said they could understand the concept of commands and sequencing and were able to tackle these puzzles easily. The skills targeting algorithmic thinking, problem-solving, debugging and functions, and loops and sequencing proved to have been successfully inculcated in the students.

- Quiz Game—Neutral

This game ended up having a neutral effect on the students since they did not really have much knowledge about programming initially. Hence, answering a set of questions regarding this topic proved to be tough for them. As a result, a majority of the students scored low marks for this game. Thus, it has been deduced that this game mode would be much more efficient if played after attempting the other games or after being exposed to some more programming principles.

8.6 Discussion

Code-Venture has several strengths as it has been designed based on the ACM/IEEE guidelines for introductory programming. The game consists of several mini-games that target different skills required for learning programming. Each mini-game differs from the other, making the gameplay experience fun, engaging, and appealing. The game has an adventure mode where the player is able to venture out and attempt programming puzzles while exploring a beautiful interactive 3D world and making the gameplay experience interesting with in-game characters to interact with. The player gets the feeling of actually playing a real game while indirectly learning about programming. The player gets even more motivated to excel in the game due to the high score/leaderboard system that triggers the element of competition among the players. The system has a mobile application, which is used by the teacher to monitor the scores and overall performance of the players while they attempt the different games. He can hence easily determine the areas of weaknesses of the students, and thus, targeted assistance can be provided to all the players based on their gaming and learning experience. The game elements used in *Code-Venture* include levels, points, leader boards, avatars, and quests. It also uses the following game mechanics: health, energy, coins, time, position, attack, interact, movement, and opening chests.

The pilot testing carried out with 35 participants resulted in some promising outcomes. After playing *Code-Venture*, 40% of the respondents found programming to be easy, compared to the 8.6% obtained prior to playing the game. Only 5.7% found programming to be hard after playing *Code-Venture*, while initially a remarkable 68.6% found it to be difficult. When asked about their intention to pursue further studies in programming, 32.4% of the respondents replied positively after playing *Code-Venture* compared to the 11.4% previously. A mere 5.8% mentioned that they did not want to have their further studies in programming compared to a substantial 65.7% previously. It can therefore be deduced after this pilot study that the mobile serious programming game *Code-Venture* does have a positive impact on the players. These findings may also have practical implications for the tutor delivering the programming module. Game-based learning strategies could be adopted, and the use of serious programming games can be included in the teaching and delivery of the introductory programming modules. Assessments or non-curricular activities may also be designed on the use of the serious games in normal classes or during practical sessions. Obviously, a more thorough testing of the application on a larger scale and for a longer duration is required to have better statistical claims about the effectiveness of the application.

While *Code-Venture* has several strengths, it does have some weaknesses. Internet connection is required for connecting to an online database to store the score of the player, which will be viewed by a teacher. *Code-Venture* has currently been developed for Android platforms only. Moreover, mobile devices older than Android 4.1 are not able to run *Code-Venture* due to incompatibility issues with the new components included in the game. The game takes some storage space, about 130 MB, on the mobile device it has been installed due to it being a 3D game consisting of heavy game objects. As for any other 3D game, *Code-Venture* can also consume a high amount of battery life when played over a prolonged period.

Some avenues for further improvement can be considered in the future. Artificial intelligence can be integrated in the mini-games to have more responsive and adaptive gaming experiences. Customized progress details and graphs concerning the student's strengths and weaknesses in the teacher's application may also be enhanced. Multiplayer game modes, whereby the players are able to challenge and compete with others for rewards and points, will definitely be a big advantage. Additionally, more quests in the game's adventure mode, an improved storyline, and expanding the game environment to increase the areas of exploration can be envisaged. Finally, the development of more mini-games in the challenge mode to tackle more programming skills may also be useful.

Based on our analysis, *Code Combat*, *Hour of Code*, *LightBot*, and *May's Journey* are the most featured currently available serious programming games. Together with *Code-Venture*, they have all been designed to encourage computational thinking among individuals of all ages and assist them in learning fundamental computer science principles. All these games have a good combination of text, audio, and graphics. What really differentiates *Code-Venture* from the other serious games is that *Code-Venture* makes use of a varying gameplay for the different integrated mini-games while the other games have a similar gameplay for all the different levels. *Code-Venture* also includes a mobile application for the tutor who can visualize the progress of the students on the different games. What makes *Code-Venture* unique is the possibility of sending personalized messages to students to advise them on their progress.

There are several threats to validity that could have an impact on the results obtained. Firstly, the sample size was very small with only 35 participants and limited to the students available from the researchers' contacts. The use of a convenience sampling poses a threat to internal validity. Moreover, some students have parental or siblings support at home, which may have affected their interaction with *Code-Venture*. Additionally, since the ages of the participants were between 16 and 21, the maturity of the respondents may also affect the results. Finally, the students were given the mobile application for a period of only 1 week. The time that the students interacted with the application may therefore not be the same. A greater sample size would have helped minimize these issues.

8.7 Conclusion

This work investigated the possibility of applying a game-based learning strategy to teach novice students about programming through a serious game named *Code-Venture*. *Code-Venture* was based on the ACM/IEEE Computing Curriculum for programming. The main objective of *Code-Venture* was to enlighten the students about this seemingly complex programming subject and show them that it could be fun and enjoyable to learn the programming principles. *Code-Venture* makes use of varying gameplays to ensure the player is engaged and motivated. While the mini-games serve as a challenge for the player, the adventure-mode creates an immersive experience for learning. This game is associated with an application that helps teachers monitor their students' performance. With the elaborate scoring system, the student's skills, strengths, and weaknesses can be evaluated.

A pilot study has been carried out with 35 students together with pre-game and post-game surveys. The survey results were analyzed, and the outcomes were very promising. The participants who tested *Code-Venture* were very entertained and engaged and showed keen interest in playing more. They got a better perception about programming and expressed their will to learn more about it. A positive change was noted in the opinion of the students regarding programming. After playing *Code-Venture*, 40% of the respondents found programming to be easy compared to the 8.6% obtained prior to playing the game. Only 5.7% found programming to be hard post-*Code-Venture*, while previously an impressive 68.6% found it to be difficult. When the participants were asked about their intention to pursue further studies in programming, 32.4% responded positively post playing *Code-Venture* compared to the 11.4% previously. Only 5.8% mentioned that they did not want to have their further studies in programming compared to a massive 65.7% previously.

Code-Venture is a promising game providing ease of access, viability, and the opportunity to sharpen one's knowledge and expand one's understanding of programming in a fun and entertaining way. While *Code-Venture* has several benefits, the game is still lacking in some areas. A more elaborated and captivating storyline has yet to be implemented in the *Adventure-Mode* to better grasp the player's attention with more interesting quests and dialogues. The mini-games could be improved by including a better tutorial system to guide the player. The teacher's mobile application has few options to view progress of the students that can also be further enhanced. Moreover, budget limitations resulted in only free assets and resources being considered for the development *Code-Venture*. The lockdown due to the COVID-19 pandemic resulted in a pilot study where the testing was carried out by only 35 participants. Hence, there is a need to carry out a thorough testing of *Code-Venture* over a long period to evaluate the impact of the application on the skills of the players and confirm its effectiveness in helping the players better grasp the main programming principles.

Disclosures No funding was received for this article, and the authors declare no conflicts of interest.

References

1. Papadakis, S., Kalogiannakis, M., Orfanakis, V., Zaranis, N.: Novice programming environments. Scratch & app inventor: a first comparison. In: Proceedings of the 2014 Workshop on Interaction Design in Educational Environments, pp. 1–7 (2014)

2. Miskon, M.T., Hilmi, F.D., Khusairi, W.A., Rustam, I.: Development of constructionist robotics to facilitate learning in C programming course. J. Phys. Conf. Ser. **1529**(2), 022039 (2020)
3. Mathew, R., Malik, S.I., Tawafak, R.M.: Teaching problem solving skills using an educational game in a computer programming course. Inf. Educ. **18**(2), 359–373 (2019)
4. Lahtinen, E., Ala-Mutka, K., Järvinen, H.M.: A study of the difficulties of novice programmers. ACM SIGCSE Bull. **37**(3), 14–18 (2005)
5. Bennedsen, J., Caspersen, M.E.: Failure rates in introductory programming. ACM SIGCSE Bull. **39**(2), 32–36 (2007)
6. Bennedsen, J., Caspersen, M.E.: Failure rates in introductory programming: 12 years later. ACM Inroads. **10**(2), 30–36 (2019)
7. Lye, S.Y., Koh, J.H.L.: Review on teaching and learning of computational thinking through programming: what is next for K-12? Comput. Hum. Behav. **41**, 51–61 (2014)
8. Halbrook, Y.J., O'Donnell, A.T., Msetfi, R.M.: When and how video games can be good: a review of the positive effects of video games on well-being. Perspect. Psychol. Sci. **14**(6), 1096–1104 (2019)
9. Schez-Sobrino, S., Vallejo, D., Glez-Morcillo, C., Redondo, M.Á., Castro-Schez, J.J.: RoboTIC: a serious game based on augmented reality for learning programming. Multimed. Tools Appl. **79**, 34079–34099 (2020)
10. Shahid, M., Wajid, A., Haq, K.U., Saleem, I., Shujja, A.H.: A review of gamification for learning programming fundamental. In: 2019 International Conference on Innovative Computing (ICIC), pp. 1–8. IEEE (2019)
11. Mathrani, A., Christian, S., Ponder-Sutton, A.: PlayIT: game based learning approach for teaching programming concepts. J. Educ. Technol. Soc. **19**(2), 5–17 (2016)
12. Boeker, M., Andel, P., Vach, W., Frankenschmidt, A.: Game-based e-learning is more effective than a conventional instructional method: a randomized controlled trial with third-year medical students. PLoS One. **8**(12) (2013)
13. Ding, D., Guan, C., Yu, Y.: Game-based learning in tertiary education: a new learning experience for the generation Z. Int. J. Inf. Educ. Technol. **7**(2), 148 (2017)
14. Cheng, M.T., Chen, J.H., Chu, S.J., Chen, S.Y.: The use of serious games in science education: a review of selected empirical research from 2002 to 2013. J. Comput. Educ. **2**, 353–375 (2015)
15. Krath, J., Schürmann, L., Von Korflesch, H.F.: Revealing the theoretical basis of gamification: a systematic review and analysis of theory in research on gamification, serious games and game-based learning. Comput. Hum. Behav. **125**, 106963 (2021)
16. Zainuddin, Z., Chu, S.K.W., Shujahat, M., Perera, C.J.: The impact of gamification on learning and instruction: a systematic review of empirical evidence. Educ. Res. Rev. **30**, 100326 (2020)
17. Qian, M., Clark, K.R.: Game-based learning and 21st century skills: a review of recent research. Comput. Hum. Behav. **63**, 50–58 (2016)
18. Tori, A.A., Tori, R., Nunes, F.L.: Serious Game Design in Health Education: A Systematic Review. IEEE Transactions on Learning Technologies (2022)
19. Jemmali, C., Yang, Z.: May's journey: a serious game to teach middle and high school girls programming. Master's thesis, Worcester Polytechnic Institute (2016)
20. Du, J., Wimmer, H., Rada, R.: "Hour of Code": can it change students' attitudes toward programming? J. Inf. Technol. Educ. Innov. Pract. **15**, 53 (2016)
21. Miljanovic, M.A., Bradbury, J.S.: Robot on! A serious game for improving programming comprehension. In: Proceedings of the 5th International Workshop on Games and Software Engineering, pp. 33–36 (2016)
22. Law, R.: Teaching programming using computer games: a program language agnostic approach. In: European Conference on Games Based Learning, pp. 368–376. Academic Conferences International Limited (2017)
23. Junaeti, E., Sutarno, H., Nurmalasari, R.R.: Genius learning strategy of basic programming in an adventure game. In: IOP Conference Series: Materials Science and Engineering, vol. 288, No. 1, p. 012057. IOP Publishing (2018)

24. Jordaan, D.B.: Board games in the computer science class to improve students' knowledge of the python programming language. In: 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC), pp. 1–5. IEEE (2018)
25. Lotfi, E., Mohammed, B.: Teaching object oriented programming concepts through a mobile serious game. In: Proceedings of the 3rd International Conference on Smart City Applications, p. 74. ACM (2018)
26. Yallihep, M., Kutlu, B.: Mobile serious games: effects on students' understanding of programming concepts and attitudes towards information technology. Educ. Inf. Technol., 1–18 (2019)
27. Zhao, D., Muntean, C., Muntean, G.: The Restaurant Game: a NEWTON PROJECT serious game for C programming courses. In: Society for Information Technology & Teacher Education International Conference, pp. 1867–1874. Association for the Advancement of Computing in Education (AACE) (2019)
28. Karram, O.: The role of computer games in teaching object-oriented programming in high schools-code combat as a game approach. WSEAS Trans. Adv. Eng. Educ. **18**, 37–46 (2021)
29. Toukiloglou, P., Xinogalos, S.: NanoDoc: designing an adaptive serious game for programming with working examples support. Eur. Conf. Games Based Learn. **16**(1), 628–636 (2022)
30. Akkaya, A., Akpinar, Y.: Experiential serious-game design for development of knowledge of object-oriented programming and computational thinking skills. Comput. Sci. Educ. **32**(4), 476–501 (2022)
31. Hacked. Google Play. [online]. https://play.google.com/store/apps/details?id=com.hackedapp&hl=en (2015). Accessed 2 Oct 2019
32. Coding Galaxy. App Store. [online]. https://apps.apple.com/us/app/coding-galaxy/id1240651393 (2022). Accessed 20 Feb 2023
33. Lightbot: Code Hour. App Store. [online]. https://apps.apple.com/us/app/lightbot-code-hour/id873943739 (2018). Accessed 20 Feb 2023
34. SpriteBox: Code Hour. App Store. [online]. https://apps.apple.com/us/app/spritebox-code-hour/id1161515477 (2018). Accessed 20 Feb 2023
35. Meoweb: The Puzzle Coding Game. Google Play. [online]. https://play.google.com/store/apps/details?id=br.com.tapps.meoweb&hl=en&gl=US (2020). Accessed 20 Feb 2023
36. BeBlocky: Kids Code Easy. Google Play. [online]. https://play.google.com/store/apps/details?id=com.beblocky.beblocky&hl=en&gl=US (2022). Accessed 20 Feb 2023
37. Coding Planets. Google Play. [online]. https://play.google.com/store/apps/details?id=com.material.design.codingplanet&hl=en&gl=US (2017). Accessed 10 Dec 2022
38. Grasshopper: Learn to Code. Google Play. [online]. https://play.google.com/store/apps/details?id=com.area120.grasshopper&hl=en&gl=US (2023). Accessed 20 Feb 2023
39. Gomes, A., Mendes, A.J.: Problem solving in programming. In: PPIG, p. 18. (2007)
40. Ahmadzadeh, M., Elliman, D., Higgins, C.: The impact of improving debugging skill on programming ability. Innov. Teach. Learn. Inf. Comput. Sci. **6**(4), 72–87 (2007)
41. Fucci, D., Turhan, B., Oivo, M.: On the effects of programming and testing skills on external quality and productivity in a test-driven development context. In: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, pp. 1–6 (2015)
42. Csernoch, M., Biró, P., Máth, J., Abari, K.: Testing algorithmic skills in traditional and non-traditional programming environments. Inf. Educ. **14**(2), 175–197 (2015)
43. Futschek, G.: Algorithmic thinking: the key for understanding computer science. In: International Conference on Informatics in Secondary Schools-Evolution and Perspectives, pp. 159–168. Springer, Berlin (2006)
44. Farrell, J.: Programming Logic and Design, Comprehensive. Cengage Learning (2014)
45. ACM/IEEE-CS Joint Task Force on Computing Curricula: Computer Science Curricula 2013. Technical Report. ACM Press and IEEE Computer Society Press (2013)

Chapter 9
Using Active Learning to Teach Software Engineering in Game Design Courses

Bruce R. Maxim and Jeffrey J. Yackley

Abstract Game developers are beginning to understand it is important to approach computer game design like how all software engineers approach projects involving large numbers of people and significant investment of time. Engineering instructors often rely on the traditional lecture model when they teach topics to a classroom of students. Students often fail to engage with the material presented by lecturers. Many engineering educators regard experiential learning as an effective way to train future generations of engineers and game developers. The authors have created two courses that focus on software engineering and game development. These courses were initially offered as traditional lecture classes to both in-person and online groups of students. This chapter describes the authors' approaches to revising these game design classes to make use of flipped classroom models that rely on active learning, role-play, and gamification to cover software engineering topics in these courses. Students learn to use Agile software engineering practices to design, implement, and test game prototypes. In-person students were surveyed to measure their perceived levels of engagement with course activities. Our assessment data suggests that students attending flipped class meetings were slightly more engaged with the course materials than those taking the class offered using lectures only. Students interacting with the active learning course materials felt better able to apply their knowledge than students in a traditional lecture course.

Keywords Active learning · Student engagement · Role-play · Game design · Agile development

B. R. Maxim (✉)
University of Michigan-Dearborn, Dearborn, MI, USA
e-mail: bmaxim@umich.edu

J. J. Yackley
University of Michigan-Flint, Flint, MI, USA
e-mail: jyackley@umich.edu

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_9

9.1 Introduction

Engineering instructors often rely on the traditional lecture model where they cover a topic, with or without a slideshow, to a classroom of students. Students often fail to engage with the material presented by lectures until an assessment activity is near. Many engineering educators regard experiential learning as the most effective way to train future generations of engineers and game developers. The authors have noticed higher levels of engagement when students participate in class activities rather than passively listening to lectures. These activities may include games, discussions, role-play, peer reviews, and group problem-solving or design exercises. This chapter describes the authors' approach to revising two lecture heavy game design courses to make use of a flipped classroom model that relies on active learning, role-play, and gamification to cover software engineering topics in game design courses.

Covid-19 restrictions forced a shift to the online delivery of all courses at our university in 2020. In Fall 2021, face-to-face class meetings were allowed if vaccination, masking, and social distancing were enforced. Often, activities developed for face-to-face delivery of software engineering topics cannot be used without modification in the online delivery of course materials. Following Covid protocols in face-to-face classes also required modification of active learning course materials.

Students learning software engineering principles and practices may find it difficult to apply them in the development of complex software projects. Software engineering involves acquiring application domain knowledge to understand the client's needs. It is therefore important to do more than simply use a game as the term project in a software engineering course as some authors have suggested [1–3]. Additionally, adding game topics to already crowded software engineering courses, as some authors have advised [3, 4], requires sacrificing important software engineering topics. Focusing on one application area in the first software engineering class is not fair to all students as not every software engineering student wants to become a game developer.

9.2 Background

Game developers are beginning to understand that it is important to treat computer game design in the same way that other software engineers approach projects involving large numbers of people and a significant investment of time [5]. Game developers can benefit from using evolutionary software process models to manage their development risks and reduce their project completion times. The process of determining the technical requirements for a game software product is like that used to specify any other type of software product. However, unlike most software products, games have an entertainment dimension. People play computer games because games are fun [6].

The authors believe that the capstone design course should not be the only opportunity for students to manage complex software development projects. This suggests the use of other courses in the curriculum such as a game design course as a means of providing additional software engineering experiences. This paper describes the authors' experiences revising and employing active learning materials to teach software engineering content in a sequence of two face-to-face game design courses with or without social distancing and online either synchronously or asynchronously spanning a 6-year period.

9.2.1 Active Learning

Engineering educators regard experiential learning as the best way to train the next generation of engineers [7]. Toward this end, it is reasonable to believe that the interaction practiced in active learning classrooms can improve software engineering education at the undergraduate level and better prepare students for the experiential learning that comes with their capstone projects [8].

Active learning is "embodied in a learning environment where the teachers and students are actively engaged with the content through discussions, problem-solving, critical thinking, debate and a host of other activities that promote interaction among learners, instructors and the material" [9]. Prince defines active learning as any classroom activity that requires students to do something other than listen and take notes [10]. Active learning opportunities can complement or replace lectures to make class delivery more interesting to the students. Active learning using flipped classes can also foster developing an attitude of lifelong learning among students [11].

Specifically, active learning helps students develop problem-solving, critical reasoning [12], and analytical skills, all of which are valuable tools that prepare students to make better decisions, become better students, and better employees [10]. Raju and Sankar undertook a study to develop teaching methodologies that could bring real-world issues into engineering classrooms [13]. The results of their research led to recommendations to engineering educators on the importance of developing interdisciplinary technical case studies that facilitate the communication of engineering innovations to students in the classroom.

Active learning helps students learn by increasing their engagement in the process [14, 15]. Active learning techniques help students better understand the topics covered in the curriculum [16]. Active learning also helps students be more excited about the study of engineering than traditional instruction [17]. The group work that often accompanies active learning instruction helps students develop their soft skills [18] and makes students more willing to meet with instructors outside of class [19]. Krause writes that engagement does not guarantee learning is taking place, but learning can be enhanced if it provides students with opportunities to reflect on their learning activities [20].

There is consensus among members of our department's professional advisory board that professional practice invariably requires strong verbal and written communication skills. To develop their oral communications skills, students need opportunities to present their work as well as observe their peers doing the same. Some instructors believe that the project activities inherent in real-world software development encourage students to improve their written and oral communication skills [21].

Day and Foley used class time exclusively for exercises by having their students prepare themselves through the study of materials provided online [22]. Bishop and Verleger presented a comprehensive survey of flipped classroom exercise implementations [23]. Wu et al. effectively implemented class exercises as active learning tools in their flipped classroom approach [24]. Research suggests that the success of flipped classroom approaches depends on the nature of the course being taught. Learning content after engaging in course activities can be easier for some students [25]. The investment in time required for instructors to develop quality out-of-class materials and in-class active learning experiences can be substantial [26].

The active learning approach of problem-based learning (PBL) has consistently been demonstrated to lead to positive learning outcomes such as self-directed learning habits, problem-solving skills, and deep disciplinary knowledge while engaging students in collaborative, authentic, and learning situations [27]. While PBL was first incorporated into medical school curricula in 1969, it is currently used in a wide variety of courses [28]. For instance, within the field of engineering, Warnock and Mohammadi-Aragh investigated the impact of PBL on student learning in a biomedical materials course and found that students made significant improvements in their problem-solving, communication, and teamwork skills [29].

PBL has also been used in senior-level engineering courses with the same positive results [30–32]. Although students in a PBL software engineering course reported that the projects were more time intensive than a typical course project, they were receptive to the approach since they thought it was related to the professional environment and provided them with opportunities to relate theory and practice. This contrasted with students taught using a traditional lecture and project approach to the course who viewed completing a traditional course project more negatively [33].

9.2.2 Student Engagement

Active learning techniques such as think-pair-share exercises [34], pair programming [35], peer instruction [36], and flipped classrooms [37] have been demonstrated to increase student engagement [11]. Many of these interventions are used in introductory-level instruction, primarily to address broadening participation in large classes [38]. Admittedly, lack of access to technology to create and access the videos needed to flip a classroom can pose challenges to both students and teachers [26].

Ham and Myers introduced process-oriented guided inquiry learning (POGIL) into a computer organization course [39]. In software engineering courses, the use of real-world, community-based projects may be an effective way to engage students with a meaningful problem while teaching them software engineering concepts [40]. Students often become more invested in their projects when they see that their products are more than simply a paper design. In our course redesign, we used the class activities to motivate students to design game software products and use software engineering techniques to solve real-world programming problems.

An important aspect of software engineering education is the development of soft skills such as communication and project management. There are several examples of courses that make use of project work to help students enhance their soft skills simultaneously with their software development skills [41]. Decker and Simkins [42] introduced the use of an extended role-play approach in a game development process class where the students were not assessed solely on the artifacts they produced but the processes by which they created their artifacts. Their role-play activities emphasize industry best practices for both technical and soft skills (project management, communication, marketing, and interdisciplinary design).

9.2.3 *Role-Play*

Simkins [43] defines role-play as simulating the real world in environments where consequences can be mitigated safely. Role-play allows students to get hands-on practice with engineering concepts and practice the soft skills that make for successful professional engineers: communication, problem-solving, and analytical skills. We believe this makes role-play a critical tool in the active learning engineering classroom. Numerous researchers have investigated the use of role-play in the software engineering classroom with success.

Moroz-Lapin [44] and Seland [45] used role-play in human-computer interaction courses to engage students with the requirement engineering process to better understand system behavior from the users' point of view. Similarly, Zowghi and Parvani [46] also investigated requirements engineering using role-play to have their students understand the process of requirements gathering from both the client and developer perspective. Role-play was used by Börstler [47] to teach students object-oriented programming concepts with class-responsibility-collaborator cards. Vold and Yayilgan [48] achieved greater student engagement with role-play in an information technology course. Further, we draw inspiration from a study that used the Second Life online virtual world as a platform for students to role-play a fictional company for enterprise resource planning [49]. Other online role-play simulations focus on students taking the role of project managers with students receiving immediate feedback on their decisions [50–52].

The redesign described in this paper builds upon the work of Maxim, Brunvand, and Decker [53], which used role-play in a re-designed game design course, CIS 488, at the University of Michigan–Dearborn. We re-use this work with some slight modifications as the second course in our two-course game design sequence [54]. The course from 2017 had the students role-play as developers of a failing game company with the goal of simulating concept ideation to creation and release of 3D computer games using Unreal Engine 4. The failing game company backstory used to motivate the role-play in our course is discussed further in Decker and Simkins [42]. Decker and Simkins provide the framework we used to build and adapt our role-play modules. These modules emphasize industry best practices for the technical game development work and soft skills development as well as the introduction of secondary learning objectives based in business and legal concerns that naturally arise during the role-play [54].

9.2.4 Gamification

Gamified learning or the gamification of learning has been defined as the use of game design elements in non-game settings to increase motivation and attention on task [55, 56]. Using active learning in the authors' experience may lead to issues with group participation and motivation if students do not feel the need to work outside of class. Adding gamification elements to active learning can help mitigate this problem.

James Gee [57] has identified 36 learning principles that are present in good games. These learning principles provide the backbone for good game design and, in turn, can be used as guiding principles when designing a gamified learning environment. For instance, good games provide players with information when they need it and within the context in which the information will be used [58]. Effective game design includes challenging players, so they are routinely working at the edge of their abilities and knowledge, also known as their zone of proximal development [59]. Having students, or players, operate within this optimal learning zone helps keep them engaged and encourages them to learn more to meet the demands of the next challenge.

According to Gee [58], games can promote collaboration and skill building, if players are required to share knowledge and skills to be successful. Games that reward teamwork can have a positive impact on the development of prosocial skills [60]. Gee contends that well-designed games are motivational specifically because of the different learning principles outlined previously [58]. Working at the limits of their abilities keeps players engaged as they continue to take on new challenges [61]. Gee refers to this process as a cycle of expertise, which requires players to constantly learn, act, revise, and learn again to demonstrate proficiency and be successful in a game [57].

In addition to the motivational aspect of the cognitive element of games, Lee and Hammer [62] suggest that the social and emotional aspects of rewards and consequences earned in gaming environments contribute to motivation as well. However, there needs to be a balance between positive and negative outcomes to prevent discouraging or overwhelming the students [56]. A well-designed game can also motivate players to stay engaged by enhancing the value of the task or tasks being completed [63]. This is particularly beneficial with educational games focused on school-related subjects that students might not otherwise choose to immerse themselves within. Toth and Kayler [64] created a role-playing game that made use of quests to motivate students' assignment completion.

Gamification can be used as a means of promoting rewards for completing tasks. Students can be rewarded for compliance to software process steps and for taking the initiative to improve their "soft skills."

It is important to acknowledge the debate that centers around gamification. There are critics such as Ian Bogost who colorfully proclaim "Gamification is bullshit" and that it is little more than a marketing term for exploitative practices [65]. A more nuanced criticism from Casey O'Donnell argues that gamification at its heart is a form of algorithmic surveillance that provides data of dubious merit and use [66].

9.3 Proposed Solution

The University of Michigan–Dearborn offers a two-course undergraduate sequence, CIS 487 and CIS 488, in game design. These courses are offered in-person on campus and paired with an online section that allows enrolled students to complete the course requirements asynchronously. Prior to 2017, these involved students attending or observing a 3-h lecture with slides. Little in-class interaction between students was observed with in-person course delivery. In our experience following students throughout the two-semester sequence, most students spent their class time with their laptops more than with the course lecture material [17]. We wanted to change the structure of these courses to better engage the students with the software engineering content covered in these courses. We describe our experiences in altering these courses to include active-learning role-play.

Given all the positive evidence discussed previously, it was determined that a PBL pedagogical approach was well suited for software engineering project courses. In our classes, students are encouraged to reflect on the lessons learned from the activities either in writing or orally during class postmortem discussions.

We included role-play activities in our course redesign to allow students to practice skills such as project management, communication, marketing, and interdisciplinary design. To encourage the development of soft skills, the investigators made use of small group activities with the expectation that students would provide written or oral summaries (either live online or using video) of the strategies used to complete their tasks and their lessons learned. The decision was made to continue to use the term-long role-play activities in CIS 488 since those students had a good grasp of software engineering and game design from the prerequisite courses CIS 487 and CIS 375 Software Engineering 1.

Gamification can be used as a means of promoting rewards for completing tasks. Students can be rewarded for compliance to software process steps and for taking the initiative to improve their "soft skills." In this way, the authors hope to resolve the discrepancies in personal efforts that are often present in student project work. We believe gamification can be accomplished in a non-manipulative and non-exploitative manner where the goal of the gamification is to provide different opportunities for involvement in the courses thereby allowing students to work on what interests them the most.

We designed tasks covering the gamut of game design and engineering process tasks and assigned them point values for successful completion. Students were allowed to negotiate their own tasks within their team structures while also being encouraged to work on a variety of different tasks to earn points toward their final course grade. These tasks encouraged development of soft skills through team communication, planning, and problem-solving. Allowing students to negotiate the nature of their activities and rewards upfront often goes a long way to ensuring that all students are engaged for the entire semester.

When Covid-19 forced us to eliminate or modify the way we offered our in-person game design courses, we developed strategies to improve online student engagement. In 2020 and Winter 2021, our game design classes were offered entirely online. Some students participated in these classes by attending synchronous class meetings using Zoom and completed small-group assignments in breakout rooms. Asynchronous online students watched online videos of class lectures and activities. Starting in Fall 2021, in-person instruction was allowed for students attending classes on campus, if they wore masks, were vaccinated, and followed social distancing rules while in the classroom or lab. Asynchronous online students continued observing class by viewing video recordings 1 day after the class meetings.

9.3.1 Course Overview: CIS 487 Computer Game Design I

The purpose of CIS 487 is to introduce students to the technology, science, and art involved in the creation of computer games. The course meets once a week for 3 h over a 15-week semester. Before the Fall 2017 semester, this course split time between lectures on game design principles and Unity 2D and 3D game engine video tutorials. The revisions to this course focused primarily on introducing active-learning activities on game design as an alternative to a lecture heavy focus for presenting course content. Table 9.1 shows a week-by-week listing of the topics for the course.

The weekly class was taught using a flipped classroom approach and was split into three principal components. The first component was a short interactive presentation on the game design material for the week. These presentations were reduced to 30–45 min on average and were then followed by the second component, an activity designed to engage the students more deeply with the material. Finally, the third component was a 30-min, tutorial video on a particular Unity engine tutorial on a particular topic usually related to the game design content for the week.

Table 9.1 A listing of the weekly topics and activities for CIS 487

Week	Software engineering topic	Activities
1	Game Design Evaluation Intellectual Property	Bartok Rule Changes Exercise Copyright Card Game
2	Game Storylines in Design Puzzle Design Process	Storyline Exercise Shocking Puzzle Design
3	Game Quality Review	Peer Review of Game Review
4	Game and Balance Storyboarding Feasibility Prototypes	Analysis of 3 Dot Game Paper Prototype—Test Feasibility of New First Person Shooter Game Design
5	Design Documents Brainstorming and Pitches Trade-off Analysis	Ideation and One Page Creation Create Game Pitch for One Page Game Analyze Impact of Adding or Removing Features Using Paper Prototypes
6	Formal Technical Reviews Playtesting	Peer Review 2D Pitch Document Playtest 2D Game Feasibility Prototype
7	User Experience Design Agile Development	Revise User Interface Design Process Improvement Game (PIG) Contest
8	UX Sound Design UX Level Design	Create Skit Using 2D Games Sounds Only Create Outline for New 2D Game level
9	2D Game Testing	Peer Review 2D Game Beta Prototype
10	Game AI design Game AI testing	Design New Finite State Game AI for 2D game Test Game AI Using Paper Prototype and Roleplay
11	Game Design Documents Formal Technical Reviews	Peer Review 3D Game Concept Presentations
12	Playtesting and Testing	Create Testing Script for 2D game External Testers use Script to Test 2D game
13	Playtesting	Playtesting of 3D Alpha Prototypes
14	Marketing	Marketing Exercise for 3D Game
15	Quality Assessment	Peer Assessment of 3D Beta Prototypes

The students were evaluated on the completion of five projects, four of which were team-based assignments and one which was an individual assignment. The group assignments involved the use of gamification to reward differential student project contributions that were broken down into elective components each with its own point value. Students could select any number of electives from the assignment to complete to earn a maximum number of points on the assignment. Students also submitted write-ups of the small-group activities completed in class. These write-ups were started in class, completed individually, and submitted for grading.

The first project was an individual review of a professionally produced computer game. Students prepared their reviews of the game and their critiques in a PowerPoint. They were then required to present them to the class. The reviews were to cover the basic information of the game (i.e., title, type, price, authors); a summary of the game, which was to include items such as the story, gameplay, user interface, etc.; and their thoughts on a number of questions such as the quality, fun, comparison to similar games, design mistakes, strengths, and weaknesses.

Projects two and three were completed by a group of three or four with the same students completing both projects together. Students selected their own partners for the projects. The two projects were the creation of a 2D Unity game pitch and the production of the game itself (delivered as two prototypes). The game pitches involved creation of a pitch document that outlined the game story, game play look and feel, and the development specifications. The 2D game required a playable game with at least one playable character, one level transition, and rudimentary physics and AI.

The fourth and fifth projects were also team-based, but the students were required to form teams of four or five individuals. The students again could choose their own partners but were not required to collaborate with the same partner from their 2D game. The fourth and fifth projects were to design and implement a 3D game alpha and beta prototype. The game requirements were like those for the 2D game with the expectation of a more polished and complete game.

9.3.2 Course Overview: CIS 488 Computer Game Design II

The CIS 488 course contains a semester-long role-play in which the students function as the employees of a struggling game company. Also, the course makes use of gamification and active-learning elements as did its predecessor, CIS 487. CIS 488 meets 1 day a week for 3 h over a 15-week semester. Table 9.2 shows the weekly topics and activities. During the first class period, students were introduced to the backstory of the role-play and how it would affect the conduct of the course. In previous offerings of this course, much of the class time was spent observing instructor lectures on Unreal4 programming techniques. In the current course offering, most class time was spent in game design studio role-play activities. Classes often began with an all-hands meeting to introduce the day’s role-playing activities. Students were expected to use video tutorials outside of class to learn to use the Unreal4 Blueprint system and level editor.

The fictitious company created for the role-play had a tradition of using a green light system for continuing or stopping development of game products. The first task was for each company developer to do a quick market research review and create a pitch for an innovative game product. The top five pitches were selected by class vote. The winning pitch authors were allowed to recruit four or five team members during the third class period. Each team was asked to provide a representative for a committee to write a company-wide software process standards document based on the scrum framework. A contest was held within the company to create a new name and logo. The developers selected their favorite, and Imagination Studio was launched.

Table 9.2 A listing of the weekly topics and activities for CIS 488

Week	Software engineering topic	Activities
1	Role-play Introduction	
2	3D Game Pitch Presentation	Peer Green Light Vote Team Formation
3	Software Process Definition	Teams Refine Game Concepts as One Pages Develop Agile Company Process Model
4	Business Plan Creation	Process Model Presentation and Approval One Page Review
5	Formal Technical Reviews	Peer Review of Draft Design Document
6	Elevator Pitches IP Ownership	Creation and Review of Game Elevator Pitch Game Theme Ownership Dispute Activity
7	Contracts and Scope Creep	Two Pitch Swaps Contract Dispute Activity Lens Presentations
8	Playtesting	Peer Review of Alpha Game Prototypes
9	Retrospective Game AI Design	Greenlight Vote on Alpha Prototypes Alpha Retrospective and Beta Planning Lens Presentations
10	Security	Game Espionage Activity Lens Presentations
11	Formal Technical Review Playtesting	Peer Review of Final Game Design Document Playtesting of Beta Game Prototype
12	Software Evolution	Create an Outline for a Game Sequel with Taking Game Asset Reuse into Consideration Lens Presentations
13	Game Packaging Marketing	Create the Script for the Team Game Project Lens Presentations
14	Marketing Presentations	Peer Review of Game Marketing Video
15	Quality Assessment	Peer Assessment of Gold Release Candidates

Each team's first task was to create a game design document and a business plan for their game. To assist them in this task, two local game company owners were recruited to act in the role of business consultants who shared their experiences with creating a company and bringing their first games to market. The second team deliverable was a game alpha prototype, which included one complete logic path and a draft user manual. This delivery signaled the end of the first sprint in the scrum framework. These games were evaluated for quality of game play. The company looked at the productivity of each team. The team leads were asked to make an oral presentation to confirm that they had sufficient resources to complete their game products on time (the end of the semester was designated as the end of the fiscal year). All developers discussed the future of the game products and decided (without the instructor's influence) to cancel one of the projects. The developers from the canceled project were reassigned to existing development teams.

The third team deliverable was a beta prototype, which needed to accommodate a requirement change. The change resulted in the addition of a significant game artificial intelligence (AI) element to their evolving design. This deliverable also included the creation of the final game design document and test plan. The final team deliverable was the gold release prototype and a marketing presentation that included a video piece to promote their game product. Company developers scored each game (other than their own) using a rubric provided by the instructor. The average of these scores was used as the grade for the prototype.

The students participated in several role-play scenarios through the semester, in addition to greenlighting the games. One element of this class that was hard to fit into the role-play framework was the assignment where each developer uses their own game to illustrate game design features from Schell's book of game design lenses [67]. In this assignment, each student selects a group of three related lenses and creates a 20-min presentation discussing how these lenses illustrate qualities from their game or not. This is sold as continuing education or inspiration for undertaking perfective maintenance activities to the company developers.

9.4 Results and Discussion

Each of the course assignments was evaluated using Canvas rubrics designed by the instructor for each type of submission. Currently, these rubrics contain 2–10 criteria, each scored from 1 to 5. Table 9.3 shows the rubric used to evaluate the active learning assignments that called for students to conduct experiments or create design artifacts. Specialized rubrics were created for the team project assignments.

No statistical comparisons of performance on the assignment write-ups were made between students in the in-person section and the asynchronous online sections of CIS 487 during Fall 2021 or for the in-person and online sections of CIS 488 in Winter 2022. However, informal comparisons of student data from the two modes of CIS 487 delivered by the instructor in Fall 2021 suggest that students attending the in-person class meetings produced work, which seemed to receive

Table 9.3 CIS 487/488 activity question rubric

Topic	Rating and feedback (0=missing, 4=satisfactory, 5=exceeds specification)
Quality of answers	
Completeness of write-up	

higher scores using similar grading rubrics. Similar observations were made for CIS 488 students in Winter 2022.

The authors created four research questions to compare the levels of engagement by students taking CIS 487 and 488 under flipped classroom in-person (FC) active learning as compared to the engagement of students taking previous offerings of CIS 487 and 488 with fewer active learning opportunities.

RQ1: Is the flipped classroom student performance worse than student performance in other course delivery modes?

To answer this question, the authors looked at data analytics (number of late and missing assignments) collected by the Canvas management system for three iterations of each course sequence shown in Tables 9.4 and 9.5. To briefly describe the difference between the semesters, Fall 2016 and Winter 2017 represented a Lecture-Heavy (LH) version of the courses before active learning activities were fully introduced in the curriculum, while Fall 2017 and Winter 2018 represent an intermediate (IM) step between the previous LH version of the courses and the flipped classroom (FC) version that fully embraced active learning techniques in Fall 2021 and Winter 2022, both of which involved heavy social distancing.

In Table 9.4 for CIS 487, it initially appears that there was a decrease in student grades and performance as the class transitioned into using active learning. However, this is due to an outlier from an underperforming student. There is no statistical difference at the 95% confidence level from the student t-test for the overall grade between F2016-LH and F2017-IM for overall grade or average number of late assignments per student. Yet, there was a statistical difference in populations for the average number of missing assignments per student. We attribute this to the increased workload caused by having students report on their activities in the

Table 9.4 CIS 487 Canvas course analytics for the Fall 2016, Fall 2017, and Fall 2021 semesters

	F2016-LH N = 24	F2017-IM N = 22	F2021-FC N = 23
Average overall course grade	91.2%	84.6%	91.8%
Average number of late assignments per student	0.4	0.7	0.2
Average number of missing assignments per student	0.1	1.8	0.5

Table 9.5 CIS 488 Canvas course analytics for winter 2017, winter 2018, and winter 2022 semesters

	W2017-LH N = 17	W2018-IM N = 18	W2022-FC N = 23
Average overall course grade	87.7%	95.3%	95.1%
Average number of late assignments per student	0.6	0.2	0.6
Average number of missing assignments per student	2.9	3.1	2.2

class. There was no statistical difference at 95% confidence between F2016-LH and F2021-FC.

Although at first glance in Table 9.5 it would appear the introduction of active learning techniques in W2018-IM and W2022-FC had a positive effect on student performance, there was no statistical difference between the W2017-LH version of the course and either W2018-IM and W2022-FC at 95% confidence for overall course grade, average number of late assignments, and average number of missing assignments. Therefore, we conclude from this that flipped class student performance is at least not hindered in active learning course modalities, but it is important to keep in mind the added burden of daily assignment write-ups as students transition to new course delivery methods.

While it is not reflected in data shown in Tables 9.4 or 9.5, students seem to be exhibiting better communication skills in the flipped classroom delivery because of the increased writing and oral presentation requirements as compared to the lecture versions of the courses. While not measured explicitly in our work, most students seem to write better and more meaningful peer reviews as they progress through the courses. Team participation is better in the active learning classes than in the lecture heavy versions of the classes.

9.4.1 Course Surveys

We surveyed the students during the final weeks of each semester, to gather the students' own perceptions of their levels of engagement with the class, active learning, and gamification. The CIS 487 survey emphasized active learning and engagement. The CIS 488 survey emphasized gamification and engagement.

RQ2: Do flipped classroom students have a different perception of their level of engagement as reported on the CIS 487 final survey than students in other course delivery modes?

Students rated each statement on their perceptions of active learning and their engagement in the survey from 1 (strongly disagree) to 5 (strongly agree). The distribution of responses to each question for CIS 487 is seen in Table 9.6. We performed a statistical analysis of the responses using the Mann-Whitney U Test. We found no statistical difference between the responses for the F2016-LH and F2017-IM groups at the 95% confidence level. This indicated that in F2017-IM, we had begun to implement some activities that students in both groups did not seem to feel differently about their active learning and engagement in the course. However, students in the F2021-FC course were significantly different at the 95% confidence level than the F2016-LH group for survey questions 2–5. Students agreed more that the course activities were useful (65% vs. 26% strongly agree) and allowed them to apply what they learned (70% vs. 44% strongly agree), and when asked if they did not understand the connection between the class activities and other aspects of the course reported, they strongly disagreed 65% to 41%.

Table 9.6 End-of-course student perception survey results focusing on agreement with the statements evaluating their engagement for three CIS 487 courses

Survey statement	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Course
1. There were opportunities for me to actively engage in learning	1 (4%)	0	1 (4%)	3 (11%)	22 (81%)	F2016-LH
	0	0	1 (5%)	7 (33%)	13 (62%)	F2017-IM
	0	0	1 (5%)	4 (20%)	15 (75%)	F2021-FC
2. Course activities were a useful way to learn	3 (11%)	0	3 (11%)	14 (52%)	7 (26%)	F2016-LH
	0	2 (10%)	3 (14%)	11 (52%)	5 (24%)	F2017-IM
	0	0	1 (5%)	6 (30%)	13 (65%)	F2021-FC
3. Course activities let me apply what I learned	2 (7%)	2 (7%)	2 (7%)	9 (33%)	12 (44%)	F2016-LH
	1 (5%)	2 (10%)	3 (14%)	10 (48%)	5 (24%)	F2017-IM
	0	0	0	6 (30%)	14 (70%)	F2021-FC
4. Course is an example of active learning	2 (7%)	0	2 (7%)	12 (44%)	11 (41%)	F2016-LH
	0	1 (5%)	3 (14%)	8 (38%)	9 (43%)	F2017-IM
	0	0	1 (5%)	2 (10%)	17 (85%)	F2021-FC
5. I didn't understand connection between class activities and other aspects of course	11 (41%)	8 (30%)	6 (22%)	0	2 (7%)	F2016-LH
	10 (48%)	8 (38%)	2 (10%)	1 (5%)	0	F2017-IM
	13 (65%)	6 (30%)	1 (5%)	0	0	F2021-FC
6. Working in groups was an effective way for me to learn	2 (7%)	2 (7%)	3 (11%)	9 (33%)	11 (41%)	F2016-LH
	0	3 (14%)	3 (14%)	8 (38%)	7 (33%)	F2017-IM
7. I prefer to learn primarily through lecture	7 (26%)	6 (22%)	6 (22%)	6 (22%)	2 (7%)	F2016-LH
	6 (29%)	9 (43%)	2 (10%)	3 (14%)	1 (5%)	F2017-IM
8. I had more opportunities to actively engage in learning in this class compared to other classes I've taken	2 (7%)	0	0	9 (33%)	16 (59%)	F2016-LH
	0	1 (5%)	1 (5%)	11 (52%)	8 (38%)	F2017-IM

In addition, we also looked at comparing F2017-IM to F2021-FC students' perceptions of how active learning was different between the intermediate implementation of the course and a fully flipped classroom. Again, we used the Mann-Whitney U Test finding that the student populations for survey questions 2–4 were significantly different at the 95% confidence level. Students more strongly felt that course activities were a useful way to learn (65% vs. 24% strongly agree) and that the course let them apply what they learned (70% vs. 24%).

We additionally asked students to rate their engagement for six survey questions on a scale from "very little of their time" to "most of the time" for specific behaviors. Unfortunately, as we redesigned the course, we modified the survey for F2021-FC to be less time intensive and do not have student response data for questions 9–14 as seen in Table 9.7. Therefore, we only compare F2016-LH to F2017-IM.

We used a Mann-Whitney U Test to statistically compare populations. At the 95% confidence interval only, question 13 had a statistical difference. Students stated stronger disagreement in F2017-IM for being expected to memorize facts

Table 9.7 End-of-course student survey focusing on rating active learning elements of their experience in three courses of CIS 487

Survey statement	Very little of the time	Less than half the time	At least half the time	Most of the time	Course
9. I was actively engaged in my learning	0	1 (4%)	9 (33%)	17 (63%)	F2016-LH
	0	0	7 (33%)	14 (67%)	F2017-IM
10. The professor created opportunities for me to actively engage in my learning	0	3 (11%)	3 (11%)	21 (78%)	F2016-LH
	0	0	8 (38%)	13 (62%)	F2017-IM
11. I applied the course material to real-world situations	3 (11%)	2 (7%)	8 (30%)	14 (52%)	F2016-LH
	2 (10%)	4 (19%)	8 (38%)	7 (33%)	F2017-IM
12. My small group worked effectively and collaboratively	1 (4%)	2 (7%)	6 (22%)	18 (67%)	F2016-LH
	0	3 (14%)	8 (38%)	10 (48%)	F2017-IM
13. I was expected to memorize facts and information	4 (15%)	16 (59%)	1 (4%)	6 (22%)	F2016-LH
	10 (48%)	9 (43%)	1 (5%)	1 (5%)	F2017-IM
14. I spent time working on activities that were too simplistic or irrelevant	15 (56%)	8 (30%)	2 (7%)	2 (7%)	F2016-LH
	10 (48%)	10 (48%)	1 (5%)	0	F2017-IM

and information than students in F2016-LH with 48% to 15% strong disagreement. We attribute this to the insertion of small activities in some lectures, which moved students from listening to lectures to investigating and discovering how the information they have learned works in practice.

Overall, student agreement seemed stronger when compared to the intermediate course than the lecture heavy. It may be possible that an incomplete or partial implementation of active learning techniques in a class prevents or diminishes students' perceptions of active learning. We suggest that those wishing to implement similar changes in their pedagogical approach may be better served by fully embracing an active learning course redesign rather than a slow or partial implementation spread out over several semesters.

Without prompting students in the active learning courses showed strong preference for working on the activities and projects, as opposed to taking exams. They felt that the activities and project-based learning approach not only prepared them better for their senior design class but also prepared them better for their careers.

Overwhelmingly, the projects are the biggest strength cited by students in the course surveys. Their comments reinforce the positive effect of projects on practical learning as well as the development of collaborative, problem-solving skills. Several students also indicated that replacing exams with projects provided a more meaningful learning experience and knowledge that would be otherwise difficult to assess with a traditional assessment approach.

RQ3: Does gamification affect the choices of flipped classroom students differently than students in other course delivery modes as reported on the CIS 488 final survey?

Gamification was examined in the CIS 488 final survey (see Table 9.8). We only administered this survey to students taking this course in Winter 2017 and Winter 2022. Students again submitted their responses as a 1 (strongly disagree) to 5 (strongly agree). We also performed a statistical analysis with the Mann-Whitney U test on the W2017-LH and W2022-FC student populations. At 95% confidence level, there was no statistical difference between the two groups' responses. Both student groups seem evenly split on statement 3, "I did what I had to, but didn't feel I had a choice," while also agreeing 70% vs. 80% with statement 2, "I felt like I

Table 9.8 CIS 488 end of course survey on student perceptions on gamification

Survey statement	Strongly disagree	Disagree	Neutral	Agree	Strongly agree	Course
1. I put more effort into assignments for than I normally do for the courses I take	0 0	1 (10%) 0	3 (30%) 2 (11%)	3 (30%) 9 (50%)	3 (30%) 7 (39%)	W2017-LH W2022-FC
2. I felt like I had more control and choice over the assignments I completed than I normally do	1 (10%) 0	1 (10%) 0	1 (10%) 2 (11%)	2 (20%) 8 (44%)	5 (50%) 8 (44%)	W2017-LH W2022-FC
3. In this course, I did what I had to, but I didn't feel like it was really my choice	1 (10%) 2 (11%)	3 (30%) 7 (39%)	2 (20%) 3 (17%)	2 (20%) 4 (22%)	2 (20%) 2 (11%)	W2017-LH W2022-FC
4. In this course, I picked assignments based on what interested me	1 (10%) 1 (6%)	0 1 (6%)	0 4 (22%)	5 (50%) 5 (28%)	4 (40%) 7 (39%)	W2017-LH W2022-FC
5. In this course, I feel I had control over how I demonstrated my understanding of the course material	1 (10%) 0	1 (10%) 1 (6%)	0 1 (6%)	4 (40%) 7 (39%)	4 (40%) 9 (50%)	W2017-LH W2022-FC
6. When picking the assignments you submitted for this course, how important to you when deciding was how many points I could earn by doing the assignment?	0 2 (11%)	1 (10%) 0	3 (30%) 11(61%)	2 (20%) 2 (11%)	4 (40%) 3 (17%)	W2017-LH W2022-FC
7. When picking the assignments you submitted for this course, how important to you when deciding was how much the assignment allowed me to collaborate with my classmates?	1 (10%) 0	3 (30%) 4 (22%)	2 (20%) 7 (39%)	2 (20%) 4 (22%)	2 (20%) 3 (17%)	W2017-LH W2022-FC

had more control and choice than I normally do." We conclude that in our limited study, it did not appear that students in the W2022-FC class were influenced by gamification than students in the lecture delivery mode.

However, student comments clearly indicated they liked being given the ability to make choices that impacted their learning. It allowed them to tailor their experiences directly to their interests and skills. We believe this contributed to the high quality of the games produced by the students during the semester.

We suggest this was due to an increase in motivation caused by being permitted to pursue their individual interests. As one student wrote reflective of multiple other comments, "I'm more driven to do a good job, since I choose to do it." Meanwhile, another student commented, "This inspires creativity and forces students to solve real world problems, along with delivering a full product." Interestingly, the point valuation seemed less important to the students when picking an assignment even if it meant fewer points were awarded.

9.4.2 Course Evaluations

Students on our campus are requested to complete a standard set of course evaluations at the end of the semester. The evaluation form is completed online and anonymously prior to receiving their final course grades. We wanted to compare the course evaluations of socially distanced students in other active learning conditions.

RQ4: Do flipped classroom students have different course experiences than students in other course delivery modes?

Questions are rated from 1 (strongly disagree) to 5 (strongly agree). Our college redesigned the course evaluations during the period between W2018 and F2021 to solicit different information, so we have included the most pertinent survey questions for CIS 487 in Table 9.9 and for CIS 488 in Table 9.10.

The student comments on the course evaluations indicated that they enjoyed the design activities and felt these activities helped them when creating their project deliverables. They also felt that sharing ideas and insights with other students during class discussions helped them learn. They enjoyed being able to apply the material covered in the lectures and tutorials to solve actual problems.

Students appreciated the class activities for a variety of reasons. They felt these activities were more engaging than just listening to a lecture accompanied by slides. The students liked the redundancy that was built in the activities that often had them look at different facets of similar design concerns. Some students wrote that they felt the group work and writing activity summaries helped them become more at ease when speaking in class.

Students felt that the strengths of this course were the dynamic learning activities, the lack of exams, and game project development. They also felt that completing the class activities collaboratively provides better opportunities for students to master the material.

Table 9.9 CIS 487 end-of-term collegiate course evaluations

1 = Strongly disagree 5 = Strongly agree	F2016-LH N = 21	F2017-IM N = 22	F2021-FC N = 24
Course met my expectations	4.33	4.55	4.56
Course objectives were clear	4.24	4.36	4.67
Typical workload compared to other courses	~	~	4.21
Course advanced my understanding of subject	~	~	4.75
Lab activities increased my understanding of lecture topics	~	~	2.92
I knew what was expected of me	~	~	4.52
Overall course rating	4.52	4.73	4.63

Table 9.10 CIS 488 end-of-term collegiate course evaluations

1 = Strongly disagree 5 = Strongly agree	W2017-LH N = 13	W2018-IM N = 13	W2022-FC N = 11
Course met my expectations	4.85	4.38	4.50
Course objectives were clear	4.85	4.31	4.64
Typical workload compared to other courses	~	~	4.00
Course advanced my understanding of subject	~	~	4.50
Lab activities increased my understanding of lecture topics	~	~	2.09
I knew what was expected of me	~	~	4.64
Overall course rating	4.85	4.46	4.45

9.4.3 Lessons Learned

We believe that some of our findings can be applied to other engineering project courses. Looking at the course analytics, course evaluations, and engagement survey data, we found two common themes. The first is that there are few statistical differences in the academic performance between students in the lecture heavy versions of the courses and flipped classroom versions. We interpret this to mean that these two courses successfully transitioned from lecture heavy to active learning. Harder to measure is the growth in the students' soft skills (written and oral communication, collaboration, and project management). The second is that students feel more engaged in the active learning versions of these courses and like the flexibility gamification brings them.

Students enjoyed the role-play (CIS 488) and felt is added to the realism of the development process. It is interesting to note that seven student teams from CIS

488 have gone on to form LLCs to continue their game development activities professionally. This did not happen prior to the introduction of the failing game company role-play in CIS 488.

Students enjoyed the class activities but sometimes needed more guidance (scaffolding) and more time to complete some of the activities. Students love working in small groups, but they do not like talking to the whole class without a script. In courses involving both in-person and online students, both types of students were included on the same project teams as this tends to increase online student engagement.

9.4.4 Threats to Validity

We recognize that one of the limitations of this study was that we did not have a control group. We also acknowledge that the instructor teaching all CIS 487 and 488 course offerings may also account for the lack of significant differences on some of the evaluation measures.

The asynchronous course delivery, pre- and post-Covid shutdown, was significantly different than that occurring using zoom during 2020 or 2021. Prior to 2019 and university implemented a policy which required the pairing of an asynchronous, distance learning section with a face-to-face section of the same course. The live class sessions were captured, verbatim, for later viewing by the asynchronous students. This course feature was implemented in most CIS courses prior to Covid. This provided an advantage to the 2019 and 2021 asynchronous online sections in that they could witness the live lecture and some class activities as a virtual classroom observer. During 2021–2022, the asynchronous online students were only able to view the class activities plus any recorded videos posted for pre-class viewing by the flipped class in-person students.

One area of uncertainty when measuring the student responses is the unknown amount of interaction between students in the synchronous and asynchronous sections of CIS 487 and 488. Students in the CIS department know each other from other classes that they have taken together. Even though a student registered in the asynchronous online section was not allowed to attend any in-person class meetings, it is quite possible that a friend from an in-person course section may have shared their course experiences with them giving them additional insight into group activities completed in the classroom. In other words, the asynchronous student may not be totally isolated from knowledge learned in the group activities. We did not attempt direct comparisons between in-person and online students in this chapter. Student engagement can only be measured indirectly in online courses using surveys and course analytics. In 2016–2017, direct observation of student behavior was used to provide insight into their levels of engagement among in-person class instruction. We did not include direct observation of students in the socially distanced in-person sections of either CIS 487 or CIS 488. Trying to measure student engagement using chat comments or interaction with shared Google documents is a viable alternative but also lacks the immediate visual feedback an instructor experiences with a real-time view of a student's face.

There were no surveys taken between the Winter 2017 and Winter 2022 offerings of CIS 488. These surveys provide the most direct and candid feedback on active learning from the student's perspective. Although 2017 CIS 488 survey data provides some good baseline data, it would have been more beneficial to have data from Winter 2018.

The 2019–2020 and 2020–2021 academic years presented extraordinary challenges for students. All students, not just those from this university, were asked to learn under circumstances never-before experienced. While it would be expected that many students were excited to return to face-to-face instruction, it may also be expected that many felt anxious or even distracted with the fresh look of face-to-face instruction. It is difficult to assess what effects, both positive and negative, this might have had on the return to an active learning classroom in Fall 2021.

9.5 Conclusions and Future Direction

During the past 3 years, most institutions across the world were required to switch to online formats. This switch to using videoconferencing often required major adjustments to course design and left many students simply watching online lecture videos and taking exams. We demonstrated in previously reported studies that it is possible to move an in-person active learning software engineering course online [68, 69]. We also believe that engineering project courses can be run using social distancing Covid protocols without observing significant reductions in student levels of engagement as compared to other course formats. We take this as evidence that it is possible to design a socially distanced active learning course that can be more engaging than its online counterpart. We credit the active learning components of the class and the levels of student interaction that accompany them for making this possible. We encourage other instructors to adopt active learning practices and modify them as needed to satisfy Covid protocol requirements in their course deliveries to achieve higher levels of student satisfaction and engagement.

We were encouraged by the enthusiasm that students exhibited while working with the active learning modules during the in-person class meetings and look forward to continuing to develop this course content. It may be important to develop ways in which asynchronous students are encouraged to be a part of some sort of face-to-face experience, even if it is not during formal online class meetings. Informal study or discussion groups that would meet online, with flexible meeting times, might be a way to increase engagement with activities. The demand for online game design offerings is strong among the students on our campus. Experiences from the Fall 2021 course delivery of CIS 487 and Winter 2022 course delivery of CIS 488 will be used to revise the next offering of these courses and their corresponding active learning materials. The challenge will be to seek ways to ensure that online students feel engaged with the class materials.

Acknowledgments This project was partially supported by a grant from the University of Michigan-Dearborn Advancement of Teaching and Learning HUB Creative Teaching Fund.

References

1. Becker, K.: Teaching with games: the minesweeper and asteroids experience. J. Comput. Sci. Coll. **17**(2), 23–33 (2001)
2. Jones, R.: Design and implementation of a computer games: a capstone course for undergraduate computer science education. In: Proceedings of 31st SIGCSE Technical Symposium (Austin, TX, March 2000), pp. 260–264. ACM Press, New York, NY (2000)
3. Pleva, G.: Game programming and the myth of child's play. J. Comput. Sci. Coll. **20**(2), 125–136 (2004)
4. Claypool, K., Claypool, M.: Software engineering design: teaching software engineering through game design. In: Proceedings of 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, Caparica, Portugal, June 2005, pp. 123–127. ACM Press, New York, NY (2005)
5. Rollings, A., Morris, D.: Game Architecture and Design. New Riders, Indianapolis, IN (2004)
6. Maxim, B.R.: Software Requirements Analysis and Design. NIIT, Atlanta, GA (2004)
7. Samavedham, L., Ragupathi, K.: Facilitating 21st century skills in engineering students. J. Eng. Educ. **XXVI**(1), 38–49 (2012)
8. Maxim, B.R., Acharya, S., Brunvand, S., Kessentini, M.: WIP: introducing active learning in a software engineering course. In: Proceedings of the 2017 Annual Meeting of the American Society for Engineering Education, Columbus, OH, June 2017, pp. 1–12
9. Promoting Active Learning. https://utah.instructure.com/courses/148446/pages/active-learning. Accessed 25 Feb 2016
10. Prince, M.: Does active learning work? A review of the research. J. Eng. Educ. **93**, 223–231 (2004)
11. Luster-Teasley, S., Hargrove-Leak, S.C., Waters, C.: NSF TUES: Transforming undergraduate environmental engineering laboratories for sustainable engineering using the case studies in the sciences instructional method. In: Proceedings of the 2014 Annual Meeting of the American Society for Engineering Education, Indianapolis, IN, June 2014
12. Jungic, V., Kaur, H., Mulholland, J., Xin, C.: On flipping the classroom in large first-year calculus courses. Int. J. Math. Educ. Sci. Technol. **46**(4), 1–8 (2015)
13. Raju, P.K., Sanker, C.C.: Teaching real-world issues through case studies. J. Eng. Educ. **88**(4), 501–508 (1999)
14. Nickels, K.M.: Do's and don'ts of introducing active learning techniques. In: Proceedings of the 2000 Annual Meeting of the American Society for Engineering Education, St. Louis, Missouri, June 2000
15. Lavelle, J.P., Stimpson, M.T., Brill, E.D.: Flipped out engineering economy: converting a traditional class to an inverted model. In: Krishnamurthy, A., Chan, W.K.V. (eds.) Proceedings of the 2013 Industrial Systems Engineering Research Conference, pp. 397–407 (2013)
16. Wood, K., Jensen, D., Dutson, A., Green, M.: Active learning approaches in engineering design courses. In: Proceedings of the 2003 Annual Meeting of the American Society for Engineering Education, Nashville, Tennessee, June 2003
17. Maxim, B.R., Decker, A., Yackley, J.J.: Student engagement in active learning software engineering courses. In: Proceedings of 49th IEEE Annual Frontiers in Education Conference, Cincinnati, OH, October 2019, pp. F3G1–F3G5
18. Yelamarthi, K., Member, S., Drake, E.: A flipped first-year digital circuits course for engineering and technology students. IEEE Trans. Educ. **58**(3), 179–186
19. Meier, R.D.: Active learning in large lectures. In: Proceedings of the 1999 Annual Meeting of the American Society for Engineering Education, Charlotte, North Carolina, June 1999

20. Krause, R., Hayton, A.C., Wonoprabowo, J., Loo, L.: Is engagement alone sufficient to ensure "active learning?". Loma Linda Univ. Stud. J. **2**(1) (2017)
21. Ardis, M., Chenoweth, S., Young, F.: The 'Soft' topics in software engineering Education. In: Proceedings of 38th Annual Frontiers in Education Conference (Vol. 1, Oct 2008), pp. F3H1–F3H6. IEEE Press, Saratoga Springs, NY (2008)
22. Day, J.A., Foley, J.D.: Evaluating a web lecture intervention in a human-computer interaction course. IEEE Trans. Educ. **49**(4), 420–431 (2006)
23. Bishop, J.L., Verleger, M.A.: The flipped classroom: a survey of the research. In: Proceedings of the 2017 Annual Meeting of the American Society for Engineering Education, Atlanta, GA. (2013)
24. Wu, P., Manohar, P., Acharya, S.: The design and evaluation of class exercises as active learning tools in software verification and validation. Inf. Syst. Educ. J. (2016)
25. Cheng, L., Ritzhaupt, A.D., Antonenko, P.: Effects of the flipped classroom instructional strategy on students' learning outcomes: a meta-analysis. Educ. Technol. Res. Dev. **67**(4), 793–824 (2018)
26. Morrison, G.R., Ross, S.M., Kemp, J.E., Kalman, H.: Designing Effective Instruction. Wiley (2010)
27. Savery, J., Duffy, T.: Problem-based learning: an instructional model and its constructivist framework. Educ. Technol. **35**(5), 31–38 (1995)
28. Silva, A., Bispo, A., Rodriguez, D., Vasquez, F.: Problem-based learning: a proposal for structuring PBL and its implications for learning among students in an undergraduate management degree program. Revista de Gestão. **25**(2), 160–177 (2018)
29. Warnock, J.N., Mohammadi-Aragh, M.J.: Case study: Use of problem-based learning to develop students' technical and professional skills. Eur. J. Eng. Educ. **41**(2), 142–153 (2016)
30. Dunlap, J.: Problem-based learning and self-efficacy: how a capstone course prepares students for a profession. Educ. Technol. Res. Dev. **53**(1), 65–83 (2005)
31. Urbanic, R.: Developing design and management skills for senior industrial engineering students. J. Learn. Des. **4**(3), 35–49 (2011)
32. Gavin, K.: Case study of a project-based learning course in civil engineering design. Eur. J. Eng. Educ. **36**(6), 547–558 (2011)
33. Souza, M., et al.: Students perception on the use of project-based learning in software engineering education. In: SBES 2019: Proceedings of the XXXIII Brazilian Symposium on Software Engineering, pp. 537–546 (2019)
34. Kothiyal, R., et al.: Effect of think-pair-share in a large CS1 class: 83% sustained engagement. In: Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research (ICER '13), pp. 137–144. ACM, New York, NY (2013)
35. Nagappan, M., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., Balik, S.: Improving the CS1 experience with pair programming. In: Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03), pp. 359–362. ACM, New York, NY (2003)
36. Porter, L., Bouvier, D., Cutts, Q., Grissom, S., Lee, C., McCartney, R., Zingaro, D., Simon, B.: A multi-institutional study of peer instruction in introductory computing. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16), pp. 358–363. ACM, New York, NY (2016)
37. Greer, T., Hao, Q., Jing, M., Barnes, B.: On the effects of active learning environments in computing education. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27–March 2, 2019, Minneapolis, MN, 6 pages. ACM, New York, NY
38. Hoffman, B., Morelli, R., Rosato, J.: Student engagement is key to broadening participation in CS. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27–March 2, 2019, Minneapolis, MN, 7 pages. ACM, New York, NY
39. Ham, Y., Myers, B.: Supporting guided onquiry with cooperative learning in computer organization. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), pp. 273–279. ACM, New York, NY (2019)

40. Stone, J.A., Madigan, E.: Experiences with community-based projects for computing majors. J. Comput. Sci. Coll. **26**(6), 64–70 (2011)
41. Kharitonova, Y., Luo, Y., Park, J.: Redesigning a software development course as a preparation for a capstone. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27–March 2, 2019, Minneapolis, MN, 7 pages. ACM, New York, NY
42. Decker, A., Simkins, D.: Leveraging role play to explore the software and game development process. In: Proceedings of 46th IEEE Annual Frontiers in Education Conference, Erie, PA, October 2016, pp. S3F6–S3F10
43. Simkins, D.: The arts of larp: design, literacy, learning, and community in live-action role play. McFarland, Jefferson, NC (2015)
44. Moroz-Lapin, K.: Role play in HCI studies. In: Proceedings of the 2009 international conference on HCI Educators: playing with our Education (HCIEd'09), pp. 12–12. British Computer Society, Swinton (2009)
45. Seland, G.: Empowering end users in design of mobile technology using role play as a method: reflections on the role-play conduction. In: Kurosu, M. (ed.) Proceedings of the 1st International Conference on Human Centered Design: Held as Part of HCI International 2009 (HCD 09), pp. 912–921. Springer, Berlin (2009)
46. Zowghi, D., Paryan, S.: Teaching requirements engineering through role playing: lessons learnt. In: Proceedings of the 11th IEEE International Conference on Requirements Engineering (RE '03), pp. 233–241. IEEE Computer Society, Washington, DC (2003)
47. Börstler, J.: Improving CRC-card role-play with role-play diagrams. In: Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '05), pp. 356–364. ACM, New York, NY (2005)
48. Vold, T., Yayilgan, S.Y.: Playful participation for learning in higher education — The introduction of participatory role play simulation in a course at Hedmark University College. In: Proceedings of 2013 International Conference on Information Technology Based Higher Education and Training (ITHET), Antalya, 2013, pp. 1–4
49. Rudra, A., Jaeger, B., Aitken, A., Chang, V., Helgheim, B.: Virtual team role play using second life for teaching business process concepts. In: Proceedings of 44th Hawaii International Conference on System Sciences (HICSS), Kauai, HI, 2011, pp. 1–8
50. Maxim, B.R., Kaur, R., Apzynski, C., Edwards, D., Evans, E.: An agile software engineering process improvement game. In: Proceedings of 46th IEEE Annual Frontiers in Education Conference, Erie, PA, October 2016, pp. S3F1–S3F5
51. Nakamura, T., Maruyama, H., Takashima, A., Sambe, Y.: Role-play exercises for project management education that incorporate a software agent. In: Proceedings 2012 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE), Hong Kong, 2012, pp. W2A-8–W2A-14
52. Navarro, E., Hoek, A.: SimSE: an interactive simulation game for software engineering education. In: Proceeding of the Seventh IASTED International Conference on Computers and Advanced Technology in Education, pp. 12–17 (2004)
53. Maxim, B.R., Decker, A., Brunvand, S.: Use of role-play and gamification in a software project course. In: Proceedings of 47th IEEE Annual Frontiers in Education Conference, Indianapolis, IN, October 2017, pp. T3D1–T3D5
54. Yackley, J.J., Maxim, B.R., Brunvand, S., Decker, A.: Active learning and gamification in game design courses. In: Proceedings of Meaningful Play 2018 Conference, East Lansing, MI, October 2018, pp. 165–178
55. Domínguez, A., Saenz-de-Navarrete, J., de-Marcos, L., Fernández-Sanz, L., Pagés, C.A., Martínez-Herráiz, J.J.: Gamifying learning experiences: practical implications and outcomes. Comput. Educ. 380–392
56. Simões, J., Redondo, R.D., Vilas, A.F.: A social gamification framework for a K-6 learning platform. Comput. Hum. Behav. **29**, 345–353 (2012)
57. Gee, J.P.: What Video Games Have to Teach Us About Learning and Literacy, 2nd edn. St. Martin's Press (2014)

58. Gee, J.P.: What video games have to teach us about learning and literacy. Comput. Entertain. **1**(1), 1–4 (2003)
59. Vygotsky, L.S.: Mind and Society: The Development of Higher Mental Processes. Harvard University Press (1978)
60. Granic, I., Lobel, A., Engels, R.: The benefits of playing video games. Am. Psychol. **69**(1), 66–78 (2014)
61. Ott, M., Tavella, M.: A contribution to the understanding of what makes young students genuinely engaged in computer-based learning tasks. Procedia Soc. Behav. Sci. **1**(1), 184–188 (2009)
62. Lee, J.J., Hammer, J.: Gamification in education: what, how, why bother? Definitions and uses. Exchange Organ. Behav. Teach. J. **15**(2), 1–5 (2011)
63. Yang, Y.T.C.: Building virtual cities, inspiring intelligent citizens: digital games for developing students' problem solving and learning motivation. Comp. Educ. **59**(2), 365–377 (2012)
64. Toth, D., Kayler, M.: Integrating role-playing games into computer science courses as a pedagogical tool. In: Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15), pp. 386–391. ACM, New York, NY (2015)
65. Bogost, I.: Gamification is bullshit. http://bogost.com/writing/blog/gamification_is_bullshit/. Accessed 30 Aug 2018
66. O'Donnell, C.: Getting played: gamification, bullshit, and the rise of algorithmic surveillance. Surveill. Soc. **12**(3), 349–359 (2014)
67. Schell, J.: The Art of Game Design: A Book of Lenses. CRC Press (2015)
68. Maxim, B.R., Limbaugh, T., Yackley, J.J.: Student engagement in an online software engineering course. In: Proceedings of 51st IEEE Annual Frontiers in Education Conference, Lincoln, NE, October 2021, pp. T3D1–T3D9
69. Maxim, B.R., Limbaugh, T.: WIP: Engaging software engineering students in synchronous asynchronous on-line course. In: Proceedings of the 2021 Annual Meeting of the American Society for Engineering Education, Long Beach, CA, July 2021, pp. 1–17

Chapter 10
A Framework for the Gamification of GUI Testing

Riccardo Coppola, Luca Ardito, Tommaso Fulcini, Giacomo Garaccione, Marco Torchiano, and Maurizio Morisio

Abstract Software testing is a critical activity in the software development process. Several techniques have been proposed, addressing different levels of granularity from low-level unit testing to higher-level exploratory testing through the software's graphical user interface (GUI). In modern software development, most test cases are obtained by automated test generation. However, while automation generally achieves high coverage in code-level white-box testing, it does not always generate realistic sequences of interactions with the GUI. By contrast, manual exploratory testing has survived as a costly, error-prone, and tedious yet crucial activity. Gamification is seen as an opportunity to increase user satisfaction and engagement while performing testing activities. It could also enable and encourage crowdsourced testing tasks. The purpose of the study described in this chapter is to provide a framework of gamification mechanics and dynamics that can be applied to the practice of manual exploratory GUI testing. We provide an implementation of the framework as an extension of an existing manual exploratory GUI testing for Web applications, and we provide a preliminary evaluation of the gamified tool in terms of provided efficiency, effectiveness, and user experience. Our results show that the gamified solution makes the testers obtain test suites with higher coverage while reducing slightly the number of bugs signalled while traversing the applications under test. The gamified tool also was considered to provide a positive user experience, and the majority of participants expressed their willingness to use such instruments again in the future. As future work, we foresee the implementation of the framework in a stand-alone tool and in-depth empirical experiment to evaluate quantitatively the benefits and drawbacks provided by such mechanics in real testing scenarios.

Keywords Gamification · Software engineering · Software testing · Web application testing · GUI testing

R. Coppola (✉) · L. Ardito · T. Fulcini · G. Garaccione · M. Torchiano · M. Morisio
Politecnico di Torino, Department of Control and Computer Engineering, Torino, Italy
e-mail: riccardo.coppola@polito.it; luca.ardito@polito.it; tommaso.fulcini@polito.it; giacomo.garaccione@polito.it; marco.torchiano@polito.it; maurizio.morisio@polito.it

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_10

10.1 Introduction

Software testing is a critical activity in the software development process. Its main purpose is to detect defects and faults in advance in the code produced, avoiding to release code affected by bugs and security issues, whose repair cost increases once the software is released to the final users. Testing is also utilized to prove conformance to functional requirements and the reliability of software artefacts. Several software testing techniques exist in the literature, ranging from low-level unit testing of atomic components of the software (unit tests) to higher-level exploratory testing through the finalized software (end-to-end or E2E testing). When conducted through the graphical user interface (GUI) of the software under test (SUT), E2E testing is typically called GUI testing.

In modern software development, test cases are often obtained through automated test generation, a practice that ensures significant time saving and repeatability of test practices. However, while automation generally achieves high coverage in code-level white-box testing activities, automated testing is not always the optimal choice to generate E2E test suites, with the generation of interaction sequences that have a low level of realism in mimicking the final user's interaction with the system. Therefore, a relevant portion of E2E testing activities is still conducted manually by the QA team. Manual testing is however renowned as a costly, error-prone, and tedious yet crucial activity [1]. Throughout this manuscript, we will refer to GUI testing as the tool-aided activity conducted by development companies or external test factories, and not to activities performed by the final users of the applications (e.g., beta testing or crowdsourced verification of Web applications). Our focus is also on GUI-based *functional* testing, i.e., to all activities related to the verification of the main features of the SUT.

Gamification, defined as *the use of game design elements in non-game contexts* [2], is gaining traction in the latest years in disciplines related to computer science, because of its proven capability in motivating, engaging, and improving the performance of the participants of tasks to which game mechanics are applied [3, 4]. Several frameworks have been proposed to govern and aid the design of gamified activities and tools. In this chapter, we adopt as a reference the Octalysis framework proposed by Yu-Kai Chou [5]. The framework identifies eight core drives that represent aspects of human behavior that can be stimulated through gamification.

Several works in software engineering literature have motivated, conceptualized, and evaluated gamification mechanics to improve the results of many activities of the software process [6]. Albeit the primary application of gamification mechanics is still used primarily in the educational field [7], there is a growing interest by practitioners in implementing them in industrial contexts and tooling [8]. Case studies in the literature have documented encouraging outcomes by such adoption [9].

Of all software engineering activities, software testing is particularly suitable for the application of the most common gamification elements. Testing activities, in fact, typically produce quantitative, measurable, and comparable results (e.g., coverage thresholds reached, number of defects found, number of crashes triggered) that can be naturally translated to game-like aspects (e.g., points and leaderboards).

The present chapter has the goal of proposing a structured methodology to apply gamification in the domain of exploratory GUI testing, a facet of the software testing practice which is still not covered by related literature. We propose a set of game mechanics that can be adopted for testing both Web and android applications, with a scoring algorithm that can be used in a general context. We aim to chart a viable path for researchers and practitioners that will approach the topic of gamification in the GUI testing discipline.

We report the process of adapting gamified mechanics to the activity of manual exploratory GUI testing. To that extent, we perform a preliminary investigation of the current state of the art and practice in the field of gamified software testing, to identify the most mentioned tools and adopted mechanics, and the benefits and drawbacks provided by the techniques. We then detail a framework of mechanics for the gamification of GUI testing, incorporating game elements like session scores, leaderboards, and live graphical feedback. The framework has been developed as a prototype for GUI testing of Web applications, but it is by design adaptable to different domains. We finally report the findings of an experiment conducted with graduate students, to evaluate the improvements in effectiveness and user experience of the gamified tool when compared to the non-gamified equivalent.

The framework we present brings a novel contribution to the current state of the art related to the gamification of software testing, being an example of a gamified tool for manual exploratory GUI testing: an analysis of the literature we have performed has revealed that this is still a relatively unexplored field. More specifically, no solutions have been proposed previously specifically for exploratory manual GUI testing of Web applications. The framework extends two previous works: the prototype framework proposed by Cacciotto et al. [10] and an extension of it by Fulcini and Ardito [11], which also saw a first preliminary evaluation of the framework.

The chapter is organized as follows: Sect. 10.2 presents a background about software testing techniques and gamification in the software engineering discipline. Section 10.3 provides a survey of the existing scientific literature regarding gamified software testing. Section 10.4, based on the findings of the literature review, presents a conceptual framework for gamification of manual exploratory GUI testing. Section 10.5 describes the methodology, setting, and results of an empirical evaluation of the framework. Section 10.6 discusses the threats to validity of the framework. Finally, Sect. 10.7 concludes the chapter with an overview of the findings and future research directions.

10.2 Background and Related Work

This section illustrates the main concepts of GUI testing: the major available technologies, the main limitations, and open challenges. We also introduce the background concepts about the utilization of gamification in software engineering.

10.2.1 GUI Testing

GUI testing is a form of functional testing, which exercises a software under test (SUT) of any given domain through its graphical user interface. GUI testing exercises the SUT by mimicking the operations that would be performed by its typical end—that in that sense, GUI testing is a form of End-to-End (E2E) or system testing since it aims at defining scenario-based test cases to cover the functional requirements of the SUT.

Many GUI testing tools and techniques have emerged in the last three decades, and now they are available for different platforms and domains. All methodologies and technologies used for GUI testing share several commonalities. GUI test scenarios are typically defined as a sequence of different *locators*, i.e., graphical elements that have to be recognized and interacted with inside the GUI; each locator in a GUI test sequence is typically associated with a specific operation, e.g., mouse clicks and movements, key presses, or text insertions. GUI test cases also make use of different forms of *oracles*, i.e., visual cues or properties of the GUIs that are checked to verify the conformance of the SUT's behavior with the functional requirements.

GUI testing activities are not inherently automated; in fact, related literature highlights that a significant portion of GUI testing is still performed manually by practitioners.

Automated support for GUI testing is made available by a large number of commercial and academic tools, which can be classified under two different categorizations. Alégroth et al. define three different *generations* of GUI testing tools, based on the type of *locators* and *oracles used* [12]:

- *First generation*, or coordinate-based testing tools, use exact coordinates to locate and verify the presence of elements on the screen. Coordinate-based locators were used in the very first tools in the field and have been largely abandoned because of their flakiness and unreliability.
- *Second generation*, or property-based testing tools, use textual properties of the graphical elements of the GUI to identify and verify them (e.g., XML attributes in layout files of mobile applications or HTML attributes and properties in DOM files describing the appearance of Web apps). Property-based testing tools are currently the most widespread methodology of GUI testing: *selenium* and *espresso* are prominent examples of such a category of GUI testing tools for the mobile and Web domains, respectively.

- *Third generation*, or visual testing tools, use image recognition to locate and verify images in the pictorial GUI of the application. Visual GUI testing tools are still an emerging research direction in the literature [13].

Another possible categorization of GUI testing techniques can be performed based on the way the sequences of interactions against the GUI are selected and recorded into test scripts [14]:

- *Automation frameworks and APIs* are tools that allow the creation of scripts involving methods that access the hierarchy of components of the GUI. The frameworks offer methods that allow executing specific operations on the located widgets.
- *Record and replay* tools allow the tester to manually execute operations against the SUT's GUI to have them recorded into re-executable test scripts.
- *Automated input generation (AIG)* tools allow generating the sequences of interactions against the GUI without human scripting. These tools can be based on random input generation or leverage models of the GUI to test.

Although many tools and techniques are available, there is evidence that GUI testing tools are not widely adopted by commercial and open-source projects, and GUI testing is conducted mostly manually, using random tools, or completely neglected.

The scarce diffusion of GUI testing is partly related to inherent technical issues: GUI test cases exhibit very high *fragility* (i.e., the necessity of performing important maintenance operations of GUI test cases when the SUT's GUI evolves [15]), *flakiness* (i.e., the possibility that the same test case produces unpredictable results when applied to the same SUT [16]), and *fragmentation* (i.e., the necessity to execute the test cases against different rendering of the GUI on varying configurations, browsers, or devices [17]). Recent surveys in the literature have identified the main limitations of GUI testing in industrial practice. Even if most of the identified challenges are technological and tool-related (e.g., the difficulty in coping with application changes that may break test execution; timing and synchronization issues between the test cases and the SUT; missing means to provide robust identification of GUI widgets), some of the identified challenges are related to the requirement of specific skills and trained professionals for the execution of even trivial GUI testing activities [18].

However, it is widely accepted in related literature that defining test scripts is typically considered by developers as a time-consuming, error-prone, and boring activity [19]. The typically low engagement of testing activities makes them often overlooked in computing curricula, thereby creating a lack of knowledge about testing techniques in junior developers [20].

10.2.2 Gamification in Software Engineering

Many works in related literature have applied gamified concepts to the field of software engineering.

In their systematic mapping, Pedreira et al. report that gamification is being adopted in many different phases of the software process [2]. The principal application areas are software implementation, project monitoring and control, and collaboration between team members. Software testing is the fifth most mentioned area in related work about gamified software engineering practices. Gamification in software engineering is also a significantly growing trend in the field, as reported in the literature review performed by Barreto et al. [21].

The mentioned surveys identified badges and leaderboards as the most frequently mentioned gamification aspects used to enhance engagement in software engineering activities. The prevalent benefits provided by the application of gamification are increased performance in performing the gamified activities, higher product quality, and better learning. A few negative effects are also mentioned, e.g., ineffective and eventually dissatisfying experiences and cognitive overload on the participants.

Gamification is typically applied to software engineering practices that require high commitment and cooperation by their participants. de Melo et al., for instance, present a gamified platform for version control systems, to build a ranking of the most active developers contributing to software projects [22]. Ašeriškis et al. describe game rules for a project management system that they evaluated through the application of the standard system usability scale (SUS). The authors identified benefits provided by the utilization of the platform while guaranteeing relatively high usability to its users [23].

The literature reviews in the software engineering discipline report, however, that a high percentage of works in gamification (more than 70%) fails to report practical results obtained by the application of the technique. This aspect underlines the immaturity of the practice in software engineering while not taking into account the fundamental aspect of the user experience provided by the gamified tools and techniques.

10.3 Gamified Software Testing: A State of the Art

This section discusses and analyzes the current applications of gamification mechanics to GUI testing. The presented results are the outcome of a semi-systematic literature review that was performed on the Google Scholar dataset, by searching for the keywords *gamification* (or *ludicization*, or *gamified*), *software*, and *testing*. We then applied a set of inclusion and exclusion criteria, which are reported in Tables 10.1 and 10.2, to the results of this first search, resulting in a total of 43 sources. The execution of a process of backward snowballing on these sources, where we applied the inclusion and the execution criteria to new research items

Table 10.1 Inclusion criteria for the semi-systematic literature review

Inclusion criteria	Description
IC1	The source is directly related to the topic of gamification applied to the field of software testing and generally defined for the software engineering discipline but with clear applicability to the testing activity or explicitly proposes, discusses, improves, or applies an approach, a framework, or a prototype related to the gamification of any facet of software testing, including education of software testing subjects
IC2	The source addresses the topics covered by the review questions
IC3	The literature item is written in a language that is directly comprehensible by the authors: English, Italian, or Chinese
IC4	The source is an item of white literature with the full text available for download and is published in a peer-reviewed journal or conference proceedings

Table 10.2 Exclusion criteria for the semi-systematic literature review

Exclusion criteria	Description
EC1	The source is not directly related to the topic of gamification applied to the field of software testing
EC2	The source is not in a language directly comprehensible by the authors
EC3	The source is an item of white literature, but the full text is not available for download or online reading
EC4	The source discusses a serious/applied game that cannot be applied to real use case scenarios of the software testing process or software testing education and training
EC5	The source is not a primary study but a secondary or tertiary study of the topic
EC6	The source has not been published between 2010 and 2021

found, followed by a similar process of forward snowballing, resulted in a total of 50 different literature sources about gamified software testing, which we have listed in the online Appendix A.[1] An analysis of this literature review has shown that there is a significant lack of gamified testing tools that focus on exploratory GUI testing: more precisely, the only works we have found that have GUI testing as a focus are the previous works we have based this framework on. We conclude that this is still an unexplored field of research, and we feel that our work can be considered a novelty; we hope that our framework can inspire new research works in the field of gamified GUI testing.

All the collected sources were analyzed to collect the following information, discussed in the subsections below: (i) the most frequently adopted mechanics and tools for gamified software testing and (ii) the benefits and drawbacks of such techniques as discussed in related literature.

[1] https://doi.org/10.6084/m9.figshare.21967361.v1.

10.3.1 Adopted Game Mechanics

To assess the adoption and diffusion of game mechanics in gamified software testing literature, all the mechanics mentioned in the selected literature underwent a process of *coding*. The codes used were the names of the mechanics in the Octalysis framework. By this analysis, 24 different mechanics of the Octalysis framework were found in the related literature.

In the following, we report the gamification mechanics that had a number of mentions above the average:

- *Score*: The mechanic belongs to the *Accomplishment* core drive of the Octalysis framework. It implies that the user can earn virtual points after performing specific actions in the gamified system. The score can be used for other game mechanics (e.g., for buying virtual goods). A scoring system is often considered a fundamental building block for a gamified system, as many other dynamics that directly stimulate emotion rely on that. This mechanic was implemented or discussed in 33 different sources.
- *Leaderboards*: The mechanic belongs to the *Accomplishment* core drive of the Octalysis Framework. It consists of comparisons and rankings between the scores obtained by different users of the gamified system. The correlation between score and leaderboard is clear: the former is a mechanic that does not produce emotional value in the user until a competition dynamic, the leaderboard, is stimulated. The mechanic was implemented or discussed in 27 different sources.
- *Levels*: The mechanic belongs to the *Unpredictability* core drive of the Octalysis framework. They consist of the progression of the player through different stages with different objectives. The mechanic was implemented or discussed in 16 different sources.

The complete set of mechanics proposed, along with their definitions, related Octalysis core drive, and mentions in the related literature, is reported in the online Appendix B of the manuscript.[2]

In Fig. 10.1, we report the number of mentions for each gamification core drive defined in the Octalysis Framework. From the graph, it is evident how the main focus of available gamification implementations for software testing is to provide *Accomplishment* to the users as a positive means of motivation (43 mentions). Conversely, only two sources implemented mechanics related to the *Avoidance* dimension, which is related to the enforcement of correct patterns by applying punishments and maluses to non-conforming users. Few mentions were also gathered by the gamification mechanics related to the *Epic Meaning* macro-category of the Octalysis framework. We consider such a low number of mentions as an effect of the still prototypical nature of most of the described tools, which did not allow for the implementation of complex narratives.

[2] https://doi.org/10.6084/m9.figshare.20425446.

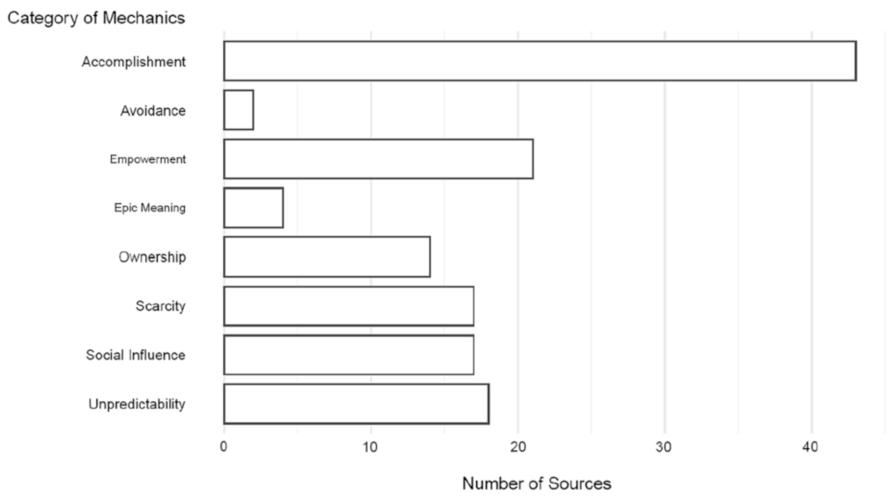


Fig. 10.1 Distribution of mentions in the literature for the core drives of the Octalysis framework

10.3.2 Gamified Software Testing Tools

In the mined set of literature about gamified software testing, we identify 27 different tools and/or frameworks for gamified software testing. In the online Appendix C,[3] we report the full list and tools, providing for each of them a brief description, the adopted mechanics (regarding the Octalysis framework), and the list of literature sources mentioning them.

It is evident from the list of tools that an important focus in gamified software testing literature is put on software testing education, as gamified mechanics are seen as a primary means of increasing the student's engagement in learning software testing topics. Several gamified tools aimed at practitioners' implementation of crowd-based mechanisms, to obtain higher coverage and effectiveness (i.e., detected bugs), by generating competition between different testers.

In the following, we report the most mentioned gamified testing tools in related literature. For each tool, we report the primary source where it has been described and the mechanics that it implements according to the Octalysis framework ; additionally, we present Table 10.3, where we list each tool and the gamified mechanics employed by said tool.

- *CodeDefenders* is a turn-based mutation testing game, in which two players are involved in competitive rounds. One player plays as the attacker, with the objective of injecting faults into the software, and the other plays as the defender, with the objective of writing test cases to spot faults. The tool has been originally

[3] https://doi.org/10.6084/m9.figshare.20425491.

Table 10.3 Gamified testing tools and their gamified mechanics

Tool	Gamified elements
CodeDefenders	Duels, scores, leaderboards, puzzles, feedback, challenges
HALO	Social interaction, quests, storytelling, achievements, levels, leaderboards
VU-BugZoo	Trophies, scores
WReSTT-Cyle	Score, leaderboards, badges, timing, levels, rewards, social interaction, quizzes
Auction-Based Bug Management	Auctions, virtual goods, timing, badges, leaderboards

described by Rojas and Fraser [24] and has originally been used to teach mutation testing in an academic context. In the following years, the tool received further developments introducing more features, along with several related experiments, which enriched the existing literature with experience reports of Code Defenders usage. The tool implements the following gamification mechanics: duels, scores, leaderboards, puzzles, feedback, and challenges.

- *HALO* is a plugin for Eclipse proposed by Sheh et al. that uses game-like mechanics to make the whole software engineering process more engaging and social [25, 26]. The tool implements an MMORPG-like (Massive Multiplayer Online Role Playing Game) approach to software testing activities. It has been used as the basis for the *Secret Ninja* approach proposed by Kiniry and Zimmerman, in which the gamification aspects are applied, while the users are not aware of their application [27]. The tool implements the following gamified mechanics: social interaction, quests, storytelling, achievements, levels, and leaderboards.
- *VU-BugZoo*, originally described by Silvis-Cividjian et al. [28], is an educational digital platform to teach software testing, based on a repository of faulty (stand-alone and embedded) code. The platform engages instructors and learners in a bug-hunting experience, which is empowered by the utilization of mechanics typical of game design. The tool implements the trophy and score gamification mechanics.
- *WReSTT-Cyle* (Web-Based Repository of Software Testing Tools Cyber-Enabled Learning Environments) is a cyber-learning environment that employs several learning and engagement strategies in order to aid the phase of software testing learning. It has been originally described by Clarke et al. [29] as a repository of learning objects to support software testing teaching and has then evolved into different projects named SEP-CyLe (Software Engineering and Programming) and STEM-CyLe (an extension to all STEM disciplines). Originally, the tool covered simple white and black-box unit testing. The cyber-enabled learning environment adopts the following gamification mechanics: score, leaderboards, badges, timing, levels, rewards, social interaction, and quizzes.
- *Auction-Based Bug Management*: originally described by Usfekes et al. [26], it is a serious game for bug tracking in Application Lifecycle Management Tools. The tool is based on an auction reward mechanism, with the aim of providing an

incentive structure for software practitioners to find, resolve, and test bugs and malfunctionings of a given SUT. The tool implements the following gamified mechanics: auctions, virtual goods, timing, badges, and leaderboards.

10.3.3 Advantages and Drawbacks of Gamification for Software Testing

On the set of literature items about the gamification of software testing activities, we applied a procedure of *coding* to extract categories of benefits and drawbacks caused by the adoption of gamified mechanics in testing procedures. After the coding was performed, we applied *axial coding* to extract higher-level categories of benefits and drawbacks. We report the full list of advantages and issues extracted from the literature in online Appendix D[4] of this chapter.

The main categories of advantages discussed in related literature are the following:

- **Better User Experience**: Under this category, we include all the benefits related to an increased quality of the user experience provided to the tester when gamified mechanics are applied. Twenty sources underline a higher *Engagement* (or *involvement*) guaranteed by game elements in the testing activity, with quantitative empirical results reported by Clegg et al. for unit testing [30, 31].

  Fifteen different sources highlighted that an important benefit of having game aspects in testing procedures is the guarantee of having more *fun* activities. This aspect was highlighted especially in the educational context [32].

  Finally, several gamification mechanics have been proven to provide additional *motivation* to the testers involved.
- **Higher Efficiency**. *Efficiency*, as defined by the ISO 9001 standard, is *the extent to which time, effort or cost is well used for the intended task or purpose* [33]. Under this category, we include advantages related to reduced efforts and costs in test case definition, generation, or execution caused by the application of gamified mechanics. One of the most positively commented aspects of gamified tools is the presence of *informative content* in the testing practices, which is able to reduce the effort required by the testers to gather information during the test cases design [34]. Several sources consider gamification a means to reduce the required effort to perform test-related activities. *Crowdsourced contributions* are mentioned in several sources and are seen as a primary mean to boost the efficiency of testing procedures [35].
- **Higher Effectiveness**. *Effectiveness*, as defined by the ISO 9001 standard, is *the extent to which planned activities are realised and planned results are achieved* [33]. Under this category, we include all the discussed advantages

[4] https://doi.org/10.6084/m9.figshare.20456574.

related to an enhancement of the outcomes of gamified testing procedures. Increased effectiveness is measured both in testing education (*improved learning*, measured through analysis of grades, [31]) and in testing practice (e.g., increased branch coverage and mutation score in gamified mutation testing, as measured by Fraser et al. [36]). Other specific effectiveness-related aspects are mentioned in other sources, e.g., effectiveness in finding bugs, identifying code smells, finding issues, and adding comments to the original code.

The main categories of disadvantages discussed in related literature are the following:

- **Design Issues**. Several studies in the literature report issues related to how the gamified aspects are designed and fit the underlying testing activities. For instance, Garcia et al. and Bryce et al. report cases where gamification interferes with the testing activities, being incompatible with other improvement efforts applied to the testing practice [37, 38]. Five of the collected studies highlight the necessity of proper calibration of gamification mechanics, to disincentivize possible exploits from the users to gain benefits.
- **Implementation issues**. A limited number of the selected sources mention implementation-related issues for gamified tools. The main concerns in this sense are related to the scalability of gamified approaches, especially the possibility to extend successfully to multiple players the game mechanics that are evaluated on a limited number of subjects [35]).
- **Bad user experience**. Some reports in the literature indicate that gamification can make the learning curve for testing procedures steeper [39]. Harranz et al. also report the possibility of *change resistance* for organizations to transition to gamified procedures [40].
- **Lower effectiveness**. Several sources in the literature report failing attempts at increasing the effectiveness of testing procedures through gamification, both in terms of the bug-finding ability of generated test cases and in learning outcomes [41].
- **Lower efficiency**. By increasing the concepts that the tester has to learn, gamification –when not properly designed and applied– can actually add overhead to the testing procedure. Pedreira et al. underline that setting up gamified work environments never has a negligible cost [42]. de Jesus et al. report that gamified environments are inherently complex and require incremental and constant efforts to be built [43].

In Fig. 10.2, we report the number of manuscripts in the set of analyzed literature, which mention at least one advantage or drawback for each of the defined categories. From the number of mentions, it is evident that the main focus of literature about gamified software testing has a primary focus on the advantages or the drawbacks that are caused to the user experience of the tools and techniques.

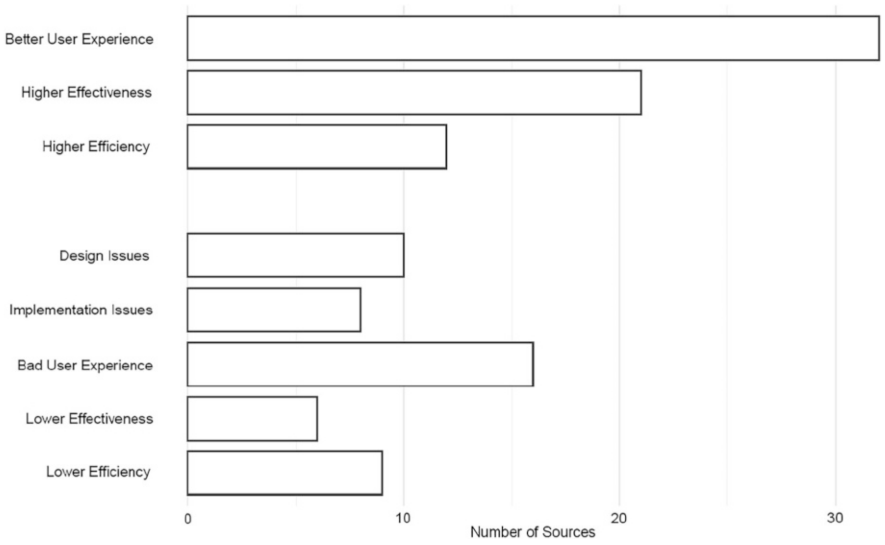


Fig. 10.2 Number of mentions in literature for each category of advantages and limitations

10.4 A Framework of Game Mechanics for GUI Testing of Web Apps

In this section, we describe and discuss a framework of gamified mechanics that can be applied to GUI Web application testing. To the best of our knowledge, the framework constitutes the first set of gamified elements specifically tailored for the practice of GUI testing. Even if some of the mechanics can be applied to other testing methodologies (e.g., to unit or mutation testing tools), most of the mechanics are specifically intended to aid the definition of second-generation GUI test cases through the record and replay methodology.

We have defined two preliminary prototypes of our framework of gamified mechanics: (i) an implementation as a plug-in for the Scout *augmented testing* tool, originally presented by Nass et al. [44], and (ii) a plug-in for the Chrome browser, to enable in-browser generation of test suites for Web applications.

Even though they are specifically implemented for Web application testing, the gamified mechanics in the framework are generalizable to GUI testing applied to any software domain (i.e., they can be adapted with little effort to mobile and desktop applications).

Figure 10.3 reports the Octalysis analysis performed for the proposed framework, by assigning one point to each dimension to which the gamified mechanics belong. The following subsections describe the individual gamification mechanics implemented.

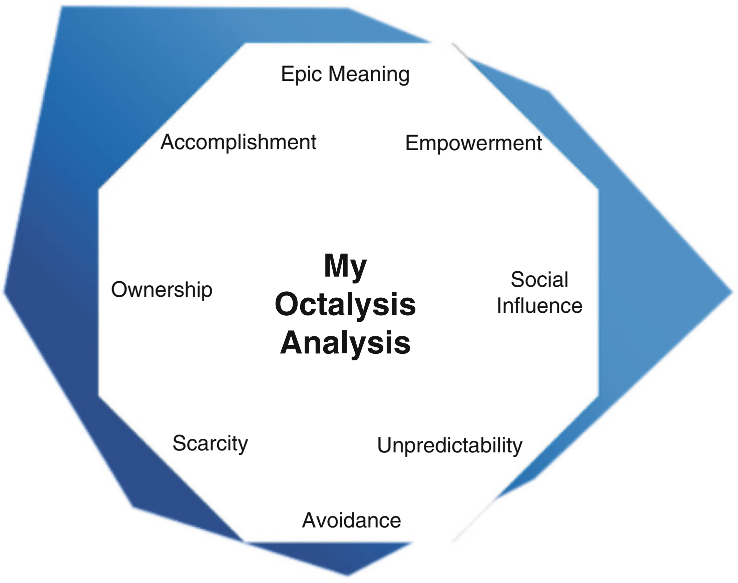


Fig. 10.3 Octalysis score for the proposed framework

10.4.1 Scoring Mechanism and Leaderboard

The principal gamification element of the proposed gamification framework is a formula to assign a score to each testing session performed by the tester. The score allows for evaluating the tester's performance, taking into account different factors, while introducing a competitive aspect that may encourage testers to put more effort into their activities.

The score is composed of two parts: a *base score*, which takes into account factors that can be used to compare different GUI test sequences on the same working application, and a *bonus score*, which takes into account the bug-finding capability of the developed test suites. As such, the score is calculated by using the following formula:

$$S = S_{base} + S_{bonus}$$

The base score is computed by using the following formula:

$$S_{base} = a \cdot C + b \cdot EX + c \cdot EF$$

The base score adds up to 100 points and is composed of three subcomponents weighted by configurable parameters:

- **Coverage component** (C): the average page coverage obtained by the tester during the session, according to the formula:

$$C = \frac{\sum\limits_{\forall i \in P} cov_i}{|P|}$$

  where cov_i is the coverage of the i-th page and P is the set of pages visited during the session. This component is multiplied by default for a coefficient equal to 60%.
- **Exploration component** (EX): a component depending on the percentage of pages visited and widgets interacted for the first time by the current tester. It is therefore computed according to the following formula:

$$EX = \frac{k}{b} \cdot \frac{p_{new}}{p_{tot}} + \frac{h}{b} \cdot \frac{w_{new}}{w_{tot}}, \qquad k + h = b$$

  where p_{new} and p_{tot} are the newly discovered and the total pages, respectively, while w_{new} and w_{tot} are the newly discovered and the total interacted widgets, respectively. By default, this component is worth 30% of the total base score.
- **Efficiency component** (EF): it is computed as the ratio between the number of interacted widgets and the total number of interactions, thereby according to the formula:

$$EF = \frac{w_{hl}}{w_{int}}$$

  By default, this component is worth 10% of the base score. This component aims at measuring the diversity of interactions performed by the tester to avoid exploitations of the scoring mechanism (Fig. 10.4).

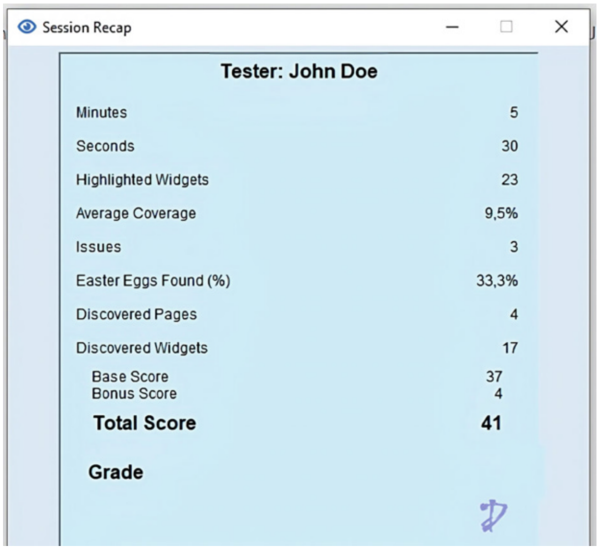
  The bonus score is computed by using the following formula:

$$S_{bonus} = d \cdot T + e \cdot P$$

  The base score adds up to 50 points and is composed of two subcomponents weighted by configurable parameters:

- **Time component** (T): it is computed on top of the duration of the test session. The rationale for the utilization of a time component is that longer test sequences should allow a more thorough exploration of the GUI. The time component is computed as follows:

$$T = \begin{cases} 0 & s_{int} \leq 2 \ \vee \ s_{int} > 30 \\ 1.5 \cdot t & 2 < s_{int} \leq 5 \\ t & 5 < s_{int} \leq 15 \\ 0.5 \cdot t & 15 < s_{int} \leq 30 \end{cases}$$


Session Recap

Tester: John Doe

Minutes	5
Seconds	30
Highlighted Widgets	23
Average Coverage	9,5%
Issues	3
Easter Eggs Found (%)	33,3%
Discovered Pages	4
Discovered Widgets	17
Base Score	37
Bonus Score	4
Total Score	**41**
Grade	

Fig. 10.4 Results screen, showing the metrics measured for the session and the related score

where t is the duration of the session (in minutes), while s_{int} is the average time spent per interaction (measured in seconds). By default, this component is multiplied by 0.3.

- **Problems component** (P): it is computed on top of the number of issues reported by the tester during the exploration of the SUT. The default coefficient of this component is equal to 0.2.

10.4.2 Progress Bars

The progress bar is a form of live graphical feedback, which shows the number of widgets that have been interacted with by the tester during the exploration of the SUT.

The progress bar is rendered so that a global progress bar shows the percentage of widgets interacted with by the user, in relation to the total amount of widgets present on the page.

For the pages that have already been explored by at least another tester, a blue line is shown on top of the progress bar to indicate the *highest score* in the page, i.e., the maximum coverage reached on such page among all past test sessions.

This element is aimed at providing satisfaction for the progress made by the testers.

10.4.3 Exploration Highlights

Exploration highlights are a form of live feedback employed to notify that a new page has been discovered by the current tester, i.e., no previous test session has ever visited it.

This element allows informing the tester when their exploration of the SUT is better –in terms of novelty– than previous testers' ones.

10.4.4 Injected Bugs

Injected bugs are visual modifications that are injected into the AUT's GUI. At the current state of implementation of the prototype, injected bugs are represented as superimposed oval-shaped visual elements, placed over randomly chosen elements in a set of pages of the SUT. The purpose of injected bugs is to encourage the tester to explore the Web application by visiting as many pages as possible while providing an immediate operation to perform in addition to only recording valid operations over a properly working SUT.

In Fig. 10.5, the three visual elements discussed are visible: the progress bar on top of the screen, an exploration highlight in the top-left corner, and an injected bug in the center of the page.

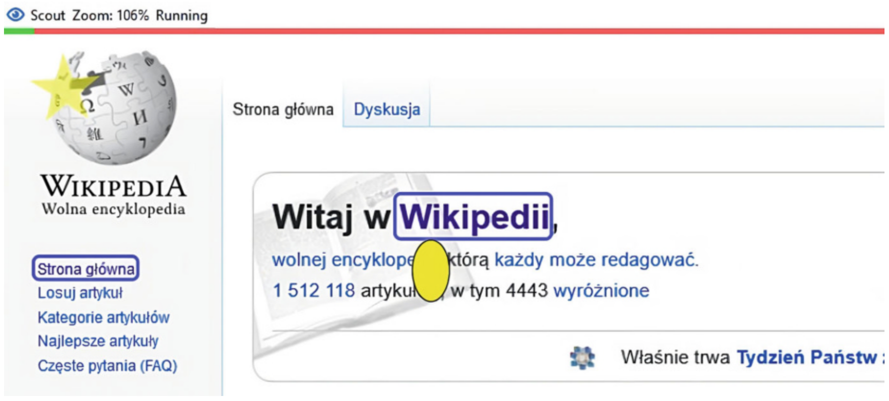


Fig. 10.5 Visual elements shown during the exploration of the SUT: progress bar, exploration highlights, injected bugs

10.4.5 Achievements

Achievements are used to provide a visual certification that the tester has met specific objectives or requirements.

Achievements are shown through graphical *badges*. The purpose of achievements is to provide gratification to the tester.

Gained achievements are shown as a form of live feedback during the test session in which they are gained. It is possible for a tester to visualize the whole set of achievements gained and those that are still missing.

10.4.6 User Profiles and Avatars

The mechanic allows defining a profile for each tester registered in the system. The tester can then customize the profile with the preferred avatar, choosing between a set of available ones. Additional items are unlockable by utilizing an in-game virtual currency, which the tester is given every time he unlocks achievements or completes quests or objectives.

Avatars belong –as a mechanic– to the *Ownership* core drive in the Octalysis framework and are based on the human need to empower their presence and properties.

Each player profile is associated with a certain amount of experience points. As in many role-playing game-based gamification mechanics, experience points allow to increase the player's *level* and to unlock additional features or customization items when certain levels are reached.

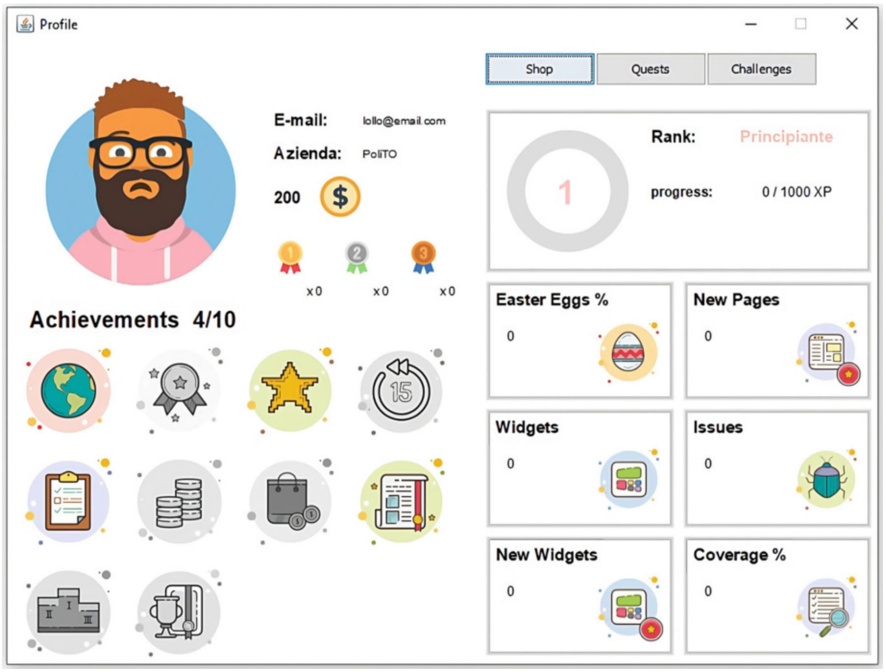
In Fig. 10.6, the profile page of the tester is reported. The profile page shows the currently selected avatar, the item shop (where to invest virtual currency to customize the avatar), and the unlocked achievements.

10.4.7 Quests and Challenges

Finally, the gamification framework includes two additional mechanics aimed at encouraging the tester to execute specific actions.

Quests are specific tasks to perform during the execution of testing sequences. They can be tied to specific types of interactions and elements or to the number of pages visited. The framework considers two different types of quests: daily quests, which are available for a single day, and questlines, proposing a set of predefined quests of increasing difficulty. Figure 10.7 shows both types of quests implemented in the framework.

Challenges are specific tasks to be completed that are available only for a set period of time (e.g., for a week). The score of all participating testers is registered



E-mail: lollo@email.com
Azienda: PoliTO
200 $
Achievements 4/10
Rank: Principiante
progress: 0 / 1000 XP
Easter Eggs % 0
New Pages 0
Widgets 0
Issues 0
New Widgets 0
Coverage % 0

Fig. 10.6 Profile page of the tester, with the chosen avatar, the achievements' badges, and the avatar shop

for each challenge, and a special leaderboard for the testers competing in a challenge is created. Prizes can be awarded to the testers that achieve a good placement in the challenge.

Although real-world rewards are indeed a successful extrinsic motivator based on tangible feedback (mostly monetary prizes), a plain usage not supported by a balanced gamified experience would result unappealing and unsustainable in the long term. Providing monetary incentives is an extrinsic motivator, the adoption of which only keeps the user effectively involved in the short term (what is defined as *left-brain* in the Octalysis core drives).

Providing an intrinsic motivating factor such as unpredictability (in the shape of daily quests) is the way we aim at keeping testers engaged with a balanced experience while attempting to mitigate the over-justification effect, whereby the effect of an extrinsic motivator diminishes as the user becomes accustomed to it.

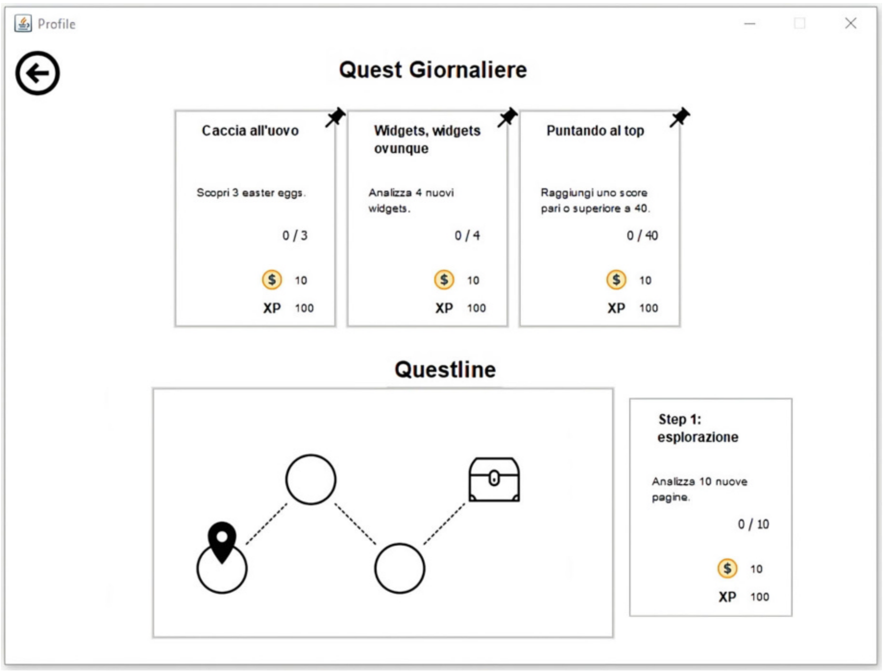


Fig. 10.7 Page displaying quests and challenges, with a section for daily challenges and one for the entire questline, each one having its set of rewards

10.5 Preliminary Evaluation

After the definition of the framework, we performed a preliminary assessment to evaluate the effectiveness and efficiency of the proposed gamified GUI testing tool.

To that extent, we performed a 2 treatments $\times$ 2 sequences $\times$ 2 objects full factorial (crossover) experiment. The treatment of the experiment was administered as two different versions of the tool: the *gamified* version (i.e., implementing our framework) and the *standard* version (i.e., the original version of the Scout tool for exploratory GUI testing of Web applications).

The experiment involved 144 participants recruited through convenience sampling among students enrolled in the Software Engineering course held at the Polytechnic University of Turin in the Spring semester of 2021. All participants received both treatments, and received two different tasks to perform, each with a specific subject application. All participants were provided with the task of generating test cases for two different Web-based SUTs, manually and utilizing the Scout tool. The applications were selected randomly from a list of open-

Table 10.4 Experiment design

	Period 1	Period 2
	Object:Treatment	Object:Treatment
Group 1	Mezzanine:Gamified	Wagtail:Standard
Group 2	Mezzanine:Standard	Wagtail:Gamified
Group 3	Wagtail:Gamified	Mezzanine:Standard
Group 4	Wagtail:Standard	Mezzanine:Gamified

source applications available in grey-literature.[5] The selected applications were *Mezzanine*[6] and *Wagtail*.[7] The experiment design is reported in Table 10.4.

In our evaluation we focused on three different aspects of the testing practice: effectiveness, efficiency and User Experience. To evaluate *Effectiveness*, we injected artificial bugs in the two experimental objects, and we measured the number of *True Positives*, i.e. bug reports provided by the testers that corresponded to bugs injected in the SUT. To evaluate *Efficiency*, we measured the *coverage* (i.e., the ratio between analyzed widgets and total widgets in the traversed Web pages) provided by the generated test cases during the test sequences recorded by the participants. At the end of the experimental sessions, the participants were administered the TAM questionnaire [45], to evaluate the User Experience of the tool. The full TAM questionnaire is reported in an online Appendix E.[8]

Figure 10.8 shows the boxplots for the distribution of average coverage per page and the total number of true positives found in each session, aggregated by treatment. Numeric results are shown in Table 10.5.

The results suggest that the gamified version of the tool achieves higher mean coverage (9.9% against 8.3%), whereas the number of true positives detected was both slightly higher for the standard version of the tool on average (2.75 vs. 2.58).

These results suggest that the inclusion of gamified mechanics primarily caused the participants to focus more on their exploratory testing. Hence, each application page was tested more thoroughly with gamified mechanics in place. We guess that the primary explanation for this behavioral change is connected to the visual highlight features that were added to the tool.

In the experiment, we verified that gamification had a negative (albeit not significant) impact on the defect-finding ability of the testers. We guess that the slightly lower number of defects found on average can be justified by the cognitive overhead introduced by the gamification concepts.

Figure 10.9 reports the distribution of the answers to the TAM Questionnaire. In the graph, we report the mean of the answers for each category of the questionnaire (i.e., Attitude toward Usage or *ATU*, Perceived Usefulness or *PU*, Behavioral Intention or *BI*, Perceived Ease of Use or *PE*).

[5] https://github.com/unicodeveloper/awesome-opensource-apps.

[6] https://github.com/stephenmcd/mezzanine.

[7] https://github.com/wagtail/wagtail.

[8] https://doi.org/10.6084/m9.figshare.20456496.

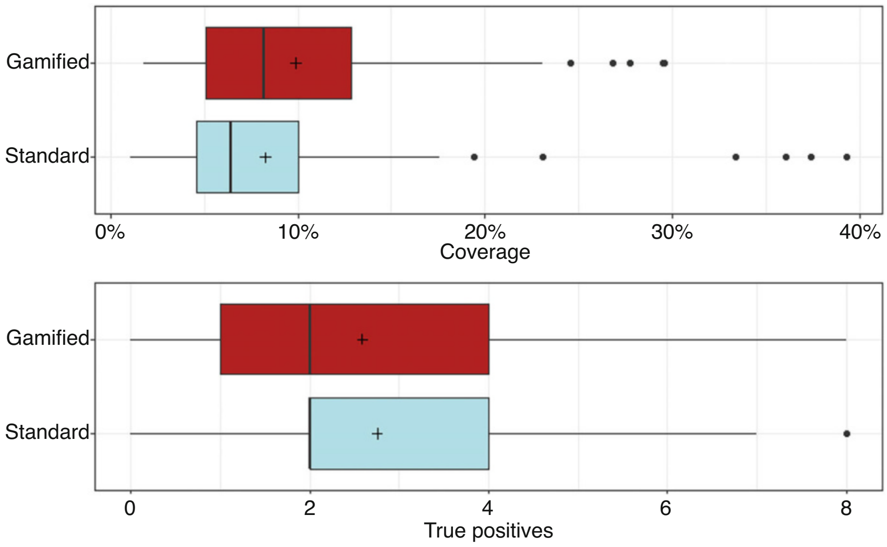


Fig. 10.8 Boxplots for the two metrics observed and measured during the experiment, average coverage over pages (percentage) and true positives found

Table 10.5 Statistics for coverage and true positives

		Mezzanine		Wagtail		All	
		S	G	S	G	S	G
Coverage	Mean	8.0%	8.9%	8.6%	10.8%	8.3%	9.9%
	Median	6.2%	7.4%	6.5%	8.8%	6.5%	8.1%
	Std. dev	6.8%	6.0%	5.5%	6.5%	6.2%	6.3%
True positives	Mean	1.87	1.57	3.68	3.58	2.75	2.58
	Median	2.00	1.00	1.00	4.00	2.00	2.00
	Std. dev	1.23	1.06	1.57	1.68	1.67	1.72

We observe, on average, that participants had mostly positive perceptions toward all the metrics measured by the TAM model. The metric with the most positive responses was the Attitude toward Usage metric, with 17% of participants strongly agreeing and 66% agreeing that gamification is a desirable addition to the practice of exploratory testing. A high value for the Attitude toward Usage can be considered as a consequence of a positive User Experience of the users in their sessions with the gamified tool. High positive perceptions were also measured for Perceived Usefulness and Behavioral Intention. These results suggest that most of the participants would use tools with gamification if they had to perform testing activities in the future. Additionally, they indicate that gamification was perceived as valuable and usable. The values for Behavioral Intention and Perceived Usefulness suggest that the gamified mechanics and aims were easily understandable by the users.

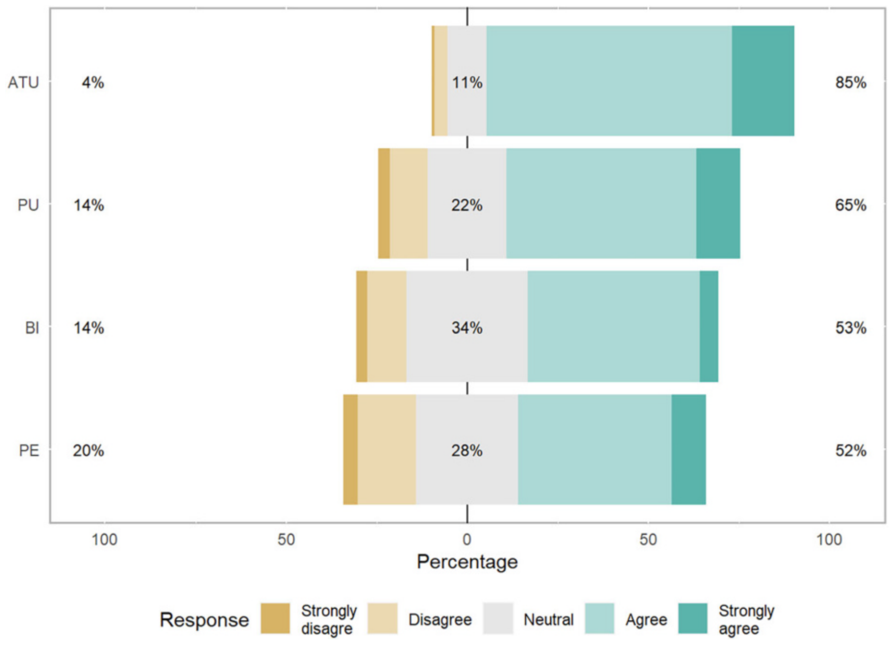


Fig. 10.9 Distribution of the answers (percentage of the responses for each level of the respective Likert scale) provided by the participants to the four categories of the TAM questionnaire: ATU (Attitude Toward Usage), PU (Perceived Usefulness), BI (Behavioral Intention), and PE (Perceived Ease of Use). The results are averaged over all the questions of each category

The most negative perceptions were aimed toward Perceived Ease of Use (16% Disagree, 4% Strongly disagree). This result, however, can be justified by the lack of experience of the participants with the practices and the inherently low ease of use of the specific tool that was extended with gamified mechanics.

10.6 Threats to Validity

The potential threats to the framework's validity are discussed according to the categories defined by Wohlin et al. [46].

Threats to internal validity concern factors that may affect the results and were not considered in the study. There is no guarantee that the measures selected to evaluate a testing session (coverage, time spent) are the most optimal ones for this purpose: a systematic literature review by Coppola and Alégroth [47] identified 55 different metrics belonging to 4 categories of GUI-based testing: functional-level metrics, GUI-level metrics, model-level metrics, and code-level metrics. With this many possible metrics present in the literature for GUI testing, it is not possible to say that the metrics we used for our frameworks are the most beneficial or effective; future experiments and studies are going to be required in order to be able to correctly gauge whether the selected metrics are effective or if they can be replaced with other ones.

Threats to construct validity concern the relationship between theory and observation. The framework defines a set of gamified mechanics that have been proven to be effective in multiple studies, but it cannot be assumed that the combination of said mechanics will prove effective for a GUI testing tool, as there is currently no study on gamification of this specific practice. Future experiments are going to be necessary to evaluate whether usage of the tool can lead to improved GUI testing practices, as well as increased interest and motivation for the testers; these experiments will have to evaluate the framework in its entirety as well as the single gamified elements, to identify eventual weak links.

Threats to external validity concern whether the results can be generalized. We cannot affirm for certain that the tool will prove to be effective for all cases of exploratory GUI testing of Web applications, mainly due to two factors: the selection of gamified mechanics, which may appear to be effective on paper but then prove itself not optimal when applied to real-world scenarios, and the absence of other studies and experiments on gamified GUI testing that can be compared to the framework. The fact that this framework consists of, to the extent of our knowledge, a novelty for the current literature means that we cannot consider our choices and methods to be generalized for all possible facets of exploratory GUI testing. There is also the risk of having selected gamified elements that cannot be applied to every possible kind of Web application or to different domains; different Web development strategies may not interact correctly with the framework, for example, and this means that the defined strategy cannot be generalized to the entire field of GUI testing for Web applications.

10.7 Conclusion and Future Directions

In this chapter, we have described a framework of gamified mechanics to be applied to exploratory GUI testing of Web applications. To the best of our knowledge, the framework constitutes the first effort in adapting gamification mechanics to GUI testing. We complement the definition of the framework with a literature review of gamification applied to software testing, to provide a view of the current state of the art about gamified testing tools, along with the most utilized mechanics and the mentioned advantages and drawbacks of the technique.

The framework contains seven different mechanics. Three of them represent novel contributions with regard to the existing literature: a scoring mechanism specifically defined for GUI testing of Web applications and graphical feedback (exploration highlights and progress bars) that are specifically designed to follow the typical procedure of exploratory testing of Web-based SUTs.

We have implemented the mechanics in a prototype tool, developed as both a plug-in for an existing augmented testing tool (Scout) and as a plug-in for the

Chrome browser. The implementation of the framework allowed us to perform a preliminary evaluation, in the form of a controlled experiment with 144 participants.

The performed experimentation showed that gamified mechanics provide higher page coverage than non-gamified GUI testing. From a bug-finding standpoint, gamification has shown to have no significant beneficial or detrimental effects. Finally, gamification was considered to provide a good user experience. Our experiment thereby confirms that with no loss in effectiveness of the generated test sequences, gamification can enhance the user experience provided to software testers. A better user experience can lead to higher productivity, engagement, and quality of test cases produced by testers. The conduction of more well-structured empirical experimentations, including more metrics to verify effectiveness, efficiency, and user experience, and the consequent evaluation of the interaction between the different measures, will be crucial to provide a dependable assessment of the benefits provided by gamification. The current positive effects on coverage can in fact be considered as direct behavioral consequence of some introduced gamification mechanics (e.g., progress bar and visualization highlights) with negligible or even detrimental effects on the real quality of generated test cases with the gamified tool.

At the current state of implementation, the tool is still in a prototypal state, and it needs to be deployed on the tester's machine to work; in our future development, we foresee a distributed implementation that will allow the utilization of the tool with crowd-testing purposes and effective utilization of competitive mechanics (e.g., leaderboards).

Future research directions include the adoption of the tool in an educational context –e.g., a software engineering course– in order to conduct a longitudinal study to evaluate the effects of utilizing gamification when teaching system-level and GUI testing to students. Moreover, we believe it is important to evaluate the impact of the individual proposed mechanics in isolation.

References

1. Borjesson, E., Feldt, R.: Automated system testing using visual GUI testing tools: a comparative study in industry. In: 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, pp. 350–359. IEEE, Piscataway (2012)
2. Pedreira, O., García, F., Brisaboa, N., Piattini, M.: Gamification in software engineering–a systematic mapping. Inf. Softw. Technol. **57**, 157–168 (2015)
3. Mäntylä, M.V., Smolander, K.: Gamification of software testing-an MLR. In: International Conference on Product-Focused Software Process Improvement, pp. 611–614. Springer, Berlin (2016)
4. Rodrigues, L., Pereira, F.D., Toda, A.M., Palomino, P.T., Pessoa, M., Carvalho, L.S.G., Fernandes, D., Oliveira, E.H., Cristea, A.I., Isotani, S.: Gamification suffers from the novelty effect but benefits from the familiarization effect: findings from a longitudinal study. Int. J. Educ. Technol. High. Educ. **19**(1), 1–25 (2022)
5. Chou, Y.-k.: Actionable Gamification: Beyond Points, Badges, and Leaderboards. Packt Publishing Ltd, Birmingham (2019)

6. Hosseini, C., Humlung, O., Fagerstrøm, A., Haddara, M.: An experimental study on the effects of gamification on task performance. In: Procedia Computer Science 196 (2022) 999–1006, International Conference on ENTERprise Information Systems/ProjMAN – International Conference on Project MANagement/HCist – International Conference on Health and Social Care Information Systems and Technologies 2021. https://www.sciencedirect.com/science/article/pii/S1877050921023255
7. Wang, C., He, J., Jin, Z., Pan, S., Lafkihi, M., Kong, X.: The impact of gamification on teaching and learning physical internet: a quasi-experimental study. Ind. Manag. Data Syst. **122**, 1499–1521 (2022)
8. Jensen, M.L., Wright, R.T., Durcikova, A., Karumbaiah, S.: Improving phishing reporting using security gamification. J. Manag. Inf. Syst. **39**(3), 793–823 (2022)
9. Liechti, O., Pasquier, J., Reis, R.: Supporting agile teams with a test analytics platform: a case study. In: 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST), pp. 9–15. IEEE, Piscataway (2017)
10. Cacciotto, F., Fulcini, T., Coppola, R., Ardito, L.: A metric framework for the gamification of web and mobile GUI testing. In: 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 126–129 (2021)
11. Fulcini, T., Ardito, L.: Gamified exploratory GUI testing of web applications: a preliminary evaluation. In: 2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 215–222 (2022)
12. Alégroth, E., Gao, Z., Oliveira, R., Memon, A.: Conceptualization and evaluation of component-based testing unified with visual GUI testing: an empirical study. In: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), pp. 1–10. IEEE, Piscataway (2015)
13. Ardito, L., Bottino, A., Coppola, R., Lamberti, F., Manigrasso, F., Morra, L., Torchiano, M.: Feature matching-based approaches to improve the robustness of android visual GUI testing. ACM Trans. Softw. Eng. Methodol. **31**(2), 1–32 (2021)
14. Vasquez, M.L., Moran, K., Poshyvanyk, D.: Continuous, evolutionary and large-scale: a new perspective for automated mobile app testing. Preprint. arXiv:1801.06267
15. Coppola, R., Morisio, M., Torchiano, M.: Mobile GUI testing fragility: a study on open-source android applications. IEEE Trans. Reliab. **68**(1), 67–90 (2018)
16. Memon, A.M., Cohen, M.B.: Automated testing of GUI applications: models, tools, and controlling flakiness. In: 2013 35th International Conference on Software Engineering (ICSE), pp. 1479–1480. IEEE, Piscataway (2013)
17. Kamran, M., Rashid, J., Nisar, M.W.: Android fragmentation classification, causes, problems and solutions. Int. J. Comput. Sci. Inf. Sec. **14**(9), 992 (2016)
18. Nass, M., Alégroth, E., Feldt, R.: Why many challenges with GUI test automation (will) remain. Inf. Softw. Technol. **138**, 106625 (2021)
19. Kochhar, P.S., Thung, F., Nagappan, N., Zimmermann, T., Lo, D.: Understanding the test automation culture of app developers. In: 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), pp. 1–10. IEEE, Piscataway (2015)
20. Krutz, D.E., Malachowsky, S.A., Reichlmayr, T.: Using a real world project in a software testing course. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education, pp. 49–54 (2014)
21. Barreto, C.F., França, C., Gamification in software engineering: a literature review. In: 2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pp. 105–108. IEEE, Piscataway (2021)
22. de Melo, A.A., Hinz, M., Scheibel, G., Berkenbrock, C.D.M., Gasparini, I., Baldo, F.: Version control system gamification: a proposal to encourage the engagement of developers to collaborate in software projects. In: Meiselwitz, G. (ed.) Social Computing and Social Media, pp. 550–558. Springer International Publishing, New York City (2014)
23. Ašeriškis, D., Damaševičius, R.: Gamification of a project management system. In: The Seventh International Conference on Advances in Computer-Human Interactions, pp. 200–207 (2014)

24. Parizi, R.M.: On the gamification of human-centric traceability tasks in software testing and coding. In: 2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA), pp. 193–200 (2016)
25. Ruohonen, J., Allodi, L.: A bug bounty perspective on the disclosure of web vulnerabilities. ArXiv abs/1805.09850
26. Üsfekes, Ç., Tüzün, E., Yilmaz, M., Macit, Y., Clarke, P.M.: Auction-based serious game for bug tracking. IET Softw. **13**, 386–392 (2019)
27. Bell, J., Sheth, S., Kaiser, G.: Secret ninja testing with halo software engineering. In: Proceedings of the 4th International Workshop on Social Software Engineering, SSE '11, pp. 43–47. Association for Computing Machinery, New York (2011)
28. Silvis-Cividjian, N., Limburg, R., Althuisius, N., Apostolov, E., Bonev, V., Jansma, R., Visser, G., Went, M.: Vu-bugzoo: a persuasive platform for teaching software testing. In: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '20, p. 553. Association for Computing Machinery, New York (2020)
29. P.E., Y.F., Clarke, P.J.: Gamification-based cyber-enabled learning environment of software testing. In: 2016 ASEE Annual Conference & Exposition, ASEE Conferences, New Orleans, Louisiana, pp. 1–16 (2016). https://peer.asee.org/27000
30. Clegg, B.S., Rojas, J.M., Fraser, G.: Teaching software testing concepts using a mutation testing game. In: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET), pp. 33–36 (2017)
31. Sun, Y.: Design and implementation of a gamified training system for software testing. Adv. Educ. **10**, 395–400 (2020)
32. Fraser, G., Gambi, A., Kreis, M., Rojas, J.M.: Gamifying a software testing course with code defenders. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE '19, pp. 571–577. Association for Computing Machinery, New York (2019)
33. I. 9001:2005, Quality management systems – requirements, Standard, International Organization for Standardization (2005)
34. Lőrincz, B., Iudean, B., Vescan, A.: Experience report on teaching testing through gamification. In: EASEAI 2021, pp. 15–22. Association for Computing Machinery, New York (2021)
35. Amiri-Chimeh, S., Haghighi, H., Vahidi-Asl, M., Setayesh-Ghajar, K., Gholami-Ghavamabad, F.: Rings: a game with a purpose for test data generation. Interact. Comput. **30**(1), 1–30 (2017)
36. Fraser, G., Gambi, A., Rojas, J.M.: Teaching software testing with the code defenders testing game: experiences and improvements. In: 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 461–464 (2020)
37. Bryce, R., Mayo, Q., Andrews, A., Bokser, D., Burton, M., Day, C., Gonzolez, J., Noble, T.: Bug catcher: a system for software testing competitions. In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13, pp. 513–518. Association for Computing Machinery, New York (2013)
38. García, F., Pedreira, O., Piattini, M., Cerdeira-Pena, A., Penabad, M.: A framework for gamification in software engineering. J. Syst. Softw. **132**, 21–40 (2017). https://www.sciencedirect.com/science/article/pii/S0164121217301218
39. Berkling, K., Thomas, C.: Gamification of a software engineering course and a detailed analysis of the factors that lead to it's failure. In: 2013 International Conference on Interactive Collaborative Learning (ICL), pp. 525–530 (2013)
40. Herranz, E., Colomo-Palacios, R., Al-Barakati, A.: Deploying a gamification framework for software process improvement: preliminary results. In: Stolfa, J., Stolfa, S., O'Connor, R.V., Messnarz, R. (eds.) Systems, Software and Services Process Improvement, pp. 231–240. Springer International Publishing, Cham (2017)
41. de Jesus, G.M., Paschoal, L.N., Ferrari, F.C., Souza, S.R.S.: Is it worth using gamification on software testing education? An experience report. In: Proceedings of the XVIII Brazilian Symposium on Software Quality, SBQS'19, pp. 178–187. Association for Computing Machinery, New York (2019)

42. Pedreira, O., García, F., Piattini, M., Cortiñas, A., Cerdeira-Pena, A.: An architecture for software engineering gamification. Tsinghua Sci. Technol. **25**(6), 776–797 (2020)
43. Jesus, G.M.d., Ferrari, F.C., Paschoal, L.N., Souza, S.d.R.S.d., Porto, D.d.P., Durelli, V.H.S.: Is it worth using gamification on software testing education? An extended experience report in the context of undergraduate students. J. Softw. Eng. Res. Dev. **8**, 6:1–6:19 (2020). https://sol.sbc.org.br/journals/index.php/jserd/article/view/738
44. Nass, M., Alégroth, E., Feldt, R.: Augmented testing: industry feedback to shape a new testing technology. In: 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), pp. 176–183. IEEE, Piscataway (2019)
45. Lee, Y., Kozar, K.A., Larsen, K.R.: The technology acceptance model: past, present, and future. Commun. Assoc. Inf. Syst. **12**(1), 50 (2003)
46. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer Science & Business Media, Berlin (2012)
47. Coppola, R., Alégroth, E.: A taxonomy of metrics for gui-based testing research: a systematic literature review. Inf. Softw. Technol. **152**, 107062 (2022). https://www.sciencedirect.com/science/article/pii/S0950584922001719

Chapter 11
Applying Leaderboards for Quality Improvement in Software Development Projects

Mathias Eggert, Philipp M. Zähl, Martin R. Wolf, and Martin Haase

Abstract Software development projects often fail because of insufficient code quality. It is now well documented that the task of testing software, for example, is perceived as uninteresting and rather boring, leading to poor software quality and major challenges to software development companies. One promising approach to increase the motivation for considering software quality is the use of gamification. Initial research works already investigated the effects of gamification on software developers and come to promising. Nevertheless, a lack of results from field experiments exists, which motivates the chapter at hand. By conducting a gamification experiment with five student software projects and by interviewing the project members, the chapter provides insights into the changing programming behavior of information systems students when confronted with a leaderboard. The results reveal a motivational effect as well as a reduction of code smells.

Keywords Software development · Software testing · Gamification · Leaderboard

11.1 Motivation

Software development projects are confronted with different stakeholders and different quality requirements. Particularly, when new business models are explored in a highly competitive environment, development speed is more important than quality, which can lower the attractiveness of products [1]. At the same time, additional quality requirements arise through the trend of mobile application development [2, 3]. It is now well documented that the task of testing software, for example, is perceived as uninteresting and rather boring, leading to poor software

M. Eggert (✉) · P. M. Zähl · M. R. Wolf · M. Haase
FH Aachen University of Applied Sciences, Aachen, Germany
e-mail: eggert@fh-aachen.de; zaehl@fh-aachen.de; m.wolf@fh-aachen.de; martin.haase@alumni.fh-aachen.de

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_11

quality [4–6] and currently posing major challenges to software development companies [7].

One promising approach to increase the motivation for considering software quality is the use of gamification, which involves the use of elements from video games to increase motivation for tasks in a non-game context [8]. Rather it considers the use of games or game elements in a serious context [9]. Game elements include awards, points, or medals, but also rankings and similar reward systems [8].

Gamification and gamified information systems (GIS) provide elements familiar from computer games to make every day mundane and often boring activities more attractive and engaging [9, 10]. Gamification can help increase the motivation and efficiency of routine tasks [11, 12, e.g., 8].

Initial research works already investigated the effects of gamification on software developers and come to promising results [13–15]. Nevertheless, a lack of results from field experiments exists, which motivates the article at hand. By conducting an experiment with five student software projects and by interviewing the project members, we want to shed light into to motivational effects of leaderboards applied by software developers to increase software quality.

The article provides both insights into the changing behavior at software development tasks, measured with the code quality checking software *Sonarcube*, and insights into the perceptions of software developers when faced with a leaderboard. The remainder of the chapter is as follows. The subsequent section summarizes relevant research works in the field of gamification applied for software developers. In addition, we clearly point out the research gap that is addressed by the work at hand. In Sect. 11.3, we describe the research method and the experimental setting. Section 11.4 comprises the results of the experiments and the interviews. Section 11.5 discusses the findings and provides an outlook on further research regarding gamification for software developers.

11.2 Related Work

11.2.1 Gamification

Precursors to the idea of gamification can be traced back to the boom in video games in the 1980s. However, the term gamification was first used almost 20 years later by the British game developer Nick Pelling in 2002 [vgl. 10, 16]. Since then, the term has been used in many ways, which means that various definitions of gamification are known today [16–18]. These definitions focus on user experience and improved motivation, as well as user retention. Thus, in non-game contexts, i.e., real-life tasks, the elements of gamification should be applied to the gamified task to achieve improvements in the areas mentioned. The most well-known and very comprehensively formulated definition comes yet from Deterding et al. [9] and also includes limitations that clearly show what constitutes gamification:

The use (rather than expansion) of design (rather than game-based technology or other game-related practices) elements (rather than full-fledged games) that exhibit characteristics of games (rather than play or playfulness) in non-game contexts (regardless of specific intended uses, contexts, or modes of implementation). [9]

The design of gamified applications is based on various elements that can also be found in conventional games. According to Deterding, S., Khaled, R., Nacke, L., and Dixon, D. [19], there are different levels of abstraction of these design elements. Ordered from concrete to abstract, they are user interface design patterns (e.g., badges, levels, leaderboards), design patterns/game mechanics, design principles/guidelines, conceptual models (MDA framework, imagination, curiosity), and game design methods (playtesting, etc.).

Widely used game design elements are therefore points, badges, and leaderboards. Points are distributed for achieving specific game objectives [20]. Badges are trophies that are visible to other players and are awarded when a milestone is reached in the game. They are not the goal of the game but complement it. They are awarded for special achievements [20, 21]. A leaderboard records the progress of all players, usually in the form of a high score [20]. While leaderboards are sometimes controversial, overall they can be said to improve motivation [22]. Other elements of gamification are progress bars, quests, or the story of the game [23].

Motivation is another important concept in the gamification approach. Motivation is divided into different types. On the one hand, there is extrinsic motivation. This is brought to the individual from the outside and is supported by rewards, for example [24, 25]. Social motivation can also be considered extrinsic [25]. Intrinsic motivation, on the other hand, arises within the individual and is caused solely by the inherent satisfaction of performing the activity in question [26].

11.2.2 Software Quality and Technical Debt

Software quality (SQ) is an expression that describes the qualitative characteristics of a software product. Not clearly defined, it has been made more tangible by various quality models, for example, ISO/IEC 25000. In this, one finds different areas that make up SQ: functionality, reliability, efficiency, usability, transferability, and modifiability [27].

Code quality (CQ) is a sub-aspect of SQ; it is the feature of SQ that is the focus of this thesis. CQ describes how well a source code satisfies formal requirements. One speaks also of programming style or code structure. This should, in terms of code created by students, be of balanced size, readable, understandable, well-structured, and not complex. There should also be a minimum of duplication or poorly formatted expressions [28].

Codesmells (CS) are particularly interesting in the context of this work. According to Fowler [29], these are functioning code components that have a poor structure. Several aspects are listed that may indicate a smell: duplicated code, long methods, large classes, long parameter lists, divergent changes, data lumps, lazy classes, or comments that try to make bad code understandable are just a few of these aspects. [29]. Thus, they are symptoms or problems at the code or design level [30]. They are an important factor that can lead to technical debt (TD) and affect the maintainability of a software system [31]. By making the software work despite smells, they are classified as mostly invisible, in terms of TD [32].

This work focusses on TD. Because it is a metaphor, a concrete definition is difficult. The metaphor states that short-term advantage comes at the expense of the long-term quality of a software system. It is first mentioned in Cunningham [33] and initially referred only to software implementation. Later research has extended this to architecture, design, documentation, requirements, and testing [34, 35]. Another extension of the metaphor says that software development tasks are postponed but carry the risk of causing problems in the future if they are not caught up [36]. A modern definition is:

> A design or construction approach that is expedient in the short term, but creates a technical context in which the same work is more expensive later than it is now (including increased costs over time). [37]

In many studies, TD is classified as a risk [35]. This is justified by internal quality, which can affect future development.

On the one hand, there is the intentional TD. This occurs when a developer needs to meet a deadline or uses a work-around solution in complex code. Code complexity, the risk of damaging the code by changing it, or compromising functionality can also be reasons to intentionally include TD [38].

On the other hand, there is unintentional TD. This occurs when a developer is not competent enough to find the cleanest solution to a problem, the team does not adhere to the necessary standards, or when the technology used needs to be actualized. Also, unintentional TD can occur when customer requirements are very specific or complex [38]. The metaphor of debt also states that there is interest that accumulates the longer the debt remains unpaid, further increasing TD should it remain unpaid [32, 33].

SonarQube is a tool used to analyze source code, among other things. It provides an evaluation of the code according to security, bugs, and CS. TD is also estimated with a temporal indication. SonarQube is widely used in the industry. Regarding this, the actual time needed to improve is lower than indicated by SonarQube [39, 40]. Saarimaki et al. [40] had students improve other people's open-source code using SonarQube for this purpose, so it is likely that an experienced developer would have been even faster with their own code. However, the most accurate data on the TD can be found at CS. This is confirmed in a downstream study, which also says that most code that generates TD is CS [41]. Therefore, in this work, the focus is also on TD generated by CS. In the data collection of this work, the SonarQube analysis result is used in the leaderboard game.

11.2.3 Gamification and Software Development Quality

The research on gamification and its influence on software quality is diverse. An overview of research results is provided in Fig. 11.1. Fraser [42] analyzed gamification of the testing process in the areas of teaching, practice, and crowdsourcing. Fraser [42] does not provide any evaluation results of his prototype. A literature review was conducted by Hamari et al. [43]. In this, the current state of research regarding gamification is analyzed.

The tool CodeSmellExplorer, which was conceptualized and developed by Raab [44], used gamified elements to teach university students about the importance of good programming styles. In the evaluation of its prototype, Raab [44] could confirm that students perceive gamification elements, particularly coding challenges, as interesting and stimulating.

Crowd development, i.e., the creation of software in a very-small-step manner (micro-tasks) by many developers, was investigated by LaToza et al. [45]. "Crowd development envisions a software development process optimized for sharing knowledge, distributing work efficiently, motivating contributions, and ensuring quality" [45]. In the context of gamification, Dubois and Tamburrelli [46] studied how a gamified application should be designed and used. They compared to groups (with and without competition) and found out that competition leads to slightly more Javadoc and test coverage. They explain this effect by the usage of metrics from other students as benchmark for their own development [46].

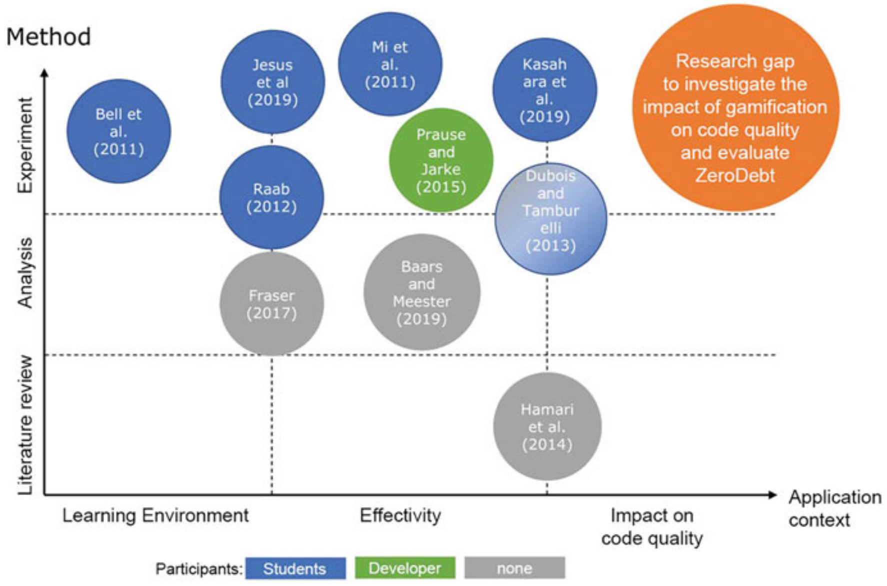


Fig. 11.1 State of the art in gamification research

To get students into the habit of good software testing, HALO by Bell et al. [47] attempts to establish this habit in a hidden way, i.e., without pointing out the actual goal. So far, the effects of HALO were not evaluated. Further attempts to teach students good programming practice through gamification are made by Kasahara et al. [48] and Mi et al. [49]. Kasahara et al. [48] applied a leaderboard containing code metrics for motivating code quality improvements. They could show that the code complexity decreases through this gamification element. Mi et al. [49] developed and evaluated a game called GamiCRS, which is an "online platform for students to learn code readability" [49]. It applies points, badges, and leaderboards in order to motivate students. The authors evaluate GamiCRS by conducting a post-application survey with technology acceptance constructs. The results confirm that the game motivates students to increase code quality. In a professional setting, maintaining good programming style using gamification is studied by Prause and Jarke [50]. In two agile software development experiments and surveys, they could show that gamification can motivate the following coding conventions. CodeArena is a tool developed by Baars and Meester [51] that attempts to incorporate CS and bugs in a game, in this case Minecraft, and encourages the player to improve them.

In this chapter, we shed more light into the impact of gamification and particularly of leaderboards on code quality and software development motivation. The research gap is also visualized in Fig. 11.1.

11.3 Research Design

11.3.1 Planning the Research

In order to answer the research question, we follow the call for more mixed-method research in information systems [52] and conduct experiment with software developers as well as semi-structured focus group interviews [53]. The research process comprises six steps and is depicted in Fig. 11.2.

As outlined in the motivation section, software developers often perceive software quality as a disturbing and annoying task. Against this background, we analyzed relevant gamification literature. As outlined in the previous section, some research on the effects of gamification on software developers exist [7, 9]. However, no research work explicitly investigates the effects of leaderboards on the software quality. In order to close this gap and to contribute to the body of gamification knowledge, the chapter at hand investigates the motivational effects of applying leaderboards to increase software quality. Our research work aims at answering the research question:

How does a leaderboard-based gamification approach motivate software developers to increase software quality?

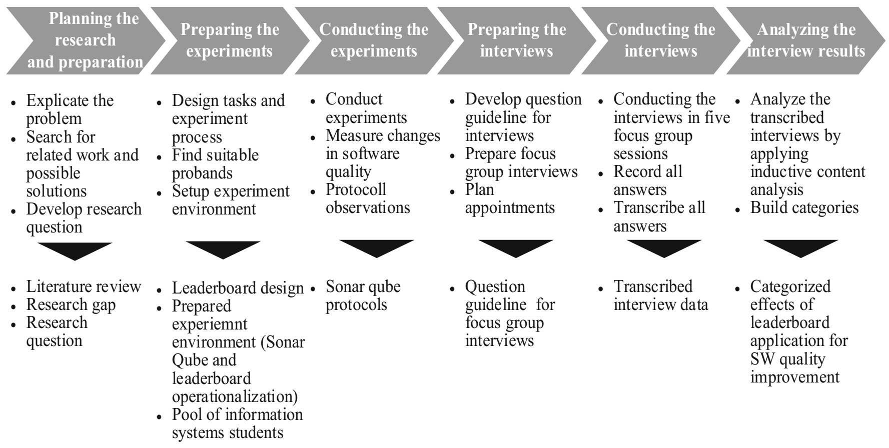

Planning the research and preparation
Preparing the experiments
Conducting the experiments
Preparing the interviews
Conducting the interviews
Analyzing the interview results
Explicate the problem
Search for related work and possible solutions
Develop research question
Design tasks and experiment process
Find suitable probands
Setup experiment environment
Conduct experiments
Measure changes in software quality
Protocoll observations
Develop question guideline for interviews
Prepare focus group interviews
Plan appointments
Conducting the interviews in five focus group sessions
Record all answers
Transcribe all answers
Analyze the transcribed interviews by applying inductive content analysis
Build categories
Literature review
Research gap
Research question
Leaderboard design
Prepared experiemnt environment (Sonar Qube and leaderboard operationalization)
Pool of information systems students
Sonar qube protocols
Question guideline for focus group interviews
Transcribed interview data
Categorized effects of leaderboard application for SW quality improvement

Fig. 11.2 Research process, activities, and outcomes

11.3.2 *Preparing and Conducting the Experiments*

To answer the research question, we applied an experiment with six information systems (IS) student groups. The experiment environment is as follows. The experiment takes place within the course "interdisciplinary information systems project," which belongs to the fifth semester of the IS bachelor curriculum. Within that course, the students need to work on software development projects, whereas the product owner and all requirements come from an industry partner. Consequently, each student group works in different software development projects with different technological requirements and applied programming languages. An overview of the project goals, with the assigned groups and the programming languages used, is provided in Table 11.1. All experiment and interview participants are listed in Table 11.2. Due to privacy concerns, the participants A3, C1, and C4 did not want to offer their age, semester, and working experience.

The teams initially work freely on their projects. This includes weekly sprint reviews and retrospectives, which take place together with the product owner from the industry or an instructor. The way of working is agile, and SCRUM is used.

The first four sprints (4 weeks) deliver the baseline without the leaderboard game. The project members should first get familiar with the new project and team. Beginning with the fifth sprint, we began analyzing the code in order to prepare a leaderboard. At the evening before the sprint reviews, the git repositories of the groups are analyzed with SonarQube to read out the TD. The analysis results are the basis for the leaderboard game.

The leaderboard game was voluntary for the project groups. The groups were given access to SonarQube code analysis of their project code. In this analysis, they can see the code locations, code smells (CS), and bugs, as well as the estimated TD. SonarQube accesses the groups' git repositories and has been updated with each

Table 11.1 Project overview

Project	Project goal	Participants	vue.js	Java	JS	TS	HTML	CSS	PHP	Kotlin
A	Task list and planner with templates	A1, …, A5		✓	✓	✓	✓	✓		✓
B	360° customer view for an insurance company's internal service staff	B1, …, B4	✓	✓		✓	✓			
C	Project management app for craft businesses	C1, …, C4			✓			✓	✓	
D	App for local retail to enter products into an online catalog	D1, …, D5	✓		✓		✓	✓		
E	Everyday life facilitation for cancer patients	E1, …, E4	✓		✓		✓	✓		
F	Yard Management Dashboard for Logistics Companies	F1, …, F4	✓	✓	✓		✓	✓		

Table 11.2 Experiment und interview participants

Participant	Project	Age	Semester	Gender	Work experience?
A1	A	23	7	M	No
A2	A	21	4	M	No
A3	A	n/a	n/a	F	n/a
A4	A	23	7	M	No
A5	A	21	5	M	No
B1	B	23	10	M	No
B2	B	20	5	M	No
B3	B	26	10	M	No
B4	B	24	7	F	No
C1	C	n/a	n/a	M	n/a
C2	C	20	4	M	No
C3	C	26	5	F	No
C4	C	n/a	n/a	M	n/a
D1	D	22	4	M	No
D2	D	23	5	F	No
D3	D	22	7	M	No
D4	D	27	5	M	Yes
D5	D	23	5	M	No
E1	E	21	5	M	No
E2	E	30+	8	M	No
E3	E	20	5	M	No
E4	E	23	5	M	No
F1	F	22	7	M	No
F2	F	21	7	M	No
F3	F	22	7	M	No

new version that is committed to the master branch. This allows the groups to track their improvements. The goal of the game is to reduce the TD. Since the projects vary in size, we developed a calculation key to scale the effort and assign points to the team's performance. This is as follows:

$$\delta TD\left(t_n, t_{n-1}\right) = \frac{TD_{LoC}\left(t_n\right)}{TD_{LoC}\left(t_{n-1}\right)} * 100 = \frac{\mathrm{TD}\left(t_n\right)}{\mathrm{TD}\left(t_{n-1}\right)} * 100 \qquad [\%]$$

$$TD_{LoC}\left(t_n\right) = \frac{TD\left(t_n\right)}{LoC\left(t_n\right)} * 100 \qquad [\text{Minutes}/100 \text{ Lines of Code}]$$

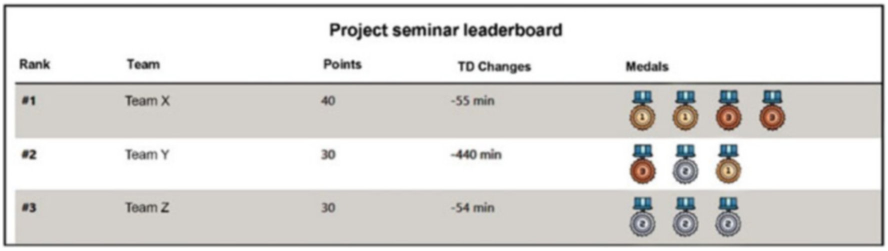
$$LoC\left(t_n\right) := Lines\ of\ Code\ in\ Project\ at\ t_n$$

$$TD\left(t_n\right) := Technical\ Depth\ at\ t_n \qquad [Minutes]$$

The formula *δTD* takes into account how much TD has been reduced within a certain period of time (here: current week compared to previous week). Thus, groups with a high TD reduction will receive a high score, while groups with a low TD reduction will receive a low score. TD_{LoC} is used as an additional metric. It describes how much TD is present per 100 lines of code at a given time. While *δTD* is thus limited to the change, TD_{LoC} provides a normalized evaluation of the code quality. *δTD* can do without TD_{LoC}, since the additional variables shorten it.

The group that achieves the highest *δTD* in the respective week receives 15 points and thus the first place. The second place receives 10 points and the third still 5. In addition, gold, silver, and bronze medals are awarded according to the placement.

This calculation was done once a week. Each student receives an email containing the current leaderboard, which exemplary is depicted in Fig. 11.3. The weekly winner team then received a medal, which remained until the end of the game and was displayed in the leaderboard overview. Thus, in addition to the weekly winner, an overall winner can also be determined. The leaderboard, which recorded the points and medals, as well as the reduced TD, was available for groups to view at any time and was updated weekly. All groups were notified when it was updated. As an additional motivation, the winning group was promised a certificate of achievement.



Project seminar leaderboard

Rank	Team	Points	TD Changes	Medals
#1	Team X	40	-55 min	🥇🥇🥉🥉
#2	Team Y	30	-440 min	🥉🥈🥇
#3	Team Z	30	-54 min	🥈🥈🥈

Fig. 11.3 Leaderboard example

11.3.3 Preparing and Conducting the Interviews

After finishing the leaderboard game and the project phase, we conducted focus group interviews to obtain feedback on the participant's perceptions and to assess the motivational effect of the game on the participants. The development of the interview guideline followed the recommendations of Brinkmann and Kvale [53]. All interviews were audio recorded and transcribed. The interview transcripts are the basis for the final analysis. The interview guideline comprises the following main questions, which are all of type open:

1. Independent of the game, how did you engage with software quality this semester?
2. To what extent did you notice the quality of your code using SonarQube?
3. What do you think of SonarQube as a tool?
4. What obstacles did you face in using it?
5. Did the leaderboard draw your attention to software quality?
6. What was the relevance of software quality before the introduction of the leaderboard?
7. What motivated you to reach the gold medal or at least the top 3?
8. Who of you ignored the game? Why?
9. What motivating aspects did the game have?
10. What did you think was bad/needs improvement?
11. How do you feel differently about code quality now?

11.3.4 Analyzing the Interview Results

To derive the perceived effects of applying a leaderboard into software development projects, we deeply analyzed the interview transcripts, which were generated during the focus group sessions. Therewith, we aim at inductively analyzing the content and generating categories of effects. For the inductive content analysis, we follow Mayring [54] and apply an iterative process. Each time we identified a new category and finished the analysis of one interview transcript, we again begin with the already analyzed transcripts. Answers to a question may contain different concepts that are categorized according to the category description. A coding example is provided in Table 11.3.

We develop concepts based on the core statements, which are then used to form categories. In addition, we count the statement and the interview frequency and evaluate the relevance of the statement. The statement frequency shows how many statements are made about this concept in the interview. The interview frequency quantifies the number of interviews, in which this concept is discussed.

Table 11.3 Example of statement categorization

Category	Concept	Example statement
Motivational effects	Competition	"When you see that others [...] are better, have more points [...], have a better grade, then you always have the incentive to keep up, [...] to catch up, that you then also want to collect more points or want to overtake"
Motivational effects	Awards	"I also find it motivating that you now receive such an award at the end"
Willingness to play	Active participation	"We did look in though and especially Code Smells we had found and removed them"

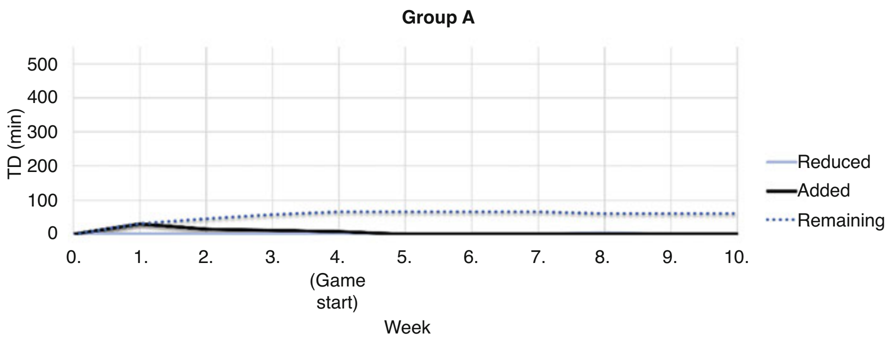


Fig. 11.4 Progress of TD from group A

11.4 Results

11.4.1 Effects of a Leaderboard on Software Quality

All observed student projects work on their tasks for a total of 10 weeks and 10 sprints. The leaderboard game starts at the beginning of Sprint 5, where teams are given access to their code analyses in SonarQube, along with the indication that the reduction of TD will be rewarded with points in the leaderboard game. Playing is optional for the students, and the results are not graded. The game was played 5 weeks. Project Group C did not participate in the game. In the TD progressions, the start time of the game is marked. It should be noted that none of the groups had previously mined TD. The project groups received the first leaderboard, and thus the first feedback on their code 1 week after the game began. In the following, we go through the results group wise.

In the first 2 weeks of the game, Group A has not made any improvements. Only in the third week after the start of the game the code is improved, and the TD is reduced by 5 min. This represents a percentage improvement of 7.5% to 4.51 min/100 LoC. After that, the group did not make any further improvements. The course of the TD of Group A can be seen in Fig. 11.4.

In the penultimate week of the game, Group B made code improvements for the first time. TD was reduced in the amount of 429 min. This corresponds with almost 94% to almost the entire TD built up until then. On 100 LoC only 3.6 min TD came thereby. However, it is noticeable that significantly less new TD was built up 1 week before the start of the game (game announcement) and afterward. Figure 11.5 shows the course of the TD of group B.

Team D made code improvements regularly, but not every week. No comparative values are available for this group before the start of the third sprint because git code pushes appears right after the third sprint. The team was able to reduce TD by a total of 68 min, which corresponds to approximately 12.4%. In terms of the number of lines of code, the TD reaches 0.46 min/100 LoC. The development of the TD of group D is provided in Fig. 11.6.

Group E has made code improvements every week since the beginning of the game. In the third week, the entire TD has already been reduced. At the same time, the team creates a continuous quality improvement, because every newly built TD is immediately reduced. A total of 144 min of TD are reduced during the ten sprints.

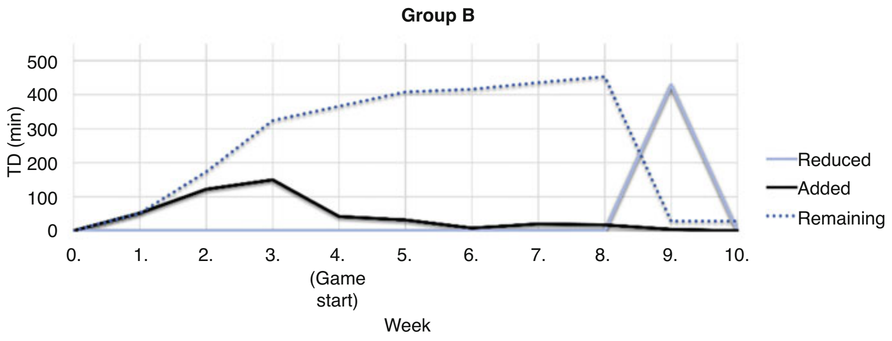


Fig. 11.5 Progress of TD from group B

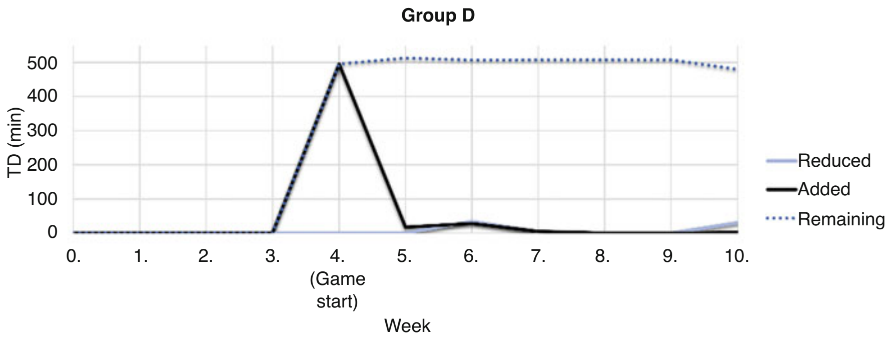


Fig. 11.6 Progress of TD from group D

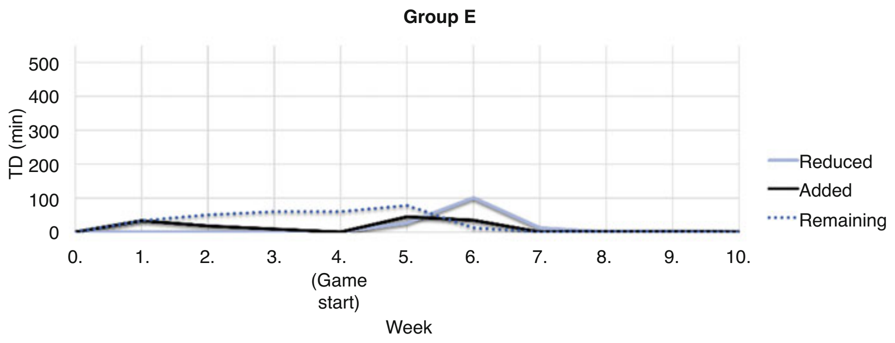


Fig. 11.7 Progress of TD from group E

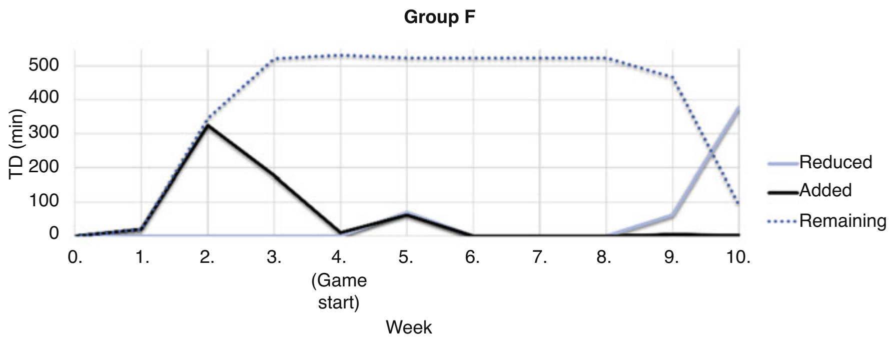


Fig. 11.8 Progress of TD from group F

The code of this group is free of TD at the end of the game, which can be seen in Fig. 11.7.

In the first week of the game, Group F has improved 70 min, and none in the following 3 weeks. Only in the last 2 weeks is code improved again, resulting in a reduction in TD amounting to 440 min. In total, the TD is reduced by 510 min, which corresponds to 84.9% of the total debt. This results in a value of 9.44 min/100 LoC. The course of the TD of group F can be found in Fig. 11.8.

Group C acts as a comparison group, since it did not participate in the game. This provides an opportunity to view the group's results in comparison to the others. In group C, no code improvements are made during the entire period. At the same time, a lot of TD was built. The progression of TD for this group can be seen in Fig. 11.9.

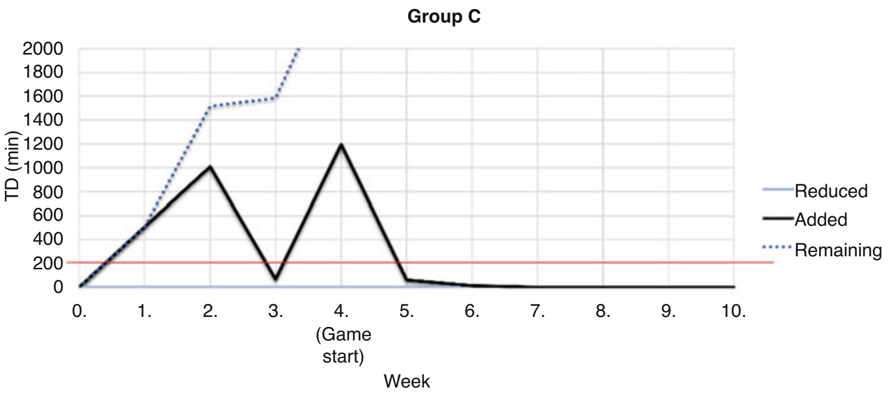


Fig. 11.9 Progress of TD from group C (comparison group)

Table 11.4 Interview results

Category	Statement frequency	Interview frequency
Motivational effects		
1. Competition	17	5
2. Quality Metrics	10	4
3. Awards	4	2
Willingness to play		
4. Active participation	10	2
5. Passive participation	5	1

11.4.2 Motivational Effects of Leaderboards on Software Quality

Next to the code quality results of the experiment, we interviewed all team members in dedicated focus group interviews. The inductive content analysis reveals two main categories, which are motivational effects and willingness to play. In total, five concepts could be retrieved by analyzing the interview transcripts (Table 11.4).

11.4.3 Motivational Effects

The first category comprises the impact of leaderboards on the participant motivation. The concepts of competition, code quality, and awards are considered. We perceive competition as the motivational effect that results from comparing and competing against the other teams. It is therefore an extrinsic motivation. Another extrinsic motivation is the award (medals) given when a team achieves one of the first three places in a given week. The concept of code metrics involves measuring code improvement as a motivating factor. It describes the motivating effect of passing quality gates and seeing code quality metrics.

The *competition* has motivated many of the participants. Thus, the claim to be better than the competitors is described as quite normal: “If you see that others [...] are better, have more points [...] or a better grade, then you always have the incentive to want to keep up with them, [...] which makes you want to collect more points or overtake them” (Interviewee A4). Attention is also paid to how the other teams are performing: “We always looked to see how the other teams were doing” (Interviewee B1). In particular, the ranking is perceived as the most motivating element, even if the general motivation is not particularly high in this group: “The motivation to really pay attention to the ranking was not particularly high for us. Still, it was the most motivating aspect because of the competitiveness” (interviewee D1). The low motivation of this group can be attributed to a lack of time and experience: “[Since] we were not yet so familiar with vue.js, [...] the topic of software quality [in the case of problems] loose attention until we received an e-mail with the updated ranking. [...] Maybe it would have been different with another programming language or another project” (Interviewee D1).

In other groups, however, motivation has increased as a result of the leaderboard. Again, the interest in the ranking of the other groups is clear: “It definitely motivates. [...] So you regularly looked to see what points the others had. And of course, you want to be [...] better—so it motivates you to deal with your mistakes through this” (Interviewee E2). “If there hadn’t been this competition, I don’t know if we would have made such small corrections” (Interviewee E2).

There is also an intrinsic motivation that one’s own CQ is improved by the leaderboard game, which is represented by the *quality metrics*: “At the end of the day, there were various metrics (CS, bugs, vulnerability and test coverage). I found it motivating to work towards passing the overall test (quality gate)” (Interviewee F3). “[It was motivating] to be better than the others, by making the code better” (Interviewee F2). Again, the competitive nature of the leaderboard is evident.

The *awards* and medals are also perceived as motivating by several persons. Statements such as “If we do [the following action] like this now, we might get the gold medal again or at least the silver medal” were made (Interviewee E2). When asked about motivating aspects, it was said, for example, “the medals definitely. [...] I think that always motivates” (interviewee E2), or “I also find it motivating that you now receive such an award at the end” (interviewee E1).

11.4.4 Willingness to Play

Willingness to play is a category that comprises reasons to participate in the game as well as limiting factors. Two forms of participation are differentiated: First, *active participation* exists by contributing code improvements to perform better in the game. In addition, there is *passive participation*, in which the code improvements are carried out independently of the game. Some participants actively participated in the game: “We especially looked for and removed the CS” (interviewee B1). “There was a lot of interest in the moments when SonarQube provided the new analysis results. That’s when everybody was kind of peeked in and looked to see what the quality of our code was” (interviewee B3).

Some participants merely played passively, which leads to *passive participation*. Here, communication among the teams and mediation in the overall context was given as a reason: “I was generally a bit lost, you also heard relatively little from the other teams. The competitive idea behind it also went down as a result—you didn’t know who was taking it seriously now and who wasn’t?” (Interviewee A5).

11.5 Discussion and Outlook

The results described in this chapter provide an answer to the research question of how a leaderboard-based gamification approach may motivate software developers to increase software quality.

First, the leaderboard game acts as motivational trigger at least for the better groups for conducting code improvements. This gets clear by considering the statement of interviewee D: “I think without this leaderboard [...], we would have thought zero about [code quality].” Just like Prause and Jarke [50], it can be concluded that gamification has a better effect in a non-time-sensitive context. Furthermore, developers who improve their code mutually within a team contribute not just to the overall understanding of SQ but also to the functionality of the software. Furthermore, the results indicate that playing software developers draw more attention on quality and are more consciously about it. However, one single team member might work more responsibly, while others contribute only little to the quality, which we could not measure. We address this challenge by a not too strict set of rules, which encourages all team members to cooperate. It is also important to consider different types of players. Some participants felt extrinsically motivated, i.e., by the comparison or competition with the other teams or the desire to achieve a particularly good place. Other participants stated that they were motivated solely by the demand for their own CQ, i.e., more intrinsically. Thus, the leaderboard game was able to trigger both types of motivation.

Second, leaderboard-based gamification stimulates and arises interest at software developers for improving code quality. Raab [44] found out that students often write unmaintainable code and fail in recognizing bad coding practices. We can solely partially confirm this observation. The participating students have at least built up a sense of what good practices are, and they were engaged with their code independently of the game. That gamification stimulates and arises interest and is consistent with the results of this work. Applying a leaderboard increases the interest for clean code and thus learning software development, which is consistent with Dubois and Tamburrelli [46]. However, predicting the desired effect is also difficult in that environment. Above all, choosing the right elements of gamification is a challenge. In line with Fraser [42], the results of this work reveal that gamification has an influence on the teaching of SQ. In contrast to our study, Fraser [42] considers solely the testing of software. Similar to Bell et al. [47], the social learning environment approach contributed to participants' engagement with SQ and CQ issues. However, this was not attempted via hidden methods but apparently as a competition with the goal of achieving better quality in the software.

Third, the results show that gamification in the form of a leaderboard with points and badges has a positive impact on CQ. All groups that participated in the game improved their code. Initially, rather small improvements were made. This was due to the initial uncertainty of how many of the other teams would even participate in the game. However, as soon as it became apparent that a competition was taking place, some teams became much more active in their efforts to improve their own code. Some groups removed all or a very high percentage of their TD. In contrast, the game CodeArena by Baars and Meester [51] was not tested with attendees of an experiment. However, the approach to fix bugs in a game is comparable to the leaderboard game we presented. The results of this work show that this approach works. In addition, we confirm that the idea not only increased the CQ but also experiences are made and learning progress is achieved. The results of this work are also consistent with those of Kasahara et al. [48]. The groups that participated in the game improved their code, while the group that did not participate did not make any improvements. Furthermore, many participants stated that they were under time pressure in the project because they had to deliver functional results for the first time and they were additionally busy with the rest of their studies. Thus, we assume that the participants were not able to divide their project working time completely free and independent. Working in full-time for the project would most likely further increase the CQ.

Against this background, we contribute to research in two ways. First, we could show that the introduction of a leaderboard game has a measurable effect on the CQ in software development projects. Second, we shed light into the motivational effect of gamification and in particular the application of leaderboards to undergraduate information systems students. Furthermore, we could prove the positive effects of gamification, which is in line with the findings of Hamari et al. [43].

The validity of the results are subject to limitations. First, the different software development teams worked on completely different software artifacts, which may bias the results to some extent. However, the teams do not necessarily need to work on identical projects, because we solely focus on the change of the software quality due to the motivational effects of the leaderboard.

Second, the leaderboard game has weaknesses. The scoring, which tries to be fair by basing TD on LoC, is only applicable for improving teams. A team that writes good code from the beginning and thus builds up little TD would probably not take one of the first places, even if its TD is minimized. Differences between programming languages that require more or fewer lines of code are also not considered.

Third, it is not possible to predict how the CQ would have developed further after finishing the observation phase. The so-called novelty effect, i.e., the increased interest in new things [55], may be a reason for the motivation during the short-observed game period (6 weeks).

The generalizability of the results is limited to IS bachelor students with no or just little software quality experience. The sample size is with 27 participants rather small. In addition, the background of the selected participants was rather homogeneous, as they are all in the same study program in similar semesters. The experiment attendees were challenged with new programming languages and the requirements that such a project entails. This also caused time problems, which led to the fact that the game or the CQ was regarded as rather secondary. A development team that is familiar with the programming languages and used to work in an agile manner would not have had these problems. It cannot be concluded that an experienced team would have made more code improvements, because the CQ in professional development teams is better from the beginning. Therefore, the measured motivational effect may differentiate at more experienced or even older software developers. At the same time, in the context of teams, it is important to note that effects on an individual, such as those that can occur through gamification, can often affect the team as well. For example, the increased engagement of a team member through gamification can have a positive impact on the team [56]. Likewise, such game elements can focus the entire team on individual quality requirements (such as code quality). By using cooperation, the team could achieve a better mental model through artificial conflicts, which improves the team's performance [57]. Based on this, we expect gamification to bring improvements at the team level as well, but this should be further verified in future research.

Further questions for future research arise from this work. The leaderboard can be used more intensively in teaching. In addition, it needs to be evaluated in a professional context with experienced developers. Furthermore, the degree of gamification needs to be investigated. How much is too much or too little? The optimal degree of gamification is an aspect that should be investigated more closely in future research works. The time spent on gamification can also be considered, which leads to the question of how much time should or can be spent in order to achieve the best possible results in CQ. In terms of motivation, it could be analyzed whether competition with others, the own performance, or the feeling of playing as a team contributes the most. In the context of a multiplayer approach, it could be considered how this affects player motivation and outcome.

References

1. Huang, M.-H., Rand, W., Rust, R.T.: Don't do it right, do it fast? Speed and quality of innovation as an emergent process (2016)
2. Flora, H.K., Wang, X., Chande, S.V.: An investigation on the characteristics of mobile applications: a survey study. IJITCS. **6**, 21–27 (2014)
3. Singh, N., Soni, D.: Proposing new model for effort estimation of mobile application development. IJCA. **170**, 14–18 (2017)

4. Deak, A., Stålhane, T., Cruzes, D.: Factors influencing the choice of a career in software testing among Norwegian students. In: Artificial Intelligence and Applications / 794: Modelling, Identification and Control / 795: Parallel and Distributed Computing and Networks / 796: Software Engineering / 792: Web-Based Education. ACTAPRESS, Calgary, AB (2013)
5. Fernández-Sanz, L., Villalba, M.T., Hilera, J.R., et al.: Factors with negative influence on software testing practice in Spain: a survey. In: O'Connor, R.V., Baddoo, N., Cuadrago Gallego, J., et al. (eds.) Software Process Improvement. 16th European Conference, EuroSPI 2009, Alcala (Madrid), Spain, September 2–4, 2009. Proceedings, vol. 42, pp. 1–12. Springer, Berlin (2009)
6. Shah, H., Harrold, M.J.: Studying human and social aspects of testing in a service-based software company. In: Dittrich, Y. (ed.) Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, pp. 102–108. ACM, New York, NY (2010)
7. Deak, A., Stålhane, T., Sindre, G.: Challenges and strategies for motivating software testing personnel. Inf. Softw. Technol. **73**, 1–15 (2016)
8. Koivisto, J., Hamari, J.: The rise of motivational information systems: a review of gamification research. Int. J. Inf. Manag., 191–210 (2019)
9. Deterding, S., Dixon, D., Khaled, R., et al.: From game design elements to gamefulness. In: Lugmayr, A., Franssila, H., Safran, C., et al. (eds.) Proceedings of the 15th International Academic MindTrek Conference Envisioning Future Media Environments, p. 9. ACM, New York, NY (2011)
10. Liu, D., Santhanam, R., Webster, J.: Toward meaningful engagement: a framework for design and research of gamified information systems. MIS Quart. **41**, 1011–1034 (2017)
11. Augustin, K., Thiebes, S., Lins, S., et al.: ARE WE PLAYING YET? A REVIEW OF GAMIFIED ENTERPRISE SYSTEMS. In: PACIS 2016 Proceedings. AIS Electronic Library (2016)
12. Blohm, I., Leimeister, J.M.: Gamification. Design of IT-based enhancing services for motivational support and behavioral change. Bus. Inf. Syst. Eng. **5**, 275–278 (2013)
13. Hermanto, S., Kaburuan, E.R., Legowo, N.: Gamified SCRUM design in software development projects. In: 2018 International Conference on Orange Technologies (ICOT), pp. 1–8. IEEE (2018)
14. Marques, R., Costa, G., Silva, M.M.: Gamifying software development scrum projects. In: 2017 9th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games). Proceedings : 6–8 September 2017, Athens, Greece. IEEE, Piscataway, NJ (2017)
15. Snipes, W., Nair, A.R., Murphy-Hill, E.: Experiences gamifying developer adoption of practices and tools. In: Jalote, P., Briand, L., van der Hoek, A. (eds.) Companion Proceedings of the 36th International Conference on Software Engineering – ICSE Companion 2014, pp. 105–114. ACM Press, New York, NY (2014)
16. Marczewski, A.: Gamification. A simple Introduction & A Bit More, 2nd edn. Lulu.com Digital >16 (2013)
17. Fitz-Walter, Z., Tjondronegoro, D., Wyeth, P.: Orientation passport. In: Stevenson, D. (ed.) Proceedings of the 23rd Australian Computer-Human Interaction Conference, pp. 122–125. ACM, New York, NY (2011)
18. Mekler, E.D., Brühlmann, F., Opwis, K., et al.: Disassembling gamification. In: Mackay, W.E. (ed.) CHI '13 Extended Abstracts on Human Factors in Computing Systems, p. 1137. ACM, New York, NY (2013)
19. Deterding, S., Khaled, R., Nacke, L., Dixon, D.: Gamification: toward a definition. In: ACM Conference on Human Factors in Computing Systems (CHI) (2011)
20. Huang, B., Foon, K.: Do points, badges and leaderboard increase learning and activity: a quasi-experiment on the effects of gamification. In: Ogata, H., Chen, W., Kong, S.C., Qiu, F. (eds.) Proceedings of the 23rd International Conference on Computers in Education (2015)
21. Hamari, J., Eranti, V.: Framework for Designing and Evaluating Game Achievements (2011)
22. Zähl, P.M., Biewendt, M., Wolf, M.R., et al.: Requirements for competence developing games in the environment of SE competence development. In: 35. Jahrestagung des Arbeitskreises Wirtschaftsinformatik an Hochschulen für Angewandte Wissenschaften im deutschsprachigen Raum (AKWI), pp. 73–88

23. Sailer, M., Hense, J., Mandl, H., et al.: Psychological perspectives on motivation through gamification. Interact. Des. Arch. J. (2013)
24. Reiss, S.: Intrinsic and extrinsic motivation. Teach. Psychol. **39**, 152–156 (2012)
25. Stieglitz, S., Lattemann, C., Robra-Bissantz, S., et al. (eds.): Gamification. Using game elements in serious contexts. Progress in IS. Springer, Cham (2017)
26. Ryan, D.: Intrinsic and extrinsic motivations: classic definitions and new directions. Contemp. Educ. Psychol. **25**, 54–67 (2000)
27. Kanellopoulos, Y., Antonellis, P., Antoniou, D., et al.: Code quality evaluation methodology using the ISO/IEC 9126 standard. IJSEA. **1**, 17–36 (2010)
28. Breuker, D.M., Derriks, J., Brunekreef, J.: Measuring static quality of student code. In: Rößling, G. (ed.) Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, p. 13. ACM, New York, NY (2011)
29. Fowler, M.: Refactoring. Improving the design of existing code, 28. printing. The Addison-Wesley object technology series. Addison-Wesley, Boston (2013)
30. Fontana, F.A., Ferme, V., Zanoni, M., et al.: Towards a prioritization of code debt: a code smell Intensity Index. In: Ernst, N.A. (ed.) 2015 IEEE 7th International Workshop on Managing Technical Debt (MTD 2015). Bremen, Germany, 2 October 2015, pp. 16–24. IEEE, Piscataway, NJ (2015)
31. Tufano, M., Palomba, F., Bavota, G., et al.: When and why your code starts to smell bad (and whether the smells go away). IIEEE Trans. Softw. Eng. **43**, 1063–1088 (2017)
32. Kruchten, P., Nord, R.L., Ozkaya, I.: Technical debt: from metaphor to theory and practice. IEEE Softw. **29**, 18–21 (2012)
33. Cunningham, W.: The WyCash portfolio management system. In: Archibald, J.L., Wilkes, M.C. (eds.) Conference proceedings / OOPSLA '92. 18–22 October 1992, Vancouver, British Columbia, pp. 29–30. Association for Computing Machinery, New York, NY (1993)
34. Brown, N., Ozkaya, I., Sangwan, R., et al.: Managing technical debt in software-reliant systems. In: Roman, G.-C. (ed.) Proceedings of the FSESDP Workshop on Future of Software Engineering Research, p. 47. ACM, New York, NY (2010)
35. Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. J. Syst. Softw. **101**, 193–220 (2015)
36. Alves, N.S., Mendes, T.S., de Mendonça, M.G., et al.: Identification and management of technical debt: a systematic mapping study. Inf. Softw. Technol. **70**, 100–121 (2016)
37. Kruchten, P., Nord, R.L., Ozkaya, I., et al.: Technical debt. SIGSOFT Softw. Eng. Notes. **38**, 51–54 (2013)
38. Yli-Huumo, J., Maglyas, A., Smolander, K.: How do software development teams manage technical debt? – An empirical study. J. Syst. Softw. **120**, 195–218 (2016)
39. Marcilio, D., Bonifacio, R., Monteiro, E., et al.: Are static analysis violations really fixed? A closer look at realistic usage of SonarQube. In: ICPC 2019. 2019 IEEE/ACM 27th International Conference on Program Comprehension : Proceedings : Montréal, Canada, 25 May 2019, pp. 209–219. IEEE, Piscataway, NJ (2019)
40. Saarimaki, N., Baldassarre, M.T., Lenarduzzi, V., et al.: On the accuracy of SonarQube technical debt remediation time. In: Staron, M., Capilla, R., Skavhaug, A. (eds.) 45th Euromicro Conference on Software Engineering and Advanced Applications. SEAA 2019 : 28–30 August 2019, Kallithea, Chalkidiki, Greece : Proceedings, pp. 317–324. IEEE, Piscataway, NJ (2019)
41. Staron, M., Capilla, R., Skavhaug, A. (eds.): 45th Euromicro Conference on Software Engineering and Advanced Applications. SEAA 2019 : 28–30 August 2019, Kallithea, Chalkidiki, Greece : Proceedings. IEEE, Piscataway, NJ (2019)
42. Fraser, G.: Gamification of software testing. In: 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing. AST 2017 : 20–21 May 2017, Buenos Aires, Argentina : Proceedings, pp. 2–7. IEEE, Piscataway, NJ (2017)
43. Hamari, J., Koivisto, J., Sarsa, H.: Does gamification work? – A literature review of empirical studies on gamification. In: 2014 47th Hawaii International Conference on System Sciences. IEEE (2014)

44. Raab, F.: CodeSmellExplorer: tangible exploration of code smells and refactorings. In: Erwig, M. (ed.) 2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2012). Innsbruck, Austria, 30 September–4 October 2012, pp. 261–262. IEEE, Piscataway, NJ (2012)
45. LaToza, T.D., Ben Towne, W., van der Hoek, A., et al.: Crowd development. In: Prikladnicki, R. (ed.) 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2013), San Francisco, California, USA, 25 May 2013; Part of the 35th International Conference on Software Engineering (ICSE), pp. 85–88. IEEE, Piscataway, NJ (2013)
46. Dubois, D.J., Tamburrelli, G.: Understanding gamification mechanisms for software development. In: Meyer, B. (ed.) Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, p. 659. ACM, New York, NY (2013)
47. Bell, J., Sheth, S., Kaiser, G.: Secret ninja testing with HALO software engineering. In: Maalej, W. (ed.) Proceedings of the 4th International Workshop on Social Software Engineering, p. 43. ACM, New York, NY (2011)
48. Kasahara, R., Sakamoto, K., Washizaki, H., et al.: Applying gamification to motivate students to write high-quality code in programming assignments. In: Scharlau, B. (ed.) Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, pp. 92–98. Association for Computing Machinery, New York, NY (2019)
49. Mi, Q., Keung, J., Mei, X., et al.: A gamification technique for motivating students to learn code readability in software engineering. In: Wang, F.L. (ed.) 2018 International Symposium on Educational Technology. ISET 2018: 31 July–2 August 2018, Osaka, Japan : Proceedings, pp. 250–254. IEEE, Piscataway, NJ (2018)
50. Prause, C.R., Jarke, M.: Gamification for enforcing coding conventions. In: Di Nitto, E. (ed.) Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, pp. 649–660. ACM, New York, NY (2015)
51. Baars, S., Meester, S.: CodeArena: inspecting and improving code quality metrics using minecraft. In: 2019 IEEE/ACM International Conference on Technical Debt. TechDebt 2019 : 26–27 May 2019, Montréal, Canada, pp. 68–70. IEEE, Piscataway, NJ (2019)
52. Venkatesh, V., Brown, S.A., Bala, H.: Bridging the qualitative-quantitative divide: guidelines for conducting mixed methods research in information systems. MIS Quart. **37**, 21–54 (2013)
53. Brinkmann, S., Kvale, S.: Doing Interviews. SAGE (2018)
54. Mayring, P.: Qualitative Inhaltsanalyse. In: Mey, G., Mruck, K. (eds.) Handbuch Qualitative Forschung in der Psychologie, 1. Aufl, pp. 601–613. VS Verlag für Sozialwissenschaften (GWV), s.l. (2010)
55. Clark, R.E.: Reconsidering research on learning from media. Rev. Educ. Res. **53**, 445 (1983)
56. Dutra, A.C.S., Prikladnicki, R., Conte, T.: What are the main characteristics of high performance teams for software development? In: Proceedings of the 17th International Conference on Enterprise Information Systems, pp. 145–152. SCITEPRESS – Science and Technology Publications (2015)
57. de Dreu, C.K.W., Weingart, L.R.: Task versus relationship conflict, team performance, and team member satisfaction: a meta-analysis. J. Appl. Psychol. **88**, 741–749 (2003)

Chapter 12
Designing a Serious Game for Cybersecurity Education

Gabriele Costa and Marina Ribaudo

Abstract Serious gaming is becoming fundamental for reducing the initial effort of studying highly complex and extremely technical subjects. Cybersecurity is no exception as it is in general perceived as one of the most difficult fields in computer science. This happens because cybersecurity is orthogonal to any specific technology. As a consequence, although many people may be interested in knowing more about cybersecurity, approaching the topic is often perceived as cumbersome, if not even frustrating. In this context, serious gaming can be adopted to create an engaging and controlled environment where players with no security skills may face realistic challenges. Needless to say, designing and implementing such games is itself a challenge. In this chapter, we present our experience with designing and implementing a serious game on cybersecurity, called A NERD DOGMA. Briefly, A NERD DOGMA is a classical escape room adventure where players have to progressively advance by solving some challenges. What truly characterizes A NERD DOGMA is that all of the enigmas are actual cybersecurity challenges. Each challenge is based on a real security scenario where the player, being the attacker or the on-field agent, has to exfiltrate data, break ciphers, and intrude in remote systems. The main objective is to provide inexpert users with a first-hand experience of how certain security operations are planned and executed. To this aim, a number of issues must be addressed. For instance, one cannot avoid introducing security tools, e.g., to scan a remote machine programmatically. However, requiring players to interact with a command line terminal might discourage most of them. Another difficulty emerges from the integration of third-party technologies. Most games are self-contained, i.e., they do not allow participants to directly interact with external systems or resources, and, in case it is necessary, they mimic the external environment. Nevertheless, this approach is not optimal for cybersecurity

G. Costa (✉)
IMT Lucca, Lucca, Italy
e-mail: gabriele.costa@imtlucca.it

M. Ribaudo
University of Genoa, Genoa, Italy
e-mail: marina.ribaudo@unige.it

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_12

where “thinking out of the box” is of paramount importance. Taking strategic design decisions requires a systematic assessment of these and other technical aspects that we present in this chapter.

Keywords Cybersecurity awareness · Gamification · Hands on training

12.1 Introduction

In the last decades, the computer security community has devoted considerable effort to investigating novel training methodologies. The main reason is that, since security is orthogonal to every technology, a modern security expert is required to master many different disciplines. For instance, a penetration tester may have to both inspect the PHP code written by expert Web developers and evaluate the user privileges configured by a system administrator.

As a result, *learning-by-doing* quickly emerged as an inspiring principle. The idea is that direct, practical experience is a highly effective and efficient training method. Unfortunately, in this context, “hands on” refers to both applied, e.g., binary code inspection, and theoretical, e.g., cryptography math, aspects of computer science and engineering. Hence, the effort in front of people willing to become security experts may appear gargantuan.

Gamification soon appeared as a valid alternative to more traditional training processes. In particular, *capture the flag* (CTF) competitions rapidly gained in popularity in the last years (see [20] and [18] for some early witnesses). A CTF provides an interactive learning environment for different skill levels, from beginners to practitioners and even experts.

CTF are organized more and more frequently; nevertheless, CTF players are usually highly motivated individuals. Since they are almost unknown to the great majority of people, CTF may be perceived as extremely complex competitions, only accessible to specialists and, thus, discouraging the participation of most individuals. In this case, a CTF would fail its primary goal: to bring knowledge and awareness about cybersecurity to a wide audience.

In this chapter, we report our experience with A NERD DOGMA, i.e., a quest CTF (see Sect. 12.3.1) in the form of an escape room experience specifically designed for beginners. The main goal of A NERD DOGMA is to provide an interactive and immersive gaming experience for allowing unskilled people to move the first steps in a real CTF and, more in general, in the field of cybersecurity. A NERD DOGMA consists of four entry-level, yet realistic challenges that must be solved by learning the rudiments during the game. All the challenges have been designed to be self-contained, i.e., no specific previous knowledge is needed to solve them. Moreover, some of the challenges must be solved by means of a (simplified) command-line terminal. The aim is to make the players understand how to interact with tools similar to those used in reality.

The design and implementation of A NERD DOGMA have been carried out in the last years. During this period, we defined the objectives and specifications used to validate the final product. Also, we implemented it in various forms, e.g., as an online video game and as a physical escape room. Finally, these implementations were utilized in various contexts for training and for increasing cybersecurity awareness. All these events confirmed the adequacy of our design and implementation.

This chapter is organized as follows. In Sect. 12.2, we survey on the related work. Section 12.3 presents the game objectives and requirements, while Sect. 12.4 describes the overall design. Section 12.5 details the different versions of the escape room implemented in the last years, and Sect. 12.6 provides our lesson learned. Finally, some concluding remarks are given in Sect. 12.7.

12.2 Related Work

Gamification techniques may be used for several purposes such as increasing motivation, improving learning outcomes, favoring teamwork, and introducing young students to STEM disciplines. This is well established in the context of computer science education as discussed, for example, in a recent study [1], which compares the performances of groups of students exposed to gamification techniques against control groups, e.g., their peers involved in more traditional activities.

Various experiences using gamification or CTF-like competitions have also been published for cybersecurity education. In most cases, these experiences involve university students attending cybersecurity classes, e.g., see [2, 5, 6]. In these papers, CTF-like exercises are used to train or evaluate students. Some results are shown, and in general, the authors state that the combination of both gamification and competition is an effective way to motivate students to put more effort into the study of cybersecurity topics, to increase their practical skills, and to promote teamwork. For a recent survey about other cybersecurity education experiences, we refer the interested reader to [17]. In the remaining part of this section, we limit our discussion to some projects we consider closer to our proposal.

The work in [8] presents a framework to design games having a pedagogic purpose for an audience of non-experts. In particular, the paper suggests the steps game designers should follow. These include the preliminary analysis of the target players and the available resources such as time constraints, budget, and technical skills of the developers. Classical software development steps such as design, deployment, and prototype assessment are also discussed. Then the game can go "live" with or without the supervision of learning facilitators. Finally, an evaluation phase should be performed to assess whether the game contributed to increasing cybersecurity awareness.

The authors of [15] suggest that the main characteristics of escape rooms, for instance, their underlying storytelling, cooperative nature, teamwork, and timing constraints, make them an ideal learning tool according to the constructivist learning theory. Players become an active part of the story, not a mere audience. The debriefing phase at the end of the game (if present) allows them to expose their view about the acquired knowledge, including possible difficulties. This narrative is common to several papers introducing learning experiences based on escape rooms.

For example, GenCyber[1] camps are summer cybersecurity camps organized in different places in the USA for middle and high school students. Each camp director can design a different learning format, and some results have been published in the literature. In one experience, presented in [11], the students have to access the IT infrastructure of the SPECTRE organization to find the password of a target account, owned by Ernst Blofeld, a villain of the James Bond universe. Further GenCyber challenges, mostly focusing on virtual reality, are presented in [7], where the authors also measure the appreciation of the camp participants. In [12], the authors describe their escape room, built following a path that is very similar to ours. For their first proof of concept, the authors turned a conference room into a physical escape room and organized teams of players who had a limited amount of time to solve several puzzles. The experience was fun and appreciated by the players, but hard to scale or move. The solution proposed to scale up is the Escape the Briefcase game, i.e., a bag with a combination lock and different puzzles that can be deciphered in different orders before the entire mystery is revealed. The bag is easy to "duplicate" to allow more teams to play at the same time. Before starting, the mission is explained (briefing), and the timer is set. After the game, a post-game discussion (debriefing) follows, to improve the comprehension of the material covered in each puzzle.

Other authors report the narration underlying their experience. For instance, [14] reports on a 10-week period during which students played an alternate reality game, based on the following story: "The daughter of a student expelled 20 years ago is back to her father's campus to avenge him, and her initial point of attack is the website of a security course [...]." The goal of the experience was to make the participants understand some key concepts of cybersecurity and to improve their skills to prevent cyberattacks. Results show that, after the course, students positively changed their perception, in terms of understanding the tasks and problems that need to be solved.

In [9], the authors use the story of a radical animal rights group, the Animal Freedom Battalion, willing to free an animal held in zoo captivity, to increase students' engagement. The goal of the students playing the role of activists is to compromise the zoo's website and then delete records about the animals from the zoo's inventory database. Then, they could break into the zoo and free the animal with no digital trace left in the database to witness that the animal ever was at the zoo. The challenge was divided into three phases that mimic an actual penetration testing methodology, i.e., reconnaissance, exploitation, and execution. The authors conclude their work by reporting positive, yet informal, feedback from the participants.

Finally, a more recent work [16] describes a first-person 3D escape room developed with the Unity game engine[2] to be used in class with the students

[1] https://www.gen-cyber.com/.

[2] https://unity.com/.

of a master's degree in cybersecurity. The plan is to integrate the scores of the students/players within the learning management system used for teaching. The game is currently under development, and it has not been tested with students yet.

Although all the experiences just described share some similarities with A NERD DOGMA, our escape room is substantially different since it targets a general audience of inexpert players, rather than university students. Also, as described in the following, since A NERD DOGMA comes with an open-source design, it can be implemented and used in different contexts.

12.3 Requirements and Objectives

In this section, we state all the requirements and objectives considered during the design and development of A NERD DOGMA.

12.3.1 Realism of Challenges

The primary goal was to provide players with a first-hand experience of cybersecurity. This objective is not new, as it is also common in several artworks, e.g., video games and movies, and contests. Thus, it is worth considering how cybersecurity has been presented in this context, possibly by comparing successful and unsuccessful results.

12.3.1.1 Nmap Cameos

Nmap[3] is perhaps the most famous network scanning tool. Interestingly enough, its developers also curate a list of Nmap appearances in movies. The list includes, for instance, *The Matrix Reloaded*,[4] *Die Hard 4*,[5] and *Snowden*.[6]

Clearly, these movies aim at being pleasant for a wide audience, and the appearances of Nmap mostly resemble Easter eggs.[7] Although no real scanning through Nmap was ever presented, the scenes gain realism by including actual penetration testing tools.

[3] https://nmap.org/.

[4] http://www.imdb.com/title/tt0234215/.

[5] http://www.imdb.com/title/tt0337978/.

[6] http://www.imdb.com/title/tt3774114/.

[7] See https://nmap.org/movies/ for details.

12.3.1.2 Movies on Hacking

A major example of the potentialities of combining realistic cybersecurity and fiction comes from the celebrated series *Mr. Robot*.[8] As a matter of fact, it is well known that the authors of the show resorted to security experts, e.g., Michael Bazzell. The result was a highly realistic fiction, which even inspired actual security tools.[9] Another prominent example is again *Snowden*, which narrates the true story of Edward Snowden, the former NSA security expert who revealed privacy violations perpetrated by some apparatuses of the US government.

Even though the security operations appearing in these movies may be realistic or based on actual events, hacking sequences are often oversimplified, and no technical details are given.

12.3.1.3 Cybersecurity in Video Games

Unlike movies, video games permit one to directly interact with a simulated environment. As a consequence, players must learn how to use the game controls to carry out specific tasks, including cybersecurity-related ones. Yet, as for movies, most video games target a general audience that might be not so interested in technical details.

In the last few years, some video games related to cybersecurity have been published. For instance, *Cyberpunk 2077* [3] and the *Watch Dogs* [13][10] saga are staged in next future's highly connected societies. There, most of the gameplay has to do with some sort of *hacking*. However, these games present no technical aspects as all the operations amount to mini-games, e.g., based on constraint solving or pattern matching.

An interesting exception is *Hacknet* [19]. This game specifically aims at providing an immersive experience on cybersecurity. This includes, for instance, a simulated terminal that accurately mimics a Linux shell. To the best of our knowledge, the level of realism achieved by Hacknet is possibly the highest for a video game.

12.3.1.4 Capture the Flag Competitions

As mentioned in the introduction, CTF competitions are real-life contests for security practitioners. A CTF amounts to a collection of challenges, each based on some technical aspect of cybersecurity. By solving the challenges, players collect

[8] https://www.imdb.com/title/tt4158110/.

[9] https://github.com/Manisso/fsociety.

[10] Interestingly enough, *Watch Dogs: Legion* was also a victim of a real data leakage attack, see https://www.zdnet.com/article/ubisoft-crytek-data-posted-on-ransomware-gangs-site/.

flags, i.e., strings in a given format, and they earn points. In most CTF events, challenges are independent, and players can select the next one from a global board. This kind of CTF is called *Jeopardy*. Another frequent type is *Attack/Defence* (A/D), where teams directly compete by exploiting and patching a vulnerable infrastructure. Less frequently, CTFs are organized in form of quests. In a *Quest*, CTF players have to progress along a storyline by solving the challenges. Clearly, in terms of engagement, a Quest CTF offers some advantages. Hence, it is not surprising that Jeopardy and A/D are often directed to expert players who have their own reference platform[11] where competitions are weekly announced so that individuals and teams can enroll, play, and compare their position in a worldwide ranking. On the other hand, Quest is frequent for entry-level events.[12]

12.3.2 Type and Complexity of Challenges

Another crucial aspect has to do with the technical content of the challenges to be included. For instance, CTF challenges may belong to various categories, depending on the security domain they refer to. Below, we briefly consider the main features of some common categories of challenges, with particular attention to their potential for implementing entry-level exercises. Also, we remark that these categories are not mutually exclusive as challenges may belong to more than one.

12.3.2.1 Binary

Binary challenges have to do with tasks such as reverse engineering and debugging of executables. These challenges always require the usage of some tool and a general understanding of how programs are written, compiled, and executed. Some very basic tools for binary challenges include, for instance, `strings` (which extracts ASCII strings from a file), `hexdump` (which shows the hexadecimal encoding of the binary), and `strace` (which monitors the system calls performed by a running program).

12.3.2.2 Network

This category includes challenges on network traffic inspection and protocol analysis. For instance, they might require to reconstruct the interactions between two or more computers from a fragment of recorded network traffic. These challenges

[11] https://ctftime.org/.

[12] For example, see https://capturetheflag.withgoogle.com/beginners-quest.

require at least some previous knowledge about the TCP/IP mechanisms and rely on tools such as Wireshark.[13]

12.3.2.3 OSInt

Open-source intelligence challenges revolve around collecting and processing data from online sources. Players may take advantage of some tools such as Maltego,[14] but in general, these challenges are meant to be solved without any specific technology. Furthermore, some of the data sources may be well known by most of the population, e.g., social media (SocMInt).

12.3.2.4 Crypto

Crypto challenges require tampering with ciphers, e.g., to reverse an encrypted text. Complex exercises may require advanced math, an understanding of algorithmic schemes, and scripting capabilities. Although tools may be useful, basic challenges, e.g., asking to break simple substitution ciphers, do not require advanced skills, and they can be solved with only pen and paper. Also, very little knowledge of ciphers, in general, is necessary to deal with them.

12.3.2.5 Web

These challenges ask for finding a flag hidden in a Web application, e.g., by exploiting a certain vulnerability. This category includes challenges on cross-site scripting and SQL injection. However, simple challenges may require inspecting the source code of a Web page, forging the parameters of an HTTP request, or inspecting the value of a cookie. This can be done by using a common Web browser, and it only requires some understanding of how the Web works.

12.3.2.6 Forensics

Loosely speaking, forensics exercises are about reconstructing a certain event from logs or from other files. For instance, logs can be from a certain OS or a recording of network traffic (network forensics). To carry out these challenges, the players must understand the log structure and, reasonably, the process that generated it.

[13] https://www.wireshark.org/.

[14] https://www.maltego.com/.

12.3.2.7 Stego

Steganography means hiding data in messages or in multimedia documents, e.g., pictures and audio files. In a stego challenge, the player has to reconstruct such hidden data. Sometimes, these challenges may require the usage of complex tools and a good knowledge of certain file formats. However, very basic challenges exist that only rely on some file features that most people might be aware of, e.g., metadata or text fonts. Often these challenges can be even solved with in-editor functionalities, e.g., by adjusting the colors of a picture.

12.3.2.8 Misc

All the challenges that cannot be mapped to one of the previous groups belong to the Misc category. Misc challenges are often related to some specific technology or sub-topic in computer science.

12.3.3 Previous Knowledge

Another goal is that of stimulating the curiosity of as many people as possible toward cybersecurity-related topics. To achieve that, we have to carefully consider the required previous knowledge. On the one hand, if challenges are only accessible to skilled players, most participants might find the game frustrating and opt out. On the other hand, if no skills are required, players might find it not challenging at all and get bored, especially when carrying out a long or repetitive task.

In general, we assume participants to have basic computer skills. These skills include the following:

- *Computer*. The player understands that computers internally perform operations and that certain tasks may be time-consuming.
- *File*. The player knows the most common file formats, e.g., pdf and txt. Also, they know how to browse directories in a file system.
- *Network*. The player roughly understands that computers are connected through networks using IP addresses.
- *Web*. The player can use a Web browser, follow hyperlinks, visit a URL, and query search engines.
- *Social*. The player is aware of the existence of the most famous social networks and has a basic understanding of how they work (e.g., they know what Facebook posts and tweets are).

- *English*. The player can read and understand a simple text, and they can follow written instructions.[15]

12.3.4 Teamwork

A further message we would like to convey is that, in most cases, cybersecurity is a team job, where people share their peculiar skills and mindset for achieving a common objective. For this reason, the game must allow small teams of people to participate. Furthermore, if possible, the game implementation should support single-player mode so to be employable in more contexts. For instance, to compensate the absence of a helping team, single players might have more time or hints on how to solve the challenges.

12.3.5 Scoring and Playability

In terms of playability, we aim at designing a challenging game that only some participants can conclude. At the same time, our goal is to provide people with an entertaining experience from which they can learn something. For instance, we might expect that around 20% of the players can finish the game and that less than 10% cannot solve even the first challenge. The scoring system should reflect the progresses done by the players through the game and, possibly, keep into account other factors, e.g., the amount of time required to finish the game.

12.4 A NERD DOGMA Design

Based on the requirements given in the previous section, below we introduce the design and gameplay of A NERD DOGMA. Briefly, A NERD DOGMA is a Quest CTF following a classical escape room game structure.

12.4.1 Game Plot

NERD corp. is an evil organization that aims at conquering the world. Currently, they plan to release their ultimate malware with the codename “A NERD DOGMA.”

[15] Although the game can be implemented in any language, we acknowledge English as the reference language for technical operations.

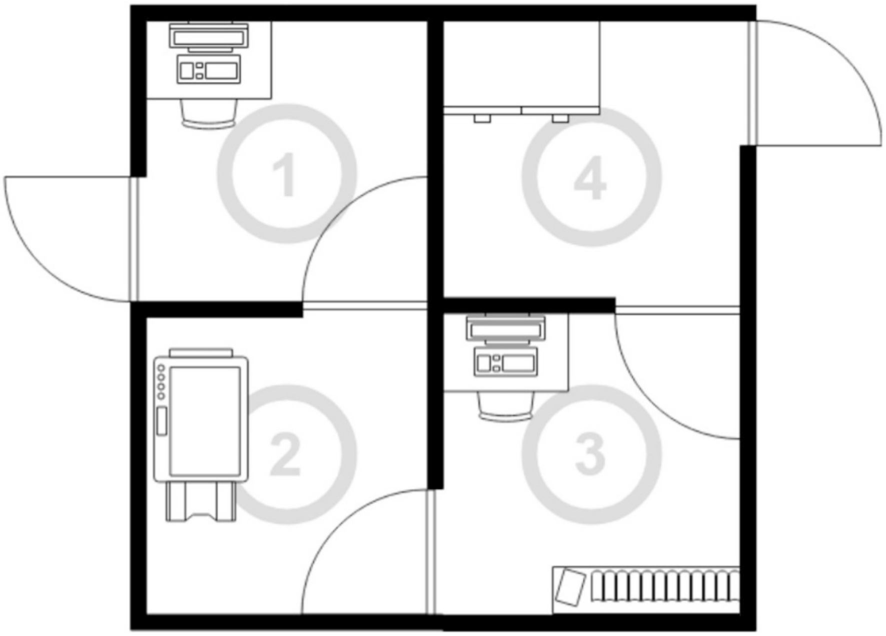


Fig. 12.1 Planimetry of the four rooms layout. From top left, in counterclockwise order: (1) secretary room, (2) copier room, (3) control room, and (4) archive room

However, the malware has a kill switch, which is a secret, alphanumeric code. The agents/players' mission is to retrieve the secret code.

To obtain it, the players must split into two teams that (i) physically infiltrate the NERD corp. HQ and (ii) remotely deal with the IT security facilities. The *on-field team* must collect evidence, move inside the headquarters, and perform actions, e.g., opening doors. Their equipment includes a UV flashlight to detect biological traces. On the other hand, the *remote team* operates through a terminal having the necessary security tools.

The HQ consists of four rooms. The general layout of the four rooms is shown in Fig. 12.1. Briefly, to complete the game, players have to make their way through the four rooms. The first three rooms, namely, the *secretary* room, the *copier* room, and the *control* room, have locked doors that must be opened by the on-field agents. Locks are controlled by PIN pads, and the unlocking PIN must be obtained by solving a game challenge. The last room, called *archive* room, has an emergency exit door. The on-field agents can leave anytime, but this will interrupt the mission. The last challenge is to find the kill switch in the archived documents. After leaving the HQ, players must reveal the kill switch to prove they solved the last challenge.

12.4.1.1 Challenges

Below we schematically describe the four challenges and provide their write-ups.

1. Secretary room	
Type:	OSInt
Hints:	Post-it #1: "Check the privacy policy of *SOCIAL*" (where *SOCIAL* is a social network); Post-it #2: "PIN = *NICK*" (where *NICK* is a nickname); Nameplate: "Ms. Eva Grandenaro"[a]
UV light:	Fingerprints on the PIN pad
Write-up:	Ms. Eva Grandenaro has an account on *SOCIAL*. A public post leaks her daughter's birth date, which is the PIN code
2. Copier room	
Type:	Stego
Hints:	Partially readable wastepaper that contains the address of a printing queue
UV light:	Fingerprints on the PIN pad
Write-up:	The printing queue is accessible, and it contains four pdf documents. One of the documents is a communication about the new PIN code. The PIN code is covered by a black overlay, which makes it unreadable but does not prevent text selection and copying
3. Control room	
Type:	Misc (Brute force)
Hints:	A network diagram showing the IP address of the electronic lock
UV light:	Fingerprints on the PIN pad
Write-up:	The PIN codes can be enumerated and tested one by one with a tool (see Sect. 12.5)
4. Archive room	
Type:	Crypto
Hints:	Four encrypted documents
Write-up:	One of the documents contains the final, secret code. All the documents are encrypted with Caesar cipher. The key can be obtained by noticing that all the documents begin with two fields, called "SUBJECT" and "CLASSIFICATION"

[a] Grandenaro is a free translation for Moneypenny in Italian

12.4.1.2 Scoring and Further Details

The game supports multiple scoring systems. We assume that players' score always ranges over [0, 100], where 100 denotes the highest score. Naively, one might assign 25 points for each room/challenge solved. However, this would result in a coarse-grained evaluation. A significant aspect is that the agents have a limited amount of time, e.g., 20 minutes, to retrieve the secret code from the HQ. Hence, a better scoring system will also consider the saved time, if any. For instance, we may opt for assigning 20 points for each challenge and 10 points for the saved time. The time points can be given for each minute remaining on the game timer. In addition, the scoring system may include a cost , e.g., 5 points, for hints, i.e., in-game suggestions on how to solve the current challenge. Thus, we should also consider that some players might ask for the first challenge's hint without solving it. In such a case, the final score would be negative, which is often unwanted.

Although different versions of the game have been implemented using distinct scoring systems, the default one follows the equation below:

$$Score = 10 + 20 \cdot Solves - 5 \cdot Hints + \min\{Time, 10\}$$

Here, $Solves$ is the number of solved rooms, $Hints$ that of requested helps, and $Time$ is the amount of (entire) minutes saved in case of game completion.

12.4.2 Compliance

Here we briefly revise the compliance of our design with regard to the objectives discussed in Sect. 12.3.

1. *Realism*. Our challenges are based on actual CTF ones, and thus they have the potential to offer a realistic experience. Clearly, this is also influenced by the actual implementation (e.g., tools). This aspect is further discussed in Sect. 12.5. Nevertheless, in principle, our challenges can provide a realistic experience. Furthermore, the escape room can contribute to making the game more immersive.
2. *Complexity*. The overall complexity of the proposed challenges might be categorized as *entry level*. Indeed, none of them require particular skills. Also, our challenges cover four of the common CTF categories described in Sect. 12.3.2, so that players can have a tasting of various security subjects.
3. *Knowledge*. No specific previous knowledge is needed. Players can solve the first two challenges by just using a regular Web browser and a pdf viewer. The last two challenges require the use of a terminal, and thus, it is important to provide them with a self-explanatory interface and adequate help. This is further explained in Sect. 12.5.
4. *Teamwork*. All the challenges must be solved by combining the on-field hints with remote support. This will stimulate the interaction between the two teams and test their communication skills.
5. *Playability*. The scoring system and the game time ensure that the game can be tuned for different types of audiences. Moreover, hints and even write-ups can be added so that players are allowed to obtain extra help in exchange for their collected points.

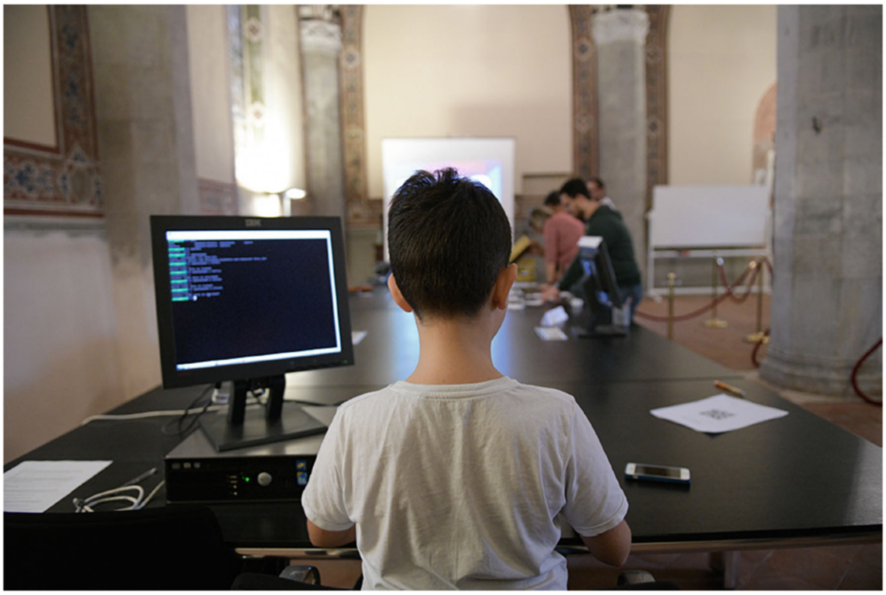


Fig. 12.2 First on-site proof of concept

12.5 Implementations

Starting from the design described above, different implementations have been realized, all sharing the same rooms/challenges, with some changes which have been introduced over time. In this section, we briefly present the various implementations, both physical and digital, in chronological order.

12.5.1 First Prototype (2019)

The first implementation of A NERD DOGMA was presented during Bright Night in September 2019.[16] The implementation consisted of a tabletop experience where most of the game operations were simulated by human supervisors (see Fig. 12.2). The remote team was provided with a Linux machine and a real terminal. The supervisor was in charge of helping the players with the terminal commands necessary to solve the challenges.

The four rooms were simulated by placing closed cardboard boxes along four segments of a long table. Each segment was delimited by red cords, and players

[16] http://www.bright-toscana.it/eventi-e-laboratori-lucca-2019/.

could access the next segment only after communicating the correct door PIN to the game supervisor. After that, the supervisor was in charge of removing the red cord and opening the hints box in the next room. Moreover, the supervisor was responsible for part of the storytelling, e.g., describing what the players would have seen by switching on the UV light.

The event was organized for teams of four members. Each team had 30 minutes to end the game, and extra hints were offered (for free) by the supervisors when teams got stuck in solving a challenge. The final score was computed as:

$$Score = 20 \cdot Solves + 2 \cdot \min\{Time, 10\}$$

Again, $Solves$ is the number of solved rooms, and $Time$ is the amount of (entire) minutes saved in case of game completion.

12.5.2 *First Video Game (2020)*

We started the implementation of the first video game in early 2020, and the first playable version went online in May 2020.[17] The online implementation has also the objective to deal with some limitations that emerged during the first event and make it accessible to a wider public. All in all, the main goal was to get rid of the human supervisor. Since her role was mainly related to helping the players with the terminal, we opted for a mock-up terminal, which we describe below.

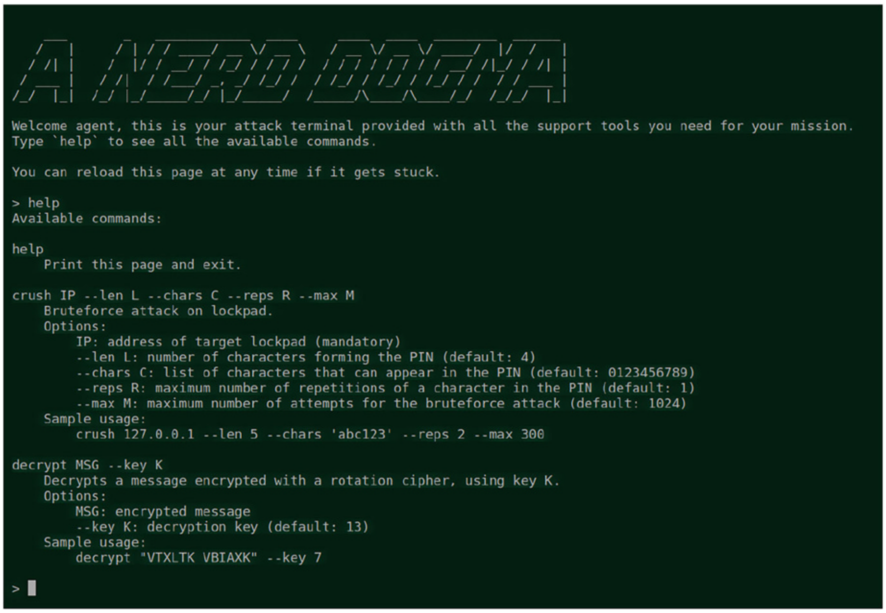
12.5.2.1 Terminal

The terminal is designed to resemble a real Unix terminal with a limited number of commands. To do this, we used JQueryTerminal.[18] Briefly, JQueryTerminal is a simulated, JavaScript-implemented terminal that entirely runs inside a Web browser. Each terminal command is coded through a JavaScript function as a member of the *terminal* object, and developers have to implement their own functions. Below, we show an excerpt of the terminal implementation for the command `help`.

```
$('body').terminal({
  help: function() {
    this.echo(/* supported commands list */);
  },
  /* Other commands implementation */
});
```

[17] https://anerddogma.it/.

[18] https://terminal.jcubic.pl/.



```
    ___       _   ____________  ____     ____  ____  ________  ______
   /   |     / | / / ____/ __ \/ __ \   / __ \/ __ \/ ____/  |/  /   |
  / /| |    /  |/ / __/ / /_/ / / / /  / / / / / / / / __/ /|_/ / /| |
 / ___ |   / /|  / /___/ _, _/ /_/ /  / /_/ / /_/ / /_/ / /  / / ___ |
/_/  |_|  /_/ |_/_____/_/ |_/_____/  /_____/\____/\____/_/  /_/_/  |_|

Welcome agent, this is your attack terminal provided with all the support tools you need for your mission.
Type `help` to see all the available commands.

You can reload this page at any time if it gets stuck.

> help
Available commands:

help
    Print this page and exit.

crush IP --len L --chars C --reps R --max M
    Bruteforce attack on lockpad.
    Options:
        IP: address of target lockpad (mandatory)
        --len L: number of characters forming the PIN (default: 4)
        --chars C: list of characters that can appear in the PIN (default: 0123456789)
        --reps R: maximum number of repetitions of a character in the PIN (default: 1)
        --max M: maximum number of attempts for the bruteforce attack (default: 1024)
    Sample usage:
        crush 127.0.0.1 --len 5 --chars 'abc123' --reps 2 --max 300

decrypt MSG --key K
    Decrypts a message encrypted with a rotation cipher, using key K.
    Options:
        MSG: encrypted message
        --key K: decryption key (default: 13)
    Sample usage:
        decrypt "VTXLTK VBIAXK" --key 7

> █
```

Fig. 12.3 A NERD DOGMA remote team terminal

The behavior of the command is defined in the function `body`. In this case, the function has no parameters, and it simply outputs a text before terminating. The printed text (omitted here for brevity) is a help message explaining what the supported commands are and how to use them (see Fig. 12.3).

The other available commands are `crush` and `decrypt`. The `crush` command has a few inputs, i.e., the IP address of the target lock, the PIN length (`--len`), the PIN charset (`--chars`), and the maximum number of repetitions of each character (`--reps`). The implementation verifies that the correct IP address of the target electronic lock is passed as an input. In that case, it enumerates all the strings that comply with the given parameters. To simulate the PIN submission process, a 0.1-second delay is forced at each cycle. This delay ensures that only a few seconds are required when the right parameters are used, while using the wrong parameters makes it impossible to find the right PIN before the game times out. When the correct PIN is generated, the command terminates with a *success* message. The correct command for solving the challenge of the third room is `crush 10.187.51.1 --len 6 --chars 137 --reps 2`.

Finally, the `decrypt` command rotates the characters of the given message according to the parameter `--key`. For simplicity, messages are all capitalized, and only letters are rotated. All the documents appearing in the fourth room are rotated by 16 positions. Thus, the command for obtaining the cleartext from *message* is `decrypt` *message* `--key 16`.

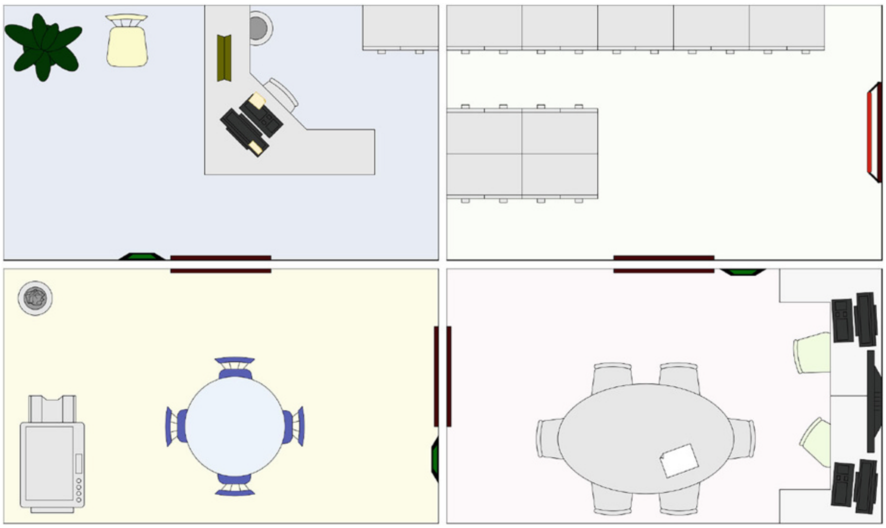


Fig. 12.4 The four rooms layout of the first version of the video game

12.5.2.2 Core Game

The core of the online version is a point-and-click game developed in Godot,[19] a lightweight, open-source game development framework. By default, Godot games can be exported for the most common desktop and mobile OSes as well as for HTML5 Web platforms. The core game consists of (i) a welcome screen, (ii) the four rooms, and (iii) a game over screen. Each screen is implemented as a Godot *scene*. Loosely speaking, a scene is a software component implementing a model-view-controller (MVC) pattern. The model is coded through a class implemented in Godot script, i.e., a python-like programming language. The view is a graphic user interface created through the Godot graphic editor that embeds the user controllers, e.g., buttons and text input fields. The overall game logic is implemented through scene transitions that occur when certain events are triggered, e.g., when the right PIN is inserted.

Figure 12.4 shows the Godot scenes for the rooms (following the layout of Fig. 12.1). Instead, Fig. 12.5 shows the Godot controls of a PIN pad and its appearance when the UV light control is triggered. All in all, the game has controls for:

- Inspecting the in-game hints
- Interacting with the PIN pads
- Activating/deactivating the UV light on PIN pads

[19] https://godotengine.org/.

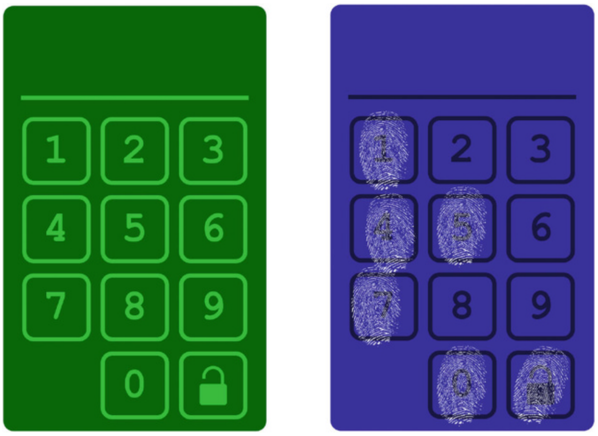


Fig. 12.5 The PIN pad of a door (left) and the effect of applying the UV flashlight (right)

- Moving between two adjacent rooms
- Opening the game terminal in a new tab of the browser
- Visualizing the extra hint in each room

12.5.2.3 Further Game Components

The last two elements needed to implement the full game are the social account of Ms. Eva Grandenaro (secretary room) and the printing queue (copier room). For the social media account, we opted for a Facebook profile. The profile has a single, public post following the specifications given in Sect. 12.4.1. Instead, the printing queue has been implemented as a directory managed by an Apache HTTP server.[20]

12.5.3 First Physical Room (2020)

Almost contemporarily to the video game implementation, we proposed another physical implementation of A NERD DOGMA during the 4th Italian Conference on Cybersecurity (ITASEC) [10] in February 2020. Roughly, the installation (see Fig. 12.6) resembled that of Bright Night 2019, with few, significant improvements.

Briefly, the peculiar aspects of this installation were the following:

- The game was implemented as a single room, delimited by movable panels. This room contained the hints of the secretary's room and three electronic safe boxes.

[20] https://intranerd.it/printers/queue37.

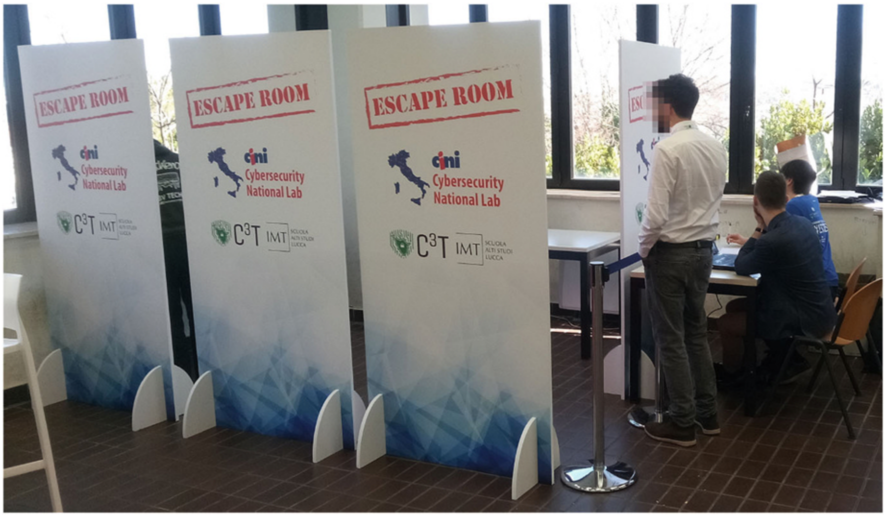


Fig. 12.6 A four-player team playing during ITASEC 2020

- Each electronic safe box contained the hints of one of the other three rooms.
- The UV light was simulated by placing QR codes on top of some objects in the room. Each code, when scanned with a QR code reader, returned a text message describing what the UV light shows, e.g., "Fingerprints appear on digits 1,3,7."

12.5.4 Second Video Game (2021)

To improve the gaming aspects of the first video game, in 2021, a group of master students was asked for reimplementing A NERD DOGMA from its specifications. Due to its relevance, the students decided to use the Unity game engine, a mainstream framework widely adopted in the video game industry. Again, this version of the video game was deployed as a Web application.[21] The main differences between this version and the previous one are the following:

12.5.4.1 3D Graphics

As shown in Fig. 12.7, a 3D graphic with four scenes built with a customized Unity asset and interactable objects was used in this case. This aims at making the experience more immersive and giving a better characterization of the rooms

[21] Now available at https://anerddogma.it/.

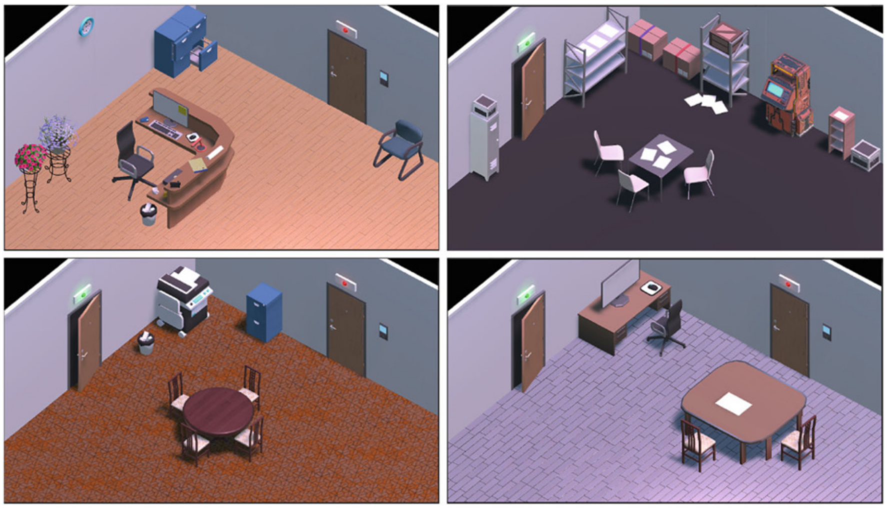


Fig. 12.7 The four rooms layout in the 3D version of A NERD DOGMA

and their nature. As a matter of fact, clearly identifying that they are in a certain environment helps the players understand the current challenge and how to approach the solution. Also, most people might find it more pleasant to interact with a curated interface and be more motivated to see how the game progresses.

12.5.4.2 Support for Multi-language and Difficulty Levels

Another improvement is that now the game supports two languages, i.e., Italian end English. This configuration can be selected before starting the game.

Contemporary, players can pick one between two difficulty levels, i.e., beginner and expert. The main difference between the two is that beginners also have access to the write-ups. A write-up provides full details on how to solve the current challenge, but it also voids the points gained for solving it. As a consequence, the scoring mechanism is changed accordingly, and the final score is computed as follows:

$$Score = 20 \cdot Solves - 16 \cdot Solutions + \min\{Time - 2 \cdot Hints, 10\}$$

Here, reading a write-up ($Solutions$) decreases the score of the corresponding room of 16 points (corresponding to 80% of the room value), while reading a hint ($Hints$) decreases the remaining time of 2 minutes. The game lasts 30 minutes.

12.5.4.3 Social Account

The last difference regards the social network used in the first challenge. In this version, Ms. Eva Grandenaro has a Twitter account (rather than a Facebook one). The reason behind this choice is twofold. First, most young players showed little confidence in using Facebook. Furthermore, Facebook is now restricting the accessibility to public posts so that only registered people can see them (while this is not required by Twitter). Since we prefer not to assume players to be registered to any particular service, we opted for dropping Facebook.

12.5.4.4 Terminal

In the first prototype of the 3D version, a new terminal was re-implemented inside the game to better integrate the command line hacking tasks with the 3D environment. Afterward, we decided to reuse the JavaScript terminal introduced in Sect. 12.5.2. This choice will guarantee easier code maintainability in case of the addition of new rooms/challenges. Indeed, the JavaScript terminal, already used in the 2D version, is also adopted in the recent mobile room presented in the next section. In case of the addition of new commands, we will have a single application to update.

12.5.5 Mobile Room (2022)

The last version of A NERD DOGMA is a 5 m × 5 m (2.30 m tall) installation (see Fig. 12.8). In this implementation, all the game aspects have been physically implemented. Hence, for instance, room doors are actually locked with six-digit PIN pads. Room walls are decorated, and actual furniture can be placed inside to make every event slightly different. More importantly, the entire escape room can be disassembled and moved in order to carry the game to different locations and, in

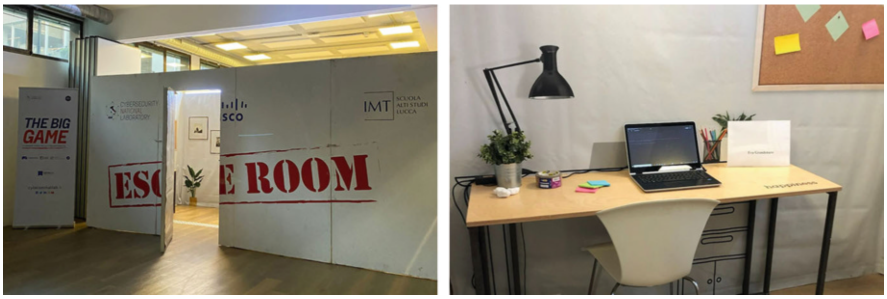


Fig. 12.8 External (left) and interior (right) of the last physical implementation of A NERD DOGMA

care, substitute damaged elements. When disassembled, the escape room is stored in a 2.30 m × 1.25 m × 1.50 m box, which weighs approximately 500 Kg.

12.6 Participation to Events

In [4], we presented some experimental activities that we carried out on and with A NERD DOGMA. Those experiments aimed at testing and measuring the effectiveness of the game in introducing cybersecurity to inexpert players. In this section, we report all the other events where the escape room was employed, and we discuss the lesson learned.

12.6.1 Physical Installations

As stated in Sect. 12.5, A NERD DOGMA was first introduced at Bright Night 2019 (as a tabletop game) and then at ITASEC 2020 (as a single room). Thanks to the movable room implementation, the game has been also presented in other venues more recently. For instance, in 2022, the game was included among the training activities of the Cybertrials,[22] i.e., a free training program for high-school female students. In particular, the movable room was included in the final event organized in Turin (see Fig. 12.9).

Another event where the escape room recently appeared was the Wired Next Fest,[23] October 2022, in Milan. There the attendance was free, upon registration, and open to the general audience present at the event. Finally, the last installation of A NERD DOGMA occurred at Lucca Comics and Games 2022,[24] i.e., the second world's largest comic festival.

12.6.2 Video Game Adoption

As detailed in [4], during the COVID-19 lockdown, the video game was used in online events for high school students. After the lockdown, other open days with high school students were (and will be) organized, either in schools' labs or online, to reach a large audience of students potentially interested in computer science and cybersecurity.

During these events, we collect informal feedback from students, and we report here some positive comments like *"Prepare many more rooms !!!!!!!"*[25] and

[22] https://www.cybertrials.it/.

[23] https://nextfest2022-milano.wired.it/.

[24] https://www.luccacomicsandgames.com/en/2022/home/.

Fig. 12.9 The escape room presentation at the Cybertrials final event in Turin (May 2022)

"Perhaps after one finds a solution give a bonus for 2 extra minutes. Anything else is perfect and beautiful".[26]

Some students also asked for extra time: *"In our opinion, more time should be given"*[27] and to *"Being able to copy and paste the final text"*[28] in the last room, with the documents encrypted with Caesar cipher. Of course, players would like to complete all the rooms, and changing these settings would mean further simplifying the game. But as we already stated in Sect. 12.3.5, we aimed at designing an engaging and challenging game that only some participants can conclude.

Besides schools, the video game was included again in Bright Night 2020, 2021, and 2022. There, A NERD DOGMA was proposed to the visitors. Since the online game can scale on a large number of participants, we did not need to set up any reservation process. A similar event was presented during Lucca Comics and Games

[25] Original (Italian): *"Fare molte più stanze !!!!!!!"*.

[26] Original (Italian): *"Magari dopo che trovi una soluzione dare un bonus per esempio 2 minuti in più di tempo. Per il resto tutto perfetto e bello"*.

[27] Original (Italian): *"Secondo noi bisognerebbe dare più tempo"*.

[28] Original (Italian): *"Poter copiare e incollare il testo finale"*.

2021, inside the Game Science Research Center exposition area,[29] and for the European Researchers' Night[30] 2022 in Genova.

12.6.3 Lesson learned

In the last four years, we have exploited A NERD DOGMA for a number of activities and events. Thousands of people played the game, either online or in person. Our experience, also experimentally assessed in [4], confirms that gamification is a powerful tool. In particular, this activity showed that also serious and highly technical topics, e.g., those related to cybersecurity, can be introduced with properly designed games. Although designing such games is nontrivial, the systematic assessment of the state of the art helped us in taking reasonable design and implementation choices. Among them, the neat decoupling between the game design and its actual implementation allowed us to develop different versions of the game. Furthermore, the Quest CTF approach provided us with interesting opportunities in terms of challenge substitution and revision, without compromising the overall structure of the game. Finally, the escape room setting supported the strong engagement of participants without interfering with the design and implementation of each challenge.

In terms of validation, we still have to carry out an in-depth analysis of our design. As a matter of fact, although feedback and data have been collected for some implementations, we still do not have a methodology that allows us to aggregate such information. As a consequence, a quantitative analysis of, e.g., the effectiveness of A NERD DOGMA is yet to come, and we consider it as future work.

12.7 Conclusion

In this chapter, we presented the design and implementation of A NERD DOGMA, a Quest CTF combined with an escape room that hosts four entry-level, yet realistic, cybersecurity challenges. Our work started with a systematic revision of the game objectives that resulted in a list of specifications. Thus, we designed A NERD DOGMA and its challenges in order to satisfy the specifications, and eventually, we implemented the game in various forms. These implementations were presented in various contexts for both online and in-person training activities. In all cases, no technical issues were reported, and we collected feedback from the players.

[29] https://sites.google.com/imtlucca.it/gamescience/events/programma-lcg2021.

[30] https://www.sharper-night.it/evento/gioco-escape-room-a-nerd-dogma/.

This helped us draw some statements about the game, its features, and potential improvements.

Our research confirms that gamification is a viable solution for the provisioning of serious and technical content to a wide audience of unskilled people. Nevertheless, as we highlighted, there are several caveats that game designers must consider. Although modern development frameworks, e.g., Unity and Godot, can support software design and implementation, the actual contents must be carefully studied as, for instance, some of them can be external to the implementation. The open-source design of A NERD DOGMA also aims at mitigating this issue by allowing challenges to be maintained over time, without redesigning the entire game.

Acknowledgments This work was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU-NGEU.

References

1. Ahmad, A., Zeshan, F., Khan, M.S., Marriam, R., Ali, A., Samreen, A.: The impact of gamification on learning outcomes of computer science majors. ACM Trans. Comput. Educ. **20**(2) (2020)
2. Beltrán, M., Calvo, M., González, S.: Experiences using capture the flag competitions to introduce gamification in undergraduate computer security labs. In: 2018 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 574–579. IEEE, Piscataway (2018)
3. CD Projekt Red. Cyberpunk 2077. PlayStation 4, PlayStation 5, Xbox One, Xbox Series X/S, Microsoft Windows, Stadia (2020)
4. Costa, G., Lualdi, M., Ribaudo, M., Valenza, A.: A NERD DOGMA: introducing CTF to non-expert audience. In: Proceedings of the 21st Annual Conference on Information Technology Education, SIGITE '20, pp. 413–418. Association for Computing Machinery, New York (2020)
5. Dabrowski, A., Kammerstetter, M., Thamm, E., Weippl, E., Kastner, W.: Leveraging competitive gamification for sustainable fun and profit in security education. In: 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 15), Washington, DC. USENIX Association, Berkeley (2015)
6. Demetrio, L., Lagorio, G., Ribaudo, M., Russo, E., Valenza, A.: ZenHackAdemy: Ethical Hacking @ DIBRIS. In: Proceedings of the 11th International Conference on Computer Supported Education, CSEDU 2019, Heraklion, Crete, May 2–4, 2019, vol. 1, pp. 405–413. SciTePress, Setúbal (2019)
7. Jin, G., Tu, M., Kim, T.-H., Heffron, J., White, J.: Game based cybersecurity training for high school students. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE '18, pp. 68–73. Association for Computing Machinery, New York (2018)
8. Le Compte, A., Elizondo, D., Watson, T.: A renewed approach to serious games for cyber security. In: 2015 7th International Conference on Cyber Conflict: Architectures in Cyberspace, pp. 203–216. NATO CCD COE Publications, Tallinn (2015)
9. Lehrfeld, M., Guest, P.: Building an ethical hacking site for learning and student engagement. In: SoutheastCon 2016, pp. 1–6. IEEE, Piscataway (2016)
10. Loreti, M., Spalazzi, L. (Eds.) Proceedings of the Fourth Italian Conference on Cyber Security, Ancona, February 4th to 7th, 2020, vol. 2597. CEUR Workshop Proceedings. CEUR-WS.org (2020)

11. McDaniel, L., Talvi, E., Hay, B.: Capture the flag as cyber security introduction. In: 2016 49th Hawaii International Conference on System Sciences (HICSS), pp. 5479–5486 (2016)
12. Mello-Stark, S., VanValkenburg, M.A., Hao, E.: Thinking Outside the Box: Using Escape Room Games to Increase Interest in Cyber Security, pp. 39–53. Springer International Publishing, Berlin (2020)
13. Montreal, U.: Watch dogs. Microsoft Windows, PlayStation 3, PlayStation 4, Xbox 360, Xbox One, Wii U (2014)
14. Morelock, J.R., Peterson, Z.: Authenticity, ethicality, and motivation: a formal evaluation of a 10-week computer security alternate reality game for CS undergraduates. In: 2018 USENIX Workshop on Advances in Security Education (ASE 18), Baltimore. USENIX Association, Berkeley (2018)
15. Papaioannou, T., Tsohou, A., Bounias, G., Karagiannis, S.: A constructive approach for raising information privacy competences: the case of escape room games. In: Katsikas, S., Furnell, S. (Eds.) Trust, Privacy and Security in Digital Business, pp. 33–49. Springer International Publishing, Berlin (2022)
16. Pappas, G., Peratikou, P., Siegel, J., Politopoulos, K., Christodoulides, C., Stavrou, S.: Cyber escape room: an educational 3d escape room game within a cyber range training realm. In: INTED2020 Proceedings, 14th International Technology, Education and Development Conference, pp. 2621–2627. IATED, 2–4 March (2020)
17. Švábenský, V., Vykopal, J., Čeleda, P.: What are cybersecurity education papers about? A systematic literature review of SIGCSE and ITiCSE conferences. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, SIGCSE '20, pp. 2–8. Association for Computing Machinery, New York (2020)
18. Tobey, D.H., Pusey, P., Burley, D.L.: Engaging learners in cybersecurity careers: lessons from the launch of the National Cyber League. ACM Inroads **5**(1), 53–56 (2014)
19. Trobbiani, M.: Hacknet. Microsoft Windows, macOS, Linux, PlayStation 4, Xbox One, Nintendo Switch, iOS, Android (2015)
20. Vigna, G.: The 2010 international capture the flag competition. IEEE Sec. Privacy **9**(1), 12–14 (2011)

Chapter 13
Grand Challenges in Software Engineering for Games in Serious Contexts

Antonio Bucchiarone

Abstract The potential benefits of using the engaging and interactive nature of games to achieve specific objectives have been recognized by researchers and professionals from various domains. Serious games have been developed to impart knowledge, skills, and awareness in areas such as education, healthcare, and the environment, while gamification has been applied to enhance the engagement, motivation, and participation of users in non-game activities such as sustainability and learning. As a result, the fields of game design, software engineering, and user experience are increasingly converging to create innovative solutions that blend the strengths of games with real-world applications.

The main goal of this book has been to foster an environment of collaboration that unites experts from both the software engineering and game development communities. The primary aim has been to facilitate knowledge sharing, exchange of experiences, and interdisciplinary perspectives to explore the latest opportunities, challenges, costs, and benefits associated with games in serious contexts. Additionally, the book seeks to establish a fresh research agenda that aligns with the emerging trends and issues in the field.

The aim of this chapter is to provide an overview of the major challenges that must be addressed by the software engineering and game development communities to fully realize the potential of serious games and gamification in various domains.

Keywords Software engineering · Games in serious context · Grand challenges · Research roadmap

A. Bucchiarone (✉)
Motivational Digital Systems (MoDiS), Fondazione Bruno Kessler (FBK), Trento, Italy
e-mail: bucchiarone@fbk.eu

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_13

13.1 Introduction

This chapter attempts to summarize the discussions of the chapters presented in this book capturing a vision of the grand challenges facing the two communities of software engineering and games. This analysis has the unique objective to provide useful context for future research challenges and directions. We start with a summary of the key challenges presented in the overall chapters of the book, and we conclude with a brief summary of where we believe the field of software engineering for games in serious context (GSC) is going.

As we have seen from the concrete application scenario introduced in this book, GSC is gaining popularity in all those domains that would benefit from the increased engagement of their target users [1]. Thus, these applications are found in disparate contexts, such as education and training [2–6], health and environmental awareness [7–10], e-banking [11], software engineering [12], everyday challenges [13], and so forth.

The growing adoption of GSC experiences make their design and development increasingly complex due to, for example, the number and variety of users, and their potential mission criticality. This complexity is nurtured, among other factors, by a lack of theoretical grounding and adequate frameworks to engineer the intended solutions. One of the main challenge in this context is to bring the attention of interdisciplinary researchers and practitioners to the opportunities and challenges involved in the new trends and issues related to the development of GSC applications.

13.2 Grand Challenges

This section describes the challenges that have emerged in the various chapters of this book. We start from the set of challenges more related to the engineering of games in serious context (GSC), and we conclude with a set of challenges where the use of GSC could improve the software engineering aspect.

13.2.1 Design of GSC

The design of GSC is quite complex and requires numerous precautions in order to achieve a well-functioning system. In fact, according to several data in the literature [14, 15], there is little cohesion with respect to theoretical underpinnings and what gamification encompasses, leading to inconsistent results related to the use of such systems. These results can be partly explained by the lack of standardized design methodologies and the extensive use of the shortcoming one-size-fits-all strategy [16]. In other words, they are often designed without taking into account that different categories of users have different interactions with these systems. To face these results' inconstancy, several solutions have been presented. Some authors suggest that the design of GSC should take into account the final users' differences and preferences [15, 17–20], while other authors have presented specific design frameworks in order to properly design GSC [21–23] taking into account specific needs. To summarize, we need to consider to whom the GSC is directed and what the characteristics of the target group are [24]. Indeed, personalized interactive systems are more effective than one-size-fits-all approaches [18].

13.2.2 Context-Awareness in GSC

In GSC, the term *context* is often associated with *user* and *goals*; actually, it is undeniable that there is a link between these various factors. Therefore, the way in which GSC is perceived by users depends on multiple factors, including the individual characteristics of the users, the context in which the GSC is implemented, and the specific task or activity being gamified. These elements all contribute to how users perceive and engage with GSC [15]. Despite that, contextual factors and the importance of the application domain are often underestimated in GSC research and design [15]. Therefore, according to Koivisto and Hamari [15], the lack of theoretical understanding surrounding the importance of the contextual influence on gamification effectiveness might produce results that in reality cannot be generalized to other contexts.

13.2.3 User Experience Evaluation Methodologies and Tools

User experience (UX) is a multifactorial concept that is difficult to be measured [25]. In GSC, we are especially interested in the differences between traditional research and emerging evaluation of UX, such as physiological data (e.g., electroencephalography, electromyography, and facial expression assessment) [26]. Since the purpose of a user experience evaluation is to record and interpret the experience experienced by users while interacting with a digital game, it is imperative that this recording is accurate and reliable in order for its results to have substance and be useful. When measuring and evaluating user experience in GSC, it is best to employ tools from different methodologies, such as quantitative tools combined with qualitative evaluation tools or objective tools combined with qualitative evaluation tools. Taking advantage of each methodology in this way will increase the reliability of the results. Utilizing tools from only one methodology may negatively affect our evaluation efforts if we choose to leverage those tools. Finally, for better understanding and in order to interpret the experience derived from a GSC, the methodology used to evaluate the user experience plays a very important role. Future research will evaluate GSC using different methodologies and tools. These studies will ultimately be aimed at finding the most effective combination of tools and methodologies for measuring the GSC potential.

13.2.4 Software Reuse in GSC

Creating a GSC can be very complicated, with many activities, elements, and team members composing this development, taking a long time to be produced. Reuse is the concept by which it aims to build some artifact from one that has already been produced to save time and money. The gaming community has already been using reuse concepts in an ad hoc manner to create new games from existing ones. Reuse can bring some advantages for the development of GSC, such as greater longevity, lower production costs, and greater diversity of solutions being created in a shorter time. In this context, *componentization* can be used to simplify the complexity of a GSC by clearly identifying and separating the concerns. This allows for easier and longer-lasting revision management. At the same time, the *Product Line* paradigm can be exploited to reuse some GSC features and create several branches from it. Finally *Model-Driven Development (MDD)* can be used to derive the characteristics of a GSC and create models from them [27]. Once the models are already created, transformations can be applied to generate a new model that will generate new adaptations of the GSC in the future.

13.2.5 Quality Design in GSC

The development of GSC implies a learning process like the process of incorporating new learning into memory, as well as retrieving and using it. This requires a software architecture designed to optimize memory and processing resources [28]. The design patterns approach offers holism and efficiency; another important advantage of this approach is that it provides reusable solutions, which benefit the maintainability and evolution of the system [29]. In the case of devices with limited storage, such as mobile phones, they allow the optimization of resources. GSC has a greater fluidity in its operation; the player's experience when interacting with the user interface is a motivation to continue looking for efficient solutions that improve the use of resources and thus the speed of response with which the exercises to be solved are presented, leading to the development of an application agile and efficient to make your learning fun, dynamic, and permanent. For these reasons, the software engineering community must contribute studies and techniques to improve the performance of these applications.

13.2.6 Adaptation in GSC

In addition to enhancing design phases, GSC need a support for monitoring and adaptation of the gameplay. In fact, there exist concrete risks for GSC of triggering and producing undesirable side effects [30]. In most of the current approaches, the design, analysis, and revision of GSC require many development activities often unrelated to each other, with the use of various general-purpose languages (e.g., rule-based). The different actors involved (e.g., domain expert, system developer, and impact managers) use different languages and tools to execute their tasks with a completely different understanding of the game concepts and their relations. In turn, this might lead to managing unexpected game deviations with ad hoc and not reusable solutions, making the monitoring and the revision of game mechanics and dynamics a complicated task. What is really needed is the provision of uniform and clean datalogs of players' game actions. In this way, a desired monitoring framework would also assist and enhance the process of monitoring gameplay, aimed at detecting and resolving upcoming design issues at runtime. Such a tool would allow an iterative, player-centric design, in contrast to a one-size-fits-all strategy, notoriously detrimental [13, 31, 32]. Instead, adaptive content is likely to increase player engagement and motivation when it is aligned to players' preferences [33] and adjusts its difficulty to the players' skills and abilities [34]. Adapting and personalizing the gameplay to the user is more likely to foster intrinsic motivation, challenging to achieve as is notably subjective to the player [35–37]. The development of such an *adaptive by design* GSC and the addition of adaptation to existing GSC are both challenging engineering problems, which require a combination of expertise in the learning domain, game development, software development, and machine learning.

13.2.7 Abstraction and Automation in GSC

A fundamental concern of gameful applications is their tailoring to the target domain and users: if a game is detached from the domain interests, the risk is to promote counterproductive/undesired behaviors; similarly, too easy or too complex games could fail engagement objectives due to loss of interest or discouragement, respectively [38]. A direct consequence of the mentioned tailoring needs is the critical contribution and cooperation of application domain and gamification experts: the former ones provide inputs about the engagement issues and desired outcomes, while the latter ones propose corresponding gamification strategies. Such a cooperation conveys gameful application specifications to be implemented in an appropriate target platform.

In the current state of practice, one available implementation option is to pick up a pre-packaged gamification application from a repository [39]. The advantage would be to have a quick development phase limited to configuration purposes, at the price of very limited customization possibilities, unless manually tuning the existing implementation. Diametrically opposite, a completely new gamified application can be developed from scratch: this solution necessarily entails longer time to market, with the advantage of realizing a fully customized implementation. Regardless of the choice, the realization and deployment phases introduce an abstraction gap between gamification stakeholders, namely, domain and gamification experts, and the gameful application itself. In fact, the target application is typically implemented as a collection of rules matching incoming event notifications with corresponding game status updates. Therefore, developers need to translate game mechanics and other elements into corresponding rules while the other stakeholders are required to backtrack state changes into corresponding gaming events.

With the growing adoption of gamification in disparate application domains and its spreading to a wider range of users, the complexity of gameful software is unavoidably increasing. In this respect, the abstraction gap between design and realization becomes a critical issue: the implementation phase is more tedious and error-prone, due to the number of rules and the customization needs. Moreover, maintenance and evolution activities are harder to manage, due to the disconnection between design and realization.

In order to close the gap between design and implementation of gameful applications, abstraction is a key aspect that should be taken into account. A developer should use a set of domain-specific languages devoted to the specification, implementation, and deployment of gameful applications, and more in general, a software engineering process should consider the following key aspects:

Separation of concerns: a gamification approach can be described by means of several perspectives. When the complexity grows, an effective way to alleviate it is to manage different perspectives as separate points of view that are later on fused into a complete solution;

Correctness by construction: given the growth of gamification employment and range of its potential users, the specification of gameful applications becomes increasingly intricate. In this respect, game rules shall be consistent with mechanisms and elements intended for the target application;

Automation: in order to close the gap between design and implementation, the amount of manually written code shall be reduced as much as possible. Or in the other way around, the degree of automation provided by the process shall be maximized.

13.2.8 GSC for Software Engineering Education and Training

Gamification means creating a game narrative that guides players through increasingly complex challenges, keeping them engaged with social activities such as group work or competitions. It means providing immediate feedback and students taking autonomous choices to progress down the individually decided path. Gamification is not an add-on. Instead, gamification mechanics are fundamental to the learning path personalization process in two ways. Not only do they keep the students engaged, but they can also be used as tools to gain insight into the student's behavior from a different perspective and thus help generate a more personalized and engaging learning path. In order to increase engagement, the gamification mechanics must be calibrated according to the underlying activities. That is why gamification mechanics should be enhanced by AI techniques to make the motivation personalized and contextualized [40].

13.2.9 GSC for Software Quality

Software development projects often fail because of insufficient code quality [41]. It is now well documented that the task of testing software, for example, is perceived as uninteresting and rather boring, leading to poor software quality and major challenges to software development companies. One promising approach to increase the motivation for considering software quality is the use of gamification. Initial research works already investigated the effects of gamification on software developers and come to promising [42]. Nevertheless, a lack of results from field experiments exists, which motivates the need of new research in this field. Preliminary results in this direction [42] show that the introduction of a leaderboard game has a measurable effect on the Code Quality (CQ) in software development projects, while further questions for future research arise. The leaderboard can be used more intensively in teaching. In addition, it needs to be evaluated in a professional context with experienced developers. Furthermore, the degree of gamification needs to be investigated. How much is too much or too little? The optimal degree of gamification is an aspect that should be investigated more closely in future research works. The time spent on gamification can also be considered, which leads to the question of how much time should or can be spent in order to achieve the best possible results in Code Quality. In terms of motivation, it could be analyzed whether competition with others, the own performance, or the feeling of playing as a team contributes the most. In the context of a multiplayer approach, it could be considered how this affects the player motivation and outcome.

13.3 Final Discussion

In this chapter, we presented the grand challenges that cover the two main perspectives covered by this book: (i) software engineering for games in serious contexts and (ii) games in serious context for software engineering. We hope that this analysis not only represents a snapshot of the challenges faced in these research fields but contributes to stimulate researchers, practitioners, and tool developers to tackle and explore some of them. At the same time, it provides a useful context for future research projects, research grant proposals, and new research directions. We hope in a few years we can look back at this list and see many of them crossed out as a sign of the continuous advancement and maturity of these two communities together.

References

1. Koivisto, J., Hamari, J.: The rise of motivational information systems: a review of gamification research. Int. J. Inf. Manag. **45**, 191–210 (2019)
2. Dicheva, D., Dichev, C., Irwin, K., Jones, E.J., (Boots) Cassel, L., Clarke, P.J.: Can game elements make computer science courses more attractive? In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE 2019, pp. 1245 (2019)
3. Cosentino, V., Gérard, S., Cabot, J.: A model-based approach to gamify the learning of modeling. In: Proceedings of the 5th Symposium on Conceptual Modeling Education and the 2nd International iStar Teaching Workshop Co-located with the 36th International Conference on Conceptual Modeling (ER 2017), Valencia, November 6–9, 2017, pp. 15–24 (2017)
4. Kim, S., Song, K., Lockee, B., Burton, J.: Gamification in Learning and Education: Enjoy Learning Like Gaming. Springer International Publishing, Berlin (2018)
5. Lee, J.J., Hammer, J.: Gamification in education: what, how, why bother? Acad. Exch. Q. **15**(2), 2 (2011)
6. Bucchiarone, A., Cicchetti, A., Bassanelli, S., Marconi, A.: How to merge gamification efforts for programming and modelling: a tool implementation perspective. In: 2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 721–726. IEEE, Piscataway (2021)
7. Johnson, D., Deterding, S., Kuhn, K.-A., Staneva, A., Stoyanov, S., Hides, L.: Gamification for health and wellbeing: a systematic review of the literature. Int. Intervent. **6**, 89–106 (2016)
8. Marconi, A., Schiavo, G., Zancanaro, M., Valetto, G., Pistore, M.: Exploring the world through small green steps: improving sustainable school transportation with a game-based learning interface. In: Proceedings of the 2018 International Conference on Advanced Visual Interfaces, AVI 2018, pp. 24:1–24:9 (2018)
9. Rajani, N.B., Mastellos, N., Filippidis, F.T.: Impact of gamification on the self-efficacy and motivation to quit of smokers: observational study of two gamified smoking cessation mobile apps. JMIR Serious Games **9**(2), e27290 (2021)
10. Vieira, V., Fialho, A., Martinez, V., Brito, J., Brito, L., Duran, A.: An exploratory study on the use of collaborative riding based on gamification as a support to public transportation. In: 2012 Brazilian Symposium on Collaborative Systems, pp. 84–93. IEEE, Piscataway (2012)
11. Rodrigues, L.F., Costa, C.J., Oliveira, A.: Gamification: a framework for designing software in e-banking. Comput. Hum. Behav. **62**, 620–634 (2016)
12. Pedreira, O., García, F., Brisaboa, N., Piattini, M.: Gamification in software engineering – a systematic mapping. Inf. Softw. Technol. **57**, 157–168 (2015)
13. Vassileva, J.: Motivating participation in social computing applications: a user modeling perspective. User Model. User Adapt. Interact. **22**(1), 177–201 (2012)
14. Seaborn, K., Fels, D.I.: Gamification in theory and action: a survey. Int. J. Hum. Comput. Stud. **74**, 14–31 (2015)
15. Koivisto, J., Hamari, J.: The rise of motivational information systems: a review of gamification research. Int. J. Inf. Manag. **45**, 191–210 (2019)
16. Böckle, M., Micheel, I., Bick, M., Novak, J.: A design framework for adaptive gamification applications. In: Proceedings of the 51st Hawaii International Conference on System Sciences (2018)

17. Bassanelli, S., Bucchiarone, A.: GamiDOC: a tool for designing and evaluating gamified solutions. In: Extended Abstracts of the 2022 Annual Symposium on Computer-Human Interaction in Play, pp. 203–208 (2022)
18. Tondello, G.F., Wehbe, R.R., Diamond, L., Busch, M., Marczewski, A., Nacke, L.E.: The gamification user types hexad scale. In: Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play, pp. 229–243 (2016)
19. Oliveira, W., Hamari, J., Shi, L., Toda, A.M., Rodrigues, L., Palomino, P.T., Isotani, S.: Tailored gamification in education: a literature review and future agenda. Educ. Inf. Technol., pp. 1–34 (2022)
20. Codish, D., Ravid, G.: Gender moderation in gamification: does one size fit all? (2017)
21. Klock, A.C.T., Gasparini, I., Pimenta, M.S.: 5w2h framework: a guide to design, develop and evaluate the user-centered gamification. In: Proceedings of the 15th Brazilian Symposium on Human Factors in Computing Systems, pp. 1–10 (2016)
22. Morschheuser, B., Maedche, A., Walter, D.: Designing cooperative gamification: conceptualization and prototypical implementation. In: Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, pp. 2410–2421 (2017)
23. Deterding, S.: The lens of intrinsic skill atoms: a method for gameful design. Hum. Comput. Interact. **30**(3–4), 294–335 (2015)
24. Kim, B.: Designing gamification in the right way. Lib. Technol. Rep. **51**(2), 29–35 (2015)
25. Moizer, J., Lean, J., Dell'Aquila, E., Walsh, P., (Alfie) Keary, A., O'Byrne, D., Di Ferdinando, A., Miglino, O., Friedrich, R., Asperges, R., Sica, L.S.: An approach to evaluating the user experience of serious games. Comput. Educ. **136**, 141–151 (2019)
26. Atrash, A., Mower, E., Shams, K., Mataric, M.J.: Recognition of physiological data for a motivational agent. In: Computational Physiology, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-04, Stanford, CA, March 21–23, 2011. AAAI, Washington (2011)
27. Di Ruscio, D., Kolovos, D.S., de Lara, J., Pierantonio, A., Tisi, M., Wimmer, M.: Low-code development and model-driven engineering: two sides of the same coin? Softw. Syst. Model. **21**(2), 437–446 (2022)
28. Mizutani, W.K., Daros, V.K., Kon, F.: Software architecture for digital game mechanics: a systematic literature review. Entertain. Comput. **38**, 100421 (2021)
29. Qu, J., Song, Y., Wei, Y.: Design patterns applied for game design patterns. In: Chen, Y. (Ed.) 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2016, Shanghai, May 30–June 1, 2016, pp. 351–356. IEEE Computer Society, Washington (2016)
30. Rapp, A., Hopfgartner, F., Hamari, J., Linehan, C., Cena, F.: Strengthening gamification studies: current trends and future opportunities of gamification research (2019)
31. Khaled, R., Fischer, R., Noble, J., Biddle, R.: A qualitative study of culture and persuasion in a smoking cessation game. In: International Conference on Persuasive Technology, pp. 224–236. Springer, Berlin (2008)
32. Orji, R., Mandryk, R.L., Vassileva, J.: Improving the efficacy of games for change using personalization models. ACM Trans. Comput. Hum. Interact. **24**(5), 32 (2017)
33. Lavoué, E., Monterrat, B., Desmarais, M., George, S.: Adaptive gamification for learning environments. IEEE Trans. Learn. Technol. **12**(1), 16–28 (2018)
34. Pastushenko, O., Oliveira, W., Isotani, S., Hruška, T.: A methodology for multimodal learning analytics and flow experience identification within gamified assignments. In: Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–9 (2020)
35. Deci, E.L., Koestner, R., Ryan, R.M.: A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. Psychol. Bull. **125**(6), 627 (1999)
36. Malone, T.W.: What makes things fun to learn? Heuristics for designing instructional computer games. In: Proceedings of the 3rd ACM SIGSMALL Symposium and the First SIGPC Symposium on Small Systems, pp. 162–169. ACM, New York (1980)
37. Tondello, G.F., Wehbe, R.R., Diamond, L., Busch, M., Marczewski, A., Nacke, L.E., The gamification user types hexad scale. In: Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play, CHI PLAY '16, pp. 229–243. ACM, New York (2016)

38. Hanus, M.D., Fox, J.: Assessing the effects of gamification in the classroom: a longitudinal study on intrinsic motivation, social comparison, satisfaction, effort, and academic performance. Comput. Educ. **80**, 152–161 (2015)
39. TechnologyAdvice.com. Compare 120+ gamification platforms (2019). https://technologyadvice.com/gamification/
40. Bucchiarone, A., Martorella, T., Colombo, D., Cicchetti, A., Marconi, A.: POLYGLOT for gamified education: mixing modelling and programming exercises. In: ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS 2021 Companion, Fukuoka, October 10–15, 2021, pp. 605–609. IEEE, Piscataway (2021)
41. Vasileva, A., Schmedding, D.: How to improve code quality by measurement and refactoring. In: Paulk, M.C., Machado, R.J., Brito, M.A., Goulão, M., Amaral, V. (Eds.) 10th International Conference on the Quality of Information and Communications Technology, QUATIC 2016, Lisbon, September 6–9, 2016, pp. 131–136. IEEE Computer Society, Washington (2016)
42. de Paula Porto, D., de Jesus, G.M., Ferrari, F.C., Fabbri, S.C.P.F.: Initiatives and challenges of using gamification in software engineering: a systematic mapping. J. Syst. Softw. **173**, 110870 (2021)