Information and Software Technology 95 (2018) 201–218

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A systematic mapping study on game-related methods for software engineering education

Mauricio R. de A. Souza[a,*], Lucas Veado[a], Renata Teles Moreira[b], Eduardo Figueiredo[a], Heitor Costa[b]

[a] Computer Science Department, Federal University of Minas Gerais, Belo Horizonte, Minas Gerais, Brazil
[b] Computer Science Department, Federal University of Lavras, Lavras, Minas Gerais, Brazil

ARTICLE INFO

Keywords:
Software engineering education
Game-based learning
Gamification
Game development based learning

ABSTRACT

Context: The use of games in software engineering education is not new. However, recent technologies have provided new opportunities for using games and their elements to enhance learning and student engagement.
Objective: The goal of this paper is twofold. First, we discuss how game-related methods have been used in the context of software engineering education by means of a systematic mapping study. Second, we investigate how these game-related methods support specific knowledge areas from software engineering. By achieving these goals, we aim not only to characterize the state of the art on the use of game-related methods on software engineering education, but also to identify gaps and opportunities for further research.
Method: We carried out a systematic mapping study to identify primary studies which address the use, proposal or evaluation of games and their elements on software engineering education. We classified primary studies based on type of approaches, learning goals based on software engineering knowledge areas, and specific characteristics of each type of approach.
Results: We identified 156 primary studies, published between 1974 and June 2016. Most primary studies describe the use of serious games (86) and game development (57) for software engineering education, while Gamification is the least explored method (10). Learning goals of these studies and their development of skills are mostly related to the knowledge areas of "Software Process", "Software Design", and "Professional Practices".
Conclusions: The use of games in software engineering education is not new. However, there are some knowledge areas where the use of games can still be further explored. Gamification is a new trend and existing research in the field is quite preliminary. We also noted a lack of standardization both in the definition of learning goals and in the classification of game-related methods.

1. Introduction

The challenges of educating new software engineers is more than just programming, they include attention to details, such as quality, schedule, and economic goals [1]. For instance, an important challenge in software engineering education arises from the dual nature of the software engineering discipline: it has roots in computer science and has emerged as an engineering discipline, and it affects both theory and practice [1]. This characteristic has a direct impact on the amount of material instructors must cover in software engineering classrooms. In addition, software professionals are required not only to understand technical challenges but also to be up-to-date with nontechnical issues, including management, communication, and teamwork.

As a result, software engineering education should engage students to experience the professional practices of software engineering in such a way that they can understand which practices and techniques are useful in various different situations [1]. The nature of the assignments and projects proposed in the classroom are limited in scope and time, increasing the challenge of finding a good balance between theory and practice. Researchers and educators have been dedicating efforts to develop and apply new teaching methods to support the extensive software engineering curriculum [2–4].

ACM and IEEE jointly give directions for software engineering education by the "Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering" (SE 2014, for short) [1]. This document defines ten knowledge areas, such as Software Design, Software Process, and Software Quality, which represent particular disciplines of software engineering that students should know after they

* Corresponding author.
E-mail addresses: mrasouza@dcc.ufmg.br, mauricio.ronny@gmail.com (M.R. . Souza), furtini@dcc.ufmg.br (L. Veado), renata@dcc.ufla.br (R.T. Moreira), figueiredo@dcc.ufmg.br (E. Figueiredo), heitor@dcc.ufla.br (H. Costa).

http://dx.doi.org/10.1016/j.infsof.2017.09.014
Received 15 April 2017; Received in revised form 22 September 2017; Accepted 24 September 2017
Available online 07 October 2017

Table 1
Knowledge areas from SE 2014 [1] used in this study.

Acronym	Name	Description
MAA	Software Modeling and Analysis	Modeling and analysis can be considered core concepts in any engineering discipline because they are essential to documenting and evaluating design decisions and alternatives.
REQ	Requirements Analysis and Specification	The construction of requirements includes elicitation and analysis of stakeholders' needs and the creation of an appropriate description of desired system behavior and qualities, along with relevant constraints and assumptions.
DES	Software Design	Software design is concerned with issues, techniques, strategies, representations, and patterns used to determine how to implement a component or a system.
VAV	Software Verification and Validation	Software verification and validation uses a variety of techniques to ensure that a software component or system satisfies its requirements and meets stakeholder expectations.
PRO	Software Process	Software process is concerned with providing appropriate and effective structures for the software engineering practices used to develop and maintain software components and systems at the individual, team, and organizational levels.
QUA	Software Quality	Software quality is a crosscutting concern, identified as a separate entity to recognize its importance and provide a context for achieving and ensuring quality in all aspects of software engineering practice and process.
PRF	Professional Practice	Professional practice is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering professionally, responsibly, and ethically.

graduate. However, as far as we are concerned, no study has yet investigated which and how these knowledge areas are supported by game-related educational methods, such as serious games, Gamification, and Game Development Based Learning.

In this context, the goal of this paper is to identify and discuss game-related methods in the context of software engineering education. We call game-related methods for software engineering education, any approach where games are used to support the process of teaching or learning software engineering. Therefore, we conducted a systematic mapping study to survey the software engineering education literature in order to: (i) identify game-related methods for software engineering education; (ii) understand how these game-related methods support the software engineering knowledge areas from SE 2014; and (iii) understand specific characteristics of each game-related method.

In our previous work [5], we identified three game-related methods for software engineering education: (i) the use of games (or serious games) as learning tools (Game-based Learning); (ii) the development of games as a context for learning (Game Development Based Learning); and (iii) the use of game elements and game design techniques to improve the learning experience (Gamification). This paper expands the results of our previous study [5] in several ways. First, we mined two additional scientific databases (EI Compendex and Scopus), increasing the set to a total of 4 databases mined for primary studies (ACM Digital Library, IEEE Xplore, EI Compendex, and Scopus). Second, as a result of more databases, we increase the number of primary studies from 106 to 156; i.e., 50 different papers found and analyzed. Third, we proposed and investigated three additional research questions, resulting now in five, to provide further understanding about the role of game-related educational methods in software engineering education.

Previous studies [3,4,6,7] investigated and mapped educational methods in the software engineering education context. This study complements previous ones by providing the following novel contributions to the community. First, we propose a classification scheme for game-related educational methods used in software engineering education. Second, we analyze the difference in the proposal and application of methods found in the literature. Third, we also discuss how these educational methods support specific software engineering education knowledge areas.

From the 156 primary studies analyzed, we identified three game-related methods used in software engineering education: "Game-based Learning", "Game Development Based Learning", and "Gamification". We mapped their learning goals and identified that these educational methods are mostly used to support the knowledge areas of "Software Process", "Software Design" and "Professional Practice" (as described in [1]). However, each educational method has different purposes and rationales to support these knowledge areas, thus serious games are more used in the context of "Software Process" education, while "Game Development Based Learning" is more used in "Software Design" education. Gamification in software engineering education is new, and initial results are promising, however, further investigation and additional empirical evidences are still needed for the discussion of its effectiveness in this context.

The remainder of this paper is organized as follows. Section 2 provides the necessary theoretical foundation for the study. In Section 3, we describe the design of this systematic mapping study. Section 4 presents the results of the study and Section 5 discusses the results and provides insights about the role of each game related method in software engineering education. In Section 6, we discuss the threats to the validity of the study while we discuss the related work in Section 7. Section 8 concludes this research paper.

2. Background

This section defines important concepts used in this study. In Section 2.1, we discuss the scope of software engineering education, in terms of knowledge areas. Section 2.2 presents the definition of game-related approaches used in this study.

2.1. Software engineering education knowledge areas

The "Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering" (or "SE 2014") [1] provides guidance to academic institutions and accreditation agencies about what should constitute an undergraduate course on software engineering. This document defines a body of knowledge suggesting what every software engineering graduate must know: The Software Engineering Education Knowledge. SE 2014 is organized in ten "knowledge areas" [1], representing particular subdisciplines of software engineering that are generally recognized as a significant part of the software engineering knowledge that a graduate should know. In this paper, we used these areas as reference for identifying what software engineering disciplines are supposed to be supported by game-related approaches.

Table 1 provides the acronym, name and a brief description of the seven knowledge areas considered in this study, namely Software Modeling and Analysis (MAA), Requirements Analysis and Specification (REQ), Software Design (DES), Software Verification and Validation (VAV), Software Process (PRO), Software Quality (QUA), and Professional Practice (PRF). For instance, Software Process is concerned with software engineering practices used to develop and maintain software components and systems at the individual, team, and organizational levels.

Although we acknowledge the relevance of the knowledge areas "Computing Essentials", "Mathematical and Engineering Fundamentals" and "Security", we did not consider them in this study

because they are not specific to the context of software engineering education. These knowledge areas cover basic/introductory topics for computing/IT and engineering courses that may be studied in more details in other domains of computer science or information systems degrees. For example, Computing Essentials cover a range of topics from algorithms, data structures, and complexity (commonly associated to CS1/CS2 courses) to computer organization and operational system basics.

2.2. Game-related methods

In context of this study, the term "game-related methods" refers to any approach that uses games or game elements for supporting teaching and learning processes in the scope of software engineering education. Therefore, this study is not limited to (serious) games. It includes the use of game elements and mechanisms, and adoption of the domain of game development as learning instruments. We used the classification scheme of our previous work [5] for classifying game-related approaches for software engineering education: (i) Game Based Learning (GBL); (ii) Game Development Based Learning (GDBL); and (iii) Gamification.

The term "Game Based Learning" (GBL) has been used to refer to any approach using games for learning purposes. We adopted the definition of GBL used by Wangenheim and Shull [2]: the use of game applications for defining learning outcomes. Games are any contest (play) among adversaries (players) operating under constraints (rules) for an objective (winning, victory, or pay-off) [2]. Hence, GBL approaches apply games with the purpose of learning specific skills and concepts, usually named as "serious games" (games with purposes), edutainment, or educational games. In this study, we did not limit this concept to digital games or to the use of games designed specifically for learning purposes (serious games), we considered any game used in educational context.

Problems and Programmers [8] and SimSE [9] are examples of GBL approaches for software engineering education. The first is a nondigital card game that simulates software development process from conception to completion. The players compete to finish their projects while avoiding the potential pitfalls of software engineering. SimSE is a computer simulation based game whose goal is to allow students to practice a "virtual" software engineering process (or sub-process) in a fully graphical, interactive, and fun setting. It provides direct, graphical feedback that enables students to learn the complex cause and effect relationships underlying the processes of software engineering [9].

Given the practical nature of software engineering and the need of hands-on experience in the process of software development education, educators have also used the domain of digital games development for educational purposes [5]. Hence, Wu and Wang [10] define Game Development Based Learning (GDBL) as an approach where students are required to modify or develop a game as a part of a course using a game development framework (GDF) to learn skills within computer science and software engineering. In the context of software engineering education, game development brings the fun factor to courses and provides pedagogical aspects of problem-based learning, cooperative learning, blended learning, and experiential learning [11]. When developing a game, students have hands-on experience on software process, design, and other skills related to software engineering.

Gamification is a relatively new term that has been used to denote the use of game elements and game-design techniques in non-gaming contexts [12]. The goal of Gamification is to use the philosophy, elements, and mechanisms of game design in non-game environments to induce certain behavior in people, as well as to improve their motivation and engagement in a particular task [13]. The key difference between Gamification and the use of games for learning is that Gamification deals with creating a game like experience, by incorporating elements of games, in real life contexts or applications, while the latter is the use of full-fledge games for educational purposes.

For instance, Steam[1] is a digital distribution platform developed by Valve Corporation where users can buy and manage software licenses (mainly digital games). The Steam platform has incorporated several game elements, such as, users collect badges for performing activities in the platform (reviewing games, updating profiles, and posting comments), users earn experience points and levels, and users are awarded with collectibles (trading cards) for buying and playing games. Although the main use of the platform is for purchasing and managing software licenses (non-game context), the platform adopts these game elements to engage and motivate users in using all its features (consuming more products) and to promote competition among users.

In the context of software engineering, researchers and practitioners have experimented the adoption of Gamification in professional activities, such as, Gamification of the software development life cycle [14,15], software process improvement initiatives [16], and also in software engineering education [17–19]. Game elements and mechanisms may engage students in performing specific tasks or incorporating specific behaviors in the teaching-learning process.

3. Study design

This section presents the goals of this study and the methodology adopted. Section 3.1 presents the study goal and research questions. Section 3.2 explains the research method and the process executed. Section 3.3 discusses the search strategy applied to mine relevant scientific databases. Section 3.4 shows the selection process filtering only relevant papers for this study. Section 3.5 presents the classification scheme and mapping strategy used to present the study results.

3.1. Goal and research questions

The goal of this study is to *identify and analyze game-related methods from the purpose of understanding their application and learning objectives in the context of software engineering education*. To achieve this goal, we established five research questions:

RQ1. What game related methods have been proposed for supporting software engineering education?
RQ2. What software engineering education knowledge areas are supported by the existing game-related methods?
RQ3. What types of games have been proposed or used to support software engineering education?
RQ4. What technologies have been used to support GDBL approaches?
RQ5. What game elements have been used for the Gamification of software engineering education?

Based on the classification scheme for game-related methods for software engineering education [5], the expected outcome of RQ1 is a list of primary studies categorized by approach type. The expected outcome of RQ2 is a mapping between learning objectives from the approaches identified in RQ1 and the software engineering knowledge areas from SE 2014. Based on the list of GBL approaches identified in RQ1, the expected outcome of RQ3 is an overview of the characteristics of games for software engineering education. Similarly, based on the list of GDBL approaches identified in RQ1, the expected outcome of RQ4 is a summary of frameworks, environments, and programming languages used to support the development of games in the context of software engineering education. Finally, the expected outcome of RQ5 is a summary of game elements, mechanics, and dynamics extracted from the list of Gamification approaches identified in RQ1.

[1] http://store.steampowered.com/.

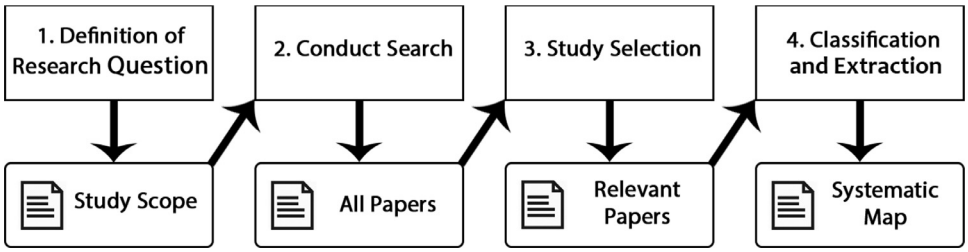

Fig. 1. Systematic mapping process, adapted from Petersen et al. [20].

3.2. Method

To achieve the study goal (Section 3.1), we have conducted a Systematic Mapping Study (SMS). SMS is a secondary study method that systematically (i.e., based on a structured and repeatable process or protocol) explores and categorizes studies in a given research field, and provides a structure of the type of research reports and results that have been published [20]. SMS results are often organized as visual summaries (maps). The expected contribution is to provide researchers and educators with an overview of the state-of-the-art on the use of games and game elements for supporting software engineering knowledge areas from the educational perspective. Additionally, we expect to identify possible research gaps and trends.

We have conducted the SMS in the period of June/2016 to December/2016, following four process steps adapted from Petersen et al. [20] (Fig. 1).

- Step 1 – Definition of research questions: We have defined five research questions based on the study goal, to establish the desired scope of the systematic study (Section 3.1);
- Step 2 – Conduct search: based on the research questions, we defined and performed a replicable process for searching for retrieving papers in selected scientific databases (Section 3.3);
- Step 3 – Study Selection: We have defined and applied a replicable process for selecting only the relevant papers for this study (Section 3.4);
- Step 4 – Study Classification and Data Extraction: Based on the research questions, we have defined a strategy for mapping the relevant data from the primary studies (Section 3.5) and for presenting the study results (Section 4).

Five researchers participated in the planning and execution of the study: an undergraduate student in CS, a PhD student in CS, and three PhD associate professors teaching software engineering courses in two different institutions.

3.3. Search strategy

The search strategy enables the inclusion of relevant studies in the search results. To identify possible primary studies for data extraction, the search was based on (i) trial searches using combinations of keywords derived from the study goal for the definition of valid search strings and (ii) the execution of automatic searches in the scientific databases using the search strings. Initially, we selected relevant keywords related to the three domains under analysis: (a) education; (b) software engineering; and (c) game-related approaches. In order to identify these relevant keywords, we considered search strings used in related systematic studies, bodies of knowledge and curricula guidelines for software engineering, and experts (software engineering educators). The resulting keywords per domain were:

- Education: teach, learn, education, train;
- Software engineering: software engineering, software process, software requirements, requirements engineering, software verification, software validation, software design, software architecture, software test, software quality, software project management;
- Game-related approaches: game, serious games, edutainment, Gamification, game based learning, gbl.

Search strings were defined by grouping keywords in the same domain with the logic operator “OR” and grouping the three domains with the logic operator “AND”. We then executed automatic searches in four selected scientific databases, using and adapting (when necessary) the search string. The databases used were ACM Digital Library,[2] IEEE Xplore,[3] EI Compendex,[4] and Scopus,[5] since they have a large amount of relevant conferences and journals indexed. We limited the results of automatic searches to return only papers written in English, and we did not apply any time limitation.

3.4. Study selection

In this step, we filtered the studies retrieved from the automatic searches by some steps to exclude papers not aligned with the study goals. First, the five researchers defined the following inclusion and exclusion criteria:

- Inclusion Criteria: Studies whose main focus was on proposal, usage, discussion or evaluation of game-related methods in the context of software engineering education;
- Exclusion Criteria: Papers not written in English; Papers not fully available to download; Studies formatted as short papers (2 or less pages), tutorials, demos, books, book chapters, and similar; secondary studies; and duplicated studies.

The study selection process was executed in two phases: (i) in the first selection phase, two researchers individually read all titles and abstracts and removed studies that did not comply with inclusion criteria; (ii) in the second selection phase, they have downloaded all remaining papers and each researcher examined all papers’ introduction and conclusion to remove studies that matched the exclusion criteria. During these steps, any conflict was solved with a third researcher. For any persisting problem, the two remaining researchers were consulted for a final decision regarding the inclusion or exclusion of a primary study. However, only a few conflicts emerged in the first selection phase (less than 7% of the selected studies), and most cases because the learning topic was not clearly related to software engineering. In the second selection phase, those conflicted studies were carefully read and the third researcher helped in the decision of inclusion or exclusion. Therefore, we believe the selection process is clear and precise enough for replication.

Table 2 summarizes the result of the selection process. The first column (*Source*) presents the scientific databases mined and the second one (*# Papers*) describes the number of papers retrieved from the use of the search strings. The third (*1st Selection*) and fourth (*2nd Selection*) columns shows the number of remaining papers after the execution of

[2] http://dl.acm.org/.
[3] http://ieeexplore.ieee.org/.
[4] http://www.engineeringvillage.com/.
[5] https://www.scopus.com/.

Table 2
Study selection process.

Source	# Papers	1st Selection	2nd Selection	% Included	Interval
ACM DL	1204	248	42	3,49%	1974–2016
IEEE Xplore	558	155	69	12,37%	1977–2015
EI Compendex	1133	313	120	10,68%	1982–2016
Scopus	1270	336	132	10,47%	1974–2016
Total	4165	1052	363	8,72%	1974–2016
Total[a]	2675	568	156	5,83%	1974–2016

[a] Total without duplicates.

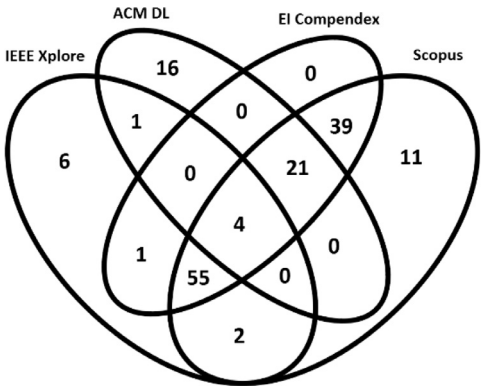

Fig. 2. Distribution of primary studies per Database.

the first and second selection phases, respectively. The fifth column (*% Included*) describes the ratio between the remaining papers after the selection process (*2nd Selection*) and the total number of papers retrieved from each database (*# Papers*). Finally, the last column (*Interval*) presents the time range of the resulting primary studies.

The study selection process resulted in 156 unique primary studies. However, we found a high number of duplicates among different scientific databases. Fig. 2 illustrates the overlapping results between databases. We removed the duplicated papers from the results of the automatic search (# Papers). However, for the purpose of documentation and analysis, we kept track of the database from where each paper was found.

3.5. Study classification and data extraction

We analyzed the resulting list of primary studies for extracting information by identifying: (i) the presented approach; (ii) the game-related method used; and (iii) learning objectives or expected developed skills. We classified each approach using game-related methods (RQ1) in the following categories [5]: Gamification, GBL, and GDBL (as described in Section 2.2). We then identified the expected learning outcomes and mapped them to the purpose and related topics of the knowledge areas from SE 2014.

4. Results

In this section, we presented the results of the systematic mapping study. Section 4.1 provides an overview of the primary studies selected for this study. Sections 4.2–4.6 describes the results for the research questions RQ1–RQ5, respectively.

4.1. Overview

The study selection resulted in 156 primary studies, published between 1974 and June 2016. Fig. 3 presents a histogram with the frequency of primary studies per year, with different colors representing the types of approaches found (discussed in details in Section 4.2). This result suggests that software engineering education has been a challenge since the early years of software engineering in the 1970s, and that the use of game-related educational methods is not a novelty. Nonetheless, since 2002 there has been a constant and growing presence of studies about the use of game-related methods for software engineering education.

Fig. 4 shows the distribution of studies per type of forum. The majority of the primary studies was published in conferences (69%), but we also retrieved studies from journals (24%) and workshops (7%). The conferences with greater occurrences of primary studies were CSEE & T, FIE, SIGCSE, and ICSE. Table 3 summarizes the most recurring publication venues in our study and their respective counting of selected primary studies. We list the publication venues that have three or more primary studies selected in this study. Appendix A provides the complete list of primary studies. In the remainder of this paper, we used unique identifications (PS < number >) when referring to primary studies.

4.2. RQ1 – game-related methods for software engineering education

This section discusses the results for the first research question.

> *RQ1. What game-related methods have been proposed to support software engineering education?*

In accordance to the classification described in Section 3.5, 88 primary studies presented the use of GBL method, 60 presented the use of GDBL method, and 11 presented Gamification approaches. Three primary studies presented hybrid approaches, i.e., the combination of two distinct methods: (i) one used Gamification and GDBL; and (ii) two used GDBL and GBL. Table 4 presents the mapping of game-related methods found (Method), the primary studies that use each method (Primary Studies) and their quantity, and the number of studies that have some sorts of evaluation in the classroom.

GBL and GDBL are the most prevalent methods found in this study. Both methods are used to support software engineering education since the decade of 1970s (PS082, PS143). GBL approaches include serious games and non-serious games in digital or non-digital formats. Some examples of serious games include GetKanban (PS087), PMG-2D (PS071), and InspectorX (PS055), designed to support the learning of Kanban, project management and software inspection, respectively. The use of games as learning tools (GBL approaches) is usually motivated by the opportunity of teaching learning topics difficult to explore in traditional lectures. This difficulty is often caused by limitations imposed by the nature of the topic, by time restrictions of traditional lectures or by practical problems related to real life experiences hard to translate to examples in the classroom. For instance, Smart Decisions (PS080) is a game developed to aid in teaching architecture design. Cervantes et al. [21] suggest “software architecture courses frequently rely on relatively simple examples (…) Such examples may present the essential concepts of design, but they are also distant from real life system design experiences”. The authors also suggest that the timeframe is too short to observe consequences of the decisions that are

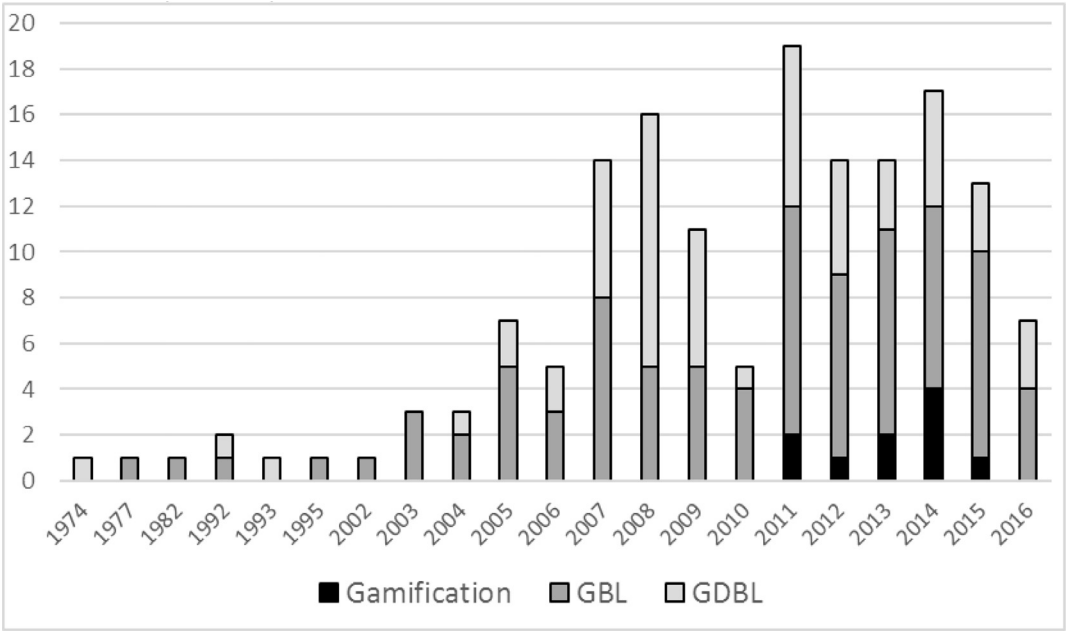

Fig. 3. Timeline of primary studies.

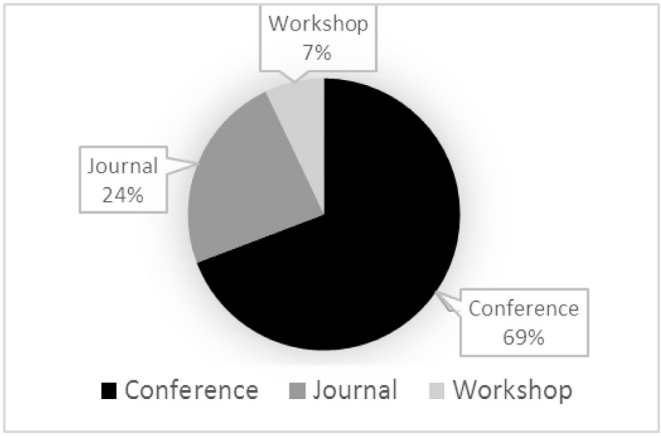

Fig. 4. Distribution of primary studies per type of forum.

Table 3
Relevant publication venues for game related approaches in software engineering education.

# Studies	Publication venues
15	IEEE Conference on Software Engineering Education and Training (CSEE & T)
13	Frontiers In Education Conference (FIE)
11	ACM Technical Symposium on Computer Science Education (SIGCSE)
10	International Conference on Software Engineering (ICSE)
5	Journal of Computing Sciences in Colleges
4	IEEE Transactions on Education
4	IEEE Global Engineering Education Conference (EDUCON)
3	International Conference on Computer Games (CGAMES)
3	International Workshop on Games and Software Engineering (GAS)
3	International Journal of Computer Games Technology
3	ASEE Annual Conference and Exposition

made as part of the design process [21]. Therefore, games are useful to support the setup of a controlled environment that can simulate real life experiences.

Primary studies proposing the use of GDBL method are divided in two types: (i) assignments on developing game products; and (ii) adoption of game development frameworks. The former studies describe experiences of students' assignment with hands-on experience in the domain of game development to introduce the challenges of software engineering. For instance, PS111 describes and evaluates a block course in which students with almost no mobile application development experience create games in just two weeks. These students learn modeling, design patterns, software configuration management, and game development. PS123 employs game "play testing" for learning about issues and challenges of modern software engineering projects and practices. The later studies advocate the use of game development frameworks as learning instruments to simplify certain aspects of development (e.g., technicalities of learning a new language) and allowing students to concentrate efforts on specific learning goals (e.g. software design). For instance, PS150 describes a case study of how a game project using the XNA Game Studio from Microsoft was implemented in a software architecture course.

Table 4
Mapping of categories of game-related methods and primary studies.

Method	Primary studies	# Primary studies	Evaluation
Gamification	PS001–PS010	10 primary studies – 6.4%	6 (60%)
GBL	PS011–PS096	86 primary studies – 55.1%	45 (52.3%)
GDBL	PS097–PS153	57 primary studies – 36.5%	18 (31.6%)
Gamification and GDBL	PS154	1 primary study – 0.6%	0
GBL and GDBL	PS155–PS156	2 primary studies – 1.2%	1 (50%)

Concerning Gamification, although the term is recent in literature [12], it was rapidly incorporated in software engineering education since 2011 (PS007, PS010). Gamification approaches found are usually applied as an innovative learning context more focused in engaging and motivating learners to perform desired behaviors. However, we observed two different strategies for the use of Gamification in software engineering education: (i) Gamification of the classroom experience and (ii) Gamification of specific software engineering activities. Gamification of the classroom experience refers to the use of game elements to engage and motivate students in performing learning activities

(PS003, PS004, PS005, PS008). For instance, PS005 describes the experiment of implementing Gamification techniques into software engineering and service-oriented architecture courses using Points, Leaderboards and Badges to promote competition and, consequently, to motivate students in the class activities. The strategy is focused in the Gamification of the classroom activities, and not specific software engineering related activities.

Gamification of specific software engineering activities, on the other hand, applies game elements to motivate learners in practicing specific skills or performing specific practices (PS001, PS002, PS006, PS007, PS010). PS006 describes an experiment in which students are encouraged to make more frequent commits to version control system, in a software project course. The authors proposed a leaderboard based on the number of commits from each student, and established milestones/thresholds that would trigger messages congratulating students and teams that reached the specified number of commits. PS01 uses weekly challenges to motivate students on applying eXtreme Programming practices to their project. In this scenario, students compete for a "challenge cup" award.

Some studies (PS154, PS155, and PS156) mix the previous approaches in order to achieve better learning results (hybrid approaches). Two studies (PS155 and PS156) discuss the advantages of learning cycles where students play and develop serious games. One study (PS154) proposes a hybrid approach consisting of developing games focusing on good design, and applying a gamified strategy for testing it.

Regarding the evaluation of those studies, most of them adopt case studies or pilot studies in classroom. However, only 70 primary studies formally describe proper evaluation methods and results, regarding effectiveness in learning or perception of users: 45 (out of 86–52,3%) studies on GBL method; 18 (out of 57–31,6%) studies on GDBL method; 6 (out of 10–60%) on Gamification; and 1 (out of 3–33,3%) on hybrid approaches. The most recurring evaluation strategies rely on pre-test and post-test, and on questionnaires. The higher number of GBL studies with proper evaluation, in comparison to GDBL, may relate to the nature of GBL approaches: The evaluation of serious games can be performed in shorter periods, while the evaluation of development project requires longer periods. Consequently, this factor may impact on availability of participants and in the effort to perform comparative studies with control groups. In the case of Gamification, the total number of primary studies is low, therefore further investigation and additional empirical evidences are still needed for the discussion of its effectiveness in software engineering education.

4.3. RQ2 - mapping learning goals to SE 2014 knowledge areas

This section discusses the results for the following research question.

> *RQ2. What software engineering education knowledge areas are supported by the existing game-related approaches?*

We identified the learning goals of each primary study and mapped them to the SE 2014 knowledge areas (KA) described in Section 2.1 (Table 5). The results show that "Software Process" is the KA with the greater number of primary studies with game related approaches supporting it (92–59%), followed by "Software Design" (60–38.5%), "Professional Practice" (48–30.8%), "Requirement Analysis and Specification" (39–24.3%), and "Software Verification and Validation" (30–19.2%). The least supported KA are "Software Quality" (10–6.4%) and "Software Modeling and Analysis" (10–6.4%).

The KA "Software Design" (DES), "Requirement Analysis and Specification" (REQ) and "Software Verification and Validation" (VAV) are directly linked to recurrent phases in software development lifecycle models (e.g., the phases of "Requirements", "Design" and "Verification" in representations of the waterfall model). Consequently, the primary studies proposed learning experiences that address each of these KA specifically or setups where students experience topics related to these KA in the context of a software development process. Most of the primary studies mapped to the KA "Professional Practice" proposes game related approaches that directly support other KA and, as secondary goal, they encourage the development of professional or "soft" skills. These skills are usually related to dynamics of working in teams or groups, interacting with stakeholders, communication, and leadership.

We found few primary studies indicating the support for KA "Software Quality" (QUA) and "Software Modeling and Analysis" (MAA). Usually, the supporting for MAA was related to applying UML (Unified Modeling Language) in software design for documentation and analysis purposes. The supporting for QUA was related to definition and appliance of quality assurance procedures in the contexts of software process and software verification. The description of these knowledge areas [1] reflects their "supportive" nature: (i) MAA is described as "essential to documenting and evaluating design decisions and alternatives"; and (ii) QUA is described as a "crosscutting concern" and is suggested that "software quality topics must be integrated into the presentation and application of material associated with other KA".

Regarding the support of knowledge areas by each game-related method, Fig. 5 describes a quantitative relationship between game-related methods and the coverage of each SE 2014 knowledge area. The different nature of each game-related method results in different strategies to support learning goals. The GBL approaches usually focus on few learning topics: 65 out of 88 (73.9%) primary studies describing GBL approaches support only one knowledge area. In contrast, 23 out of 60 (38.3%) primary studies describing GDBL approaches support a single knowledge area. This analysis shows that GBL approaches are, usually more specific in terms of learning goals, while the practical nature of GDBL approaches are broader.

The mapping of Fig. 5 also shows that "Software Design" (DES) is predominant in GDBL approaches, while "Software Process" (PRO) is predominant in GBL approaches. We observed that GDBL method favors hands-on experience on software design in a domain of systems familiar to students. The use of game development frameworks helps overcoming technology barriers (e.g. complexity of learning new programming languages) in software development, allowing students to focus on learning good design. This practical nature of GDBL approaches also reflects the higher number of primary studies supporting "Software Quality" (QUA) and "Software Modeling and Analysis" (MAA), providing learners with the experience of learning by doing.

The challenges and nuances of software process are difficult to simulate in student projects because of diverse factors (e.g., limitation of time, limitation of students' availability). Therefore, games and simulations play a great role in creating abstractions and simplifications of real life software development.

4.4. RQ3 - games for software engineering education

This section discusses the results for the following research question.

> *RQ3. What types of games have been proposed or used to support software engineering education?*

We analyzed the GBL approaches identified in RQ1, to understand what types of games have been proposed. Out of 88 GBL approaches, 54 primary studies (61.4%) described the use of digital games, 32 primary studies (36.4%) described the use of non-digital games, and 2 primary studies (2.2%) described the use of games with digital and non-digital components.

Table 5
Mapping of knowledge areas (KA) from SE 2014 and primary studies.

K.A.	Primary studies	#
PRO	PS001; PS002; PS006; PS007; PS011; PS012; PS014; PS015; PS016; PS017; PS019; PS020; PS021; PS022; PS023; PS025; PS026; PS027; PS028; PS029; PS030; PS034; PS035; PS036; PS037; PS039; PS040; PS042; PS043; PS044; PS045; PS046; PS049; PS050; PS052; PS054; PS057; PS058; PS059; PS060; PS061; PS062; PS063; PS064; PS068; PS069; PS070; PS071; PS074; PS075; PS076; PS077; PS079; PS081; PS084; PS086; PS087; PS088; PS089; PS091; PS092; PS094; PS096; PS097; PS102; PS107; PS110; PS111; PS115; PS116; PS118; PS119; PS120; PS121; PS122; PS124; PS128; PS129; PS130; PS131; PS134; PS135; PS140; PS141; PS142; PS144; PS145; PS146; PS148; PS151; PS155; PS156;	92 (59%)
DES	PS007; PS015; PS020; PS026; PS048; PS056; PS065; PS073; PS080; PS081; PS089; PS095; PS097; PS098; PS099; PS100; PS101; PS103; PS105; PS107; PS108; PS110; PS111; PS112; PS113; PS114; PS116; PS117; PS119; PS120; PS121; PS122; PS123; PS124; PS125; PS126; PS127; PS129; PS130; PS132; PS133; PS134; PS135; PS136; PS137; PS138; PS139; PS140; PS142; PS143; PS144; PS145; PS146; PS148; PS149; PS150; PS152; PS153; PS154; PS155;	60 (38.5%)
PRF	PS007; PS012; PS013; PS014; PS015; PS017; PS018; PS020; PS025; PS026; PS030; PS031; PS043; PS061; PS064; PS072; PS081; PS085; PS086; PS090; PS091; PS097; PS099; PS102; PS106; PS107; PS110; PS111; PS119; PS121; PS124; PS128; PS129; PS130; PS134; PS135; PS139; PS141; PS143; PS144; PS145; PS146; PS147; PS148; PS151; PS152; PS155; PS156;	48 (30.8%)
REQ	PS007; PS013; PS014; PS015; PS017; PS020; PS026; PS032; PS043; PS047; PS048; PS051; PS053; PS078; PS090; PS093; PS097; PS101; PS107; PS110; PS114; PS116; PS117; PS119; PS120; PS121; PS122; PS123; PS124; PS128; PS129; PS130; PS134; PS140; PS142; PS145; PS146; PS148; PS155;	39 (24.3%)
VAV	PS007; PS010; PS015; PS024; PS033; PS038; PS048; PS055; PS066; PS067; PS083; PS107; PS109; PS114; PS116; PS119; PS120; PS121; PS122; PS123; PS129; PS134; PS142; PS144; PS145; PS146; PS148; PS152; PS154; PS155;	30 (19.2%)
MAA	PS007; PS116; PS120; PS131; PS134; PS138; PS140; PS142; PS148; PS152;	10 (6.4%)
QUA	PS007; PS014; PS058; PS097; PS110; PS116; PS129; PS140; PS142; PS148;	10 (6.4%)

GBL approaches found are varied in genres and formats. The most recurring type of digital games were simulation-based games. Besides, we found non-digital games such as card-games, board games, role-playing games, and group activities. Table 6 describes some types of games found. Similar to the findings of other authors [2], we observed digital games are predominantly single player games, while non-digital games are mostly multiplayer games.

We found 10 recurring games in the primary studies (Table 7). SimSE (PS035, PS063, PS077, PS092, PS096) is the most cited digital serious game (5 primary studies). It simulates software engineering processes and it was designed to support learners in situations that require understanding and handling of software process issues [9]. It was also a direct inspiration for three other games described in primary studies PS017, PS043, and PS079. Problems and Programmers (PS021, PS022, PS057, PS070) is a non-digital card game that simulates the software engineering process; it was designed to teach process issues non-sufficiently highlighted by lectures and projects. SimSoft (PS076, PS084) is a game designed to teach and educate players about some dynamic complexities of software development projects in a safe and inexpensive environment. ARMI (PS027, PS028) is a card-based non-digital game for teaching risk management in software processes; it was later ported to a web application version – ARMI 2.0 (PS027). eRisk (PS059, PS090) focuses on improving the learning process in risk management in software projects, more specifically in the Planning, Control and Monitoring (budget, time schedule and software quality). Pex4Fun (PS033, PS038, PS067, PS083) and its successor CodeHunt (PS033) focus on developing students' abilities in software engineering

Table 6
Types of games found.

	Genres and formats	Examples
Digital games	Simulation based games	PS030, PS054, PS062, PS069, PS071, PS074, PS075, PS077, PS079, PS092, PS096
Non-digital games	Card-games	PS057, PS068, PS070
	Board games	PS034, PS051, PS094
	Role-playing games	PS014, PS015, PS085
	Group activities	PS050, PS064, PS082, PS091

Table 7
Recurring games in the selected primary studies.

Recurring games	Primary studies
ARMI	PS027, PS028
eRisk	PS059, PS090
Groupthink	PS090, PS043
Pex4Fun	PS033, PS038, PS067, PS083
Problems and Programmers	PS021; PS022; PS057; PS070
SDSim	PS020; PS026
SimSE	PS035; PS063; PS077; PS092; PS096
SimSoft	PS076, PS084
Software Hut	PS064; PS082
ViRPlay3D	PS065; PS073; PS095

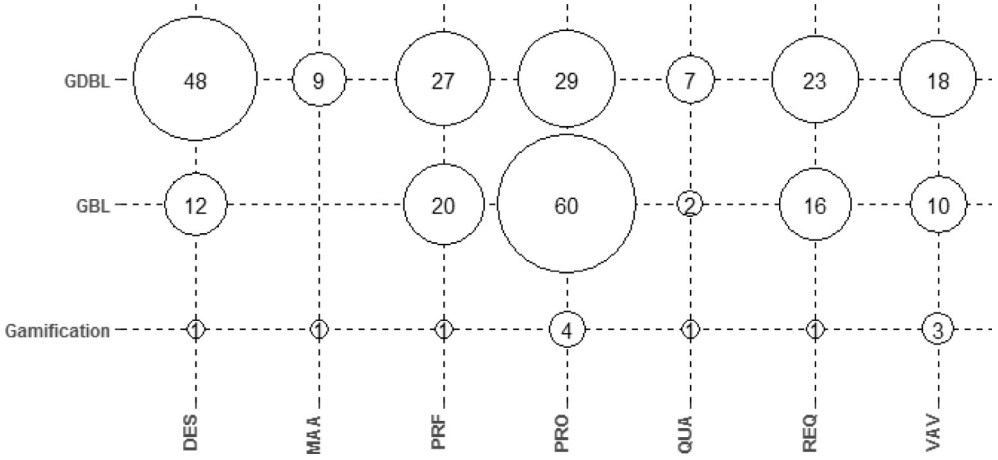

Fig. 5. Coverage of software engineering knowledge areas per game-related method.

Table 8
Technologies used in GDBL approaches.

Game development frameworks, engines, and libraries	Primary studies
AgentWEB	PS103
Construct 2	PS123
CryEngine 3	PS130
EiffelMedia	PS104
Game Maker	PS97; PS142; PS145
Jypeli library	PS109
Panda3D	PS147
PlayN library	PS135
POP Framework	PS116
Quest3D	PS130
Sheep	PS113
Slick Game Engine	PS144
Torque	PS129; PS148
Unity3D	PS130
Unreal Tournament	PS124; PS148
Valve Source	PS102; PS151
WarCraft III engine	PS124
WebDriver	PS124
XNA	PS101; PS114; PS117; PS118; PS125; PS127; PS150; PS156
XQUEST	PS100; PS153

and programming through debugging and unitary tests [22].

4.5. RQ4 - technologies used in GDBL approaches for software engineering education

This section discusses the results for the fourth research question.

> *RQ4. What technologies have been used to support GDBL approaches?*

Our results show that the use of Game Development Frameworks (GDF), libraries, and engines is recurring resource in GDBL approaches. Table 8 identifies the technologies supporting GDBL approaches mentioned in the primary studies. Although the specific technologies used are not the main goal of our mapping, the number of studies using GDF indicates that the analysis and development of GDF for education are possible topics of interest for future research in software engineering education.

Twenty-eight primary studies (46.7%), out of the 60 describing GDBL approaches, mentioned the use of specific technologies for game development. We mapped frameworks, engines, and libraries for game development used in educational game development projects (Table 8). Microsoft XNA Framework (a free framework to support game development) is the most recurrent GDF found. We also found occurrences of famous game engines such as CryEngine 3, Unreal Engine, Source Engine, Game Maker, and Unity3D. Authors also described GDF specifically designed to support students' projects: XQUEST, AgentWEB, Sheep, and EiffelMedia.

Wu et al. [23] (PS101) suggested that "the main benefit of using a GDF as a teaching aid is that it can be a motivating initiative in courses to learn about various topics within computer science and software engineering, and that it is useful for learning new skills and methods, and for testing and rehearsing theory by applying previous knowledge in a project". The student motivation factor is also defended by Wang and Wu [24] (PS100).

The studies also support the use of GDF for their easiness of use. Wu et al. [25] (PS113) claimed that their experience of using a specific library for game development enabled the students to focus more on the actual learning goal (software architecture) and less on issues related to the programming. As described by Wu et al. [23] (PS101), the GDF they chose (XNA) was easy to use, required little time to develop, and supported different types of game development. In PS118, Denninger and Schimmel [26] observed that XNA framework relieves programmers from repetitive and time-consuming tasks, allowing them to develop complete games and to gain experience with the process of software development.

In contrast, Wu et al. [23] (PS101) claimed that the main disadvantages are the student might spend too much time on game-specific issues rather than the expected learning goals and that the project results might be difficult to compare. In PS117, the authors suggested that time, cost and expertise are significant barriers to experimenting with GDF in educational settings, and there are limitations to what skills can be acquired using these technologies [27].

Table 9
Programming languages used in GDBL approaches.

Programing languages	Primary studies
APL	PS143
C	PS131
C#	PS106; PS107; PS120; PS127; PS129; PS156
C++	PS107; PS131; PS137; PS140; PS152; PS156
Eiffel	PS104; PS149;
Java	PS099; PS098; PS112; PS132; PS133; PS138; PS139; PS142; PS144; PS147; PS152; PS155
Java (Android)	PS105; PS128, PS113
Lua	PS131
Python	PS121; PS147
Swift	PS111
Visual C++	PS148

Regarding programming languages used in GDBL projects (Table 9), Java was most cited (12 primary studies). Besides, C++, and C# were also frequently cited by in primary studies described, more frequently. Less often, we found mentions of the use of APL, C, Eiffel, Java for Android, Lua, Python, Swift, and Visual C++.

4.6. RQ5 - Gamification elements used in software engineering education

This section discusses the results for the last research question.

> *RQ5. What game elements have been used for the Gamification of software engineering education?*

In order to understand how Gamification is being used in the context of software engineering education, we analyzed which game elements are used in each primary study (Table 10). Leaderboards, Points, and Levels are the recurrent game mechanics found (5, 4, and 4 primary

Table 10
Game elements identified in primary studies.

Game element	Primary studies
Leaderboards	PS003, PS002, PS005, PS007, PS006
Points	PS010, PS003, PS002, PS005, PS154
Milestones, Levels, Paths and Progress	PS010, PS003, PS007, PS006, PS154
Competition	PS001, PS003, PS005, PS007
Collaboration, Altruism, Teams	PS010, PS003, PS007, PS154
Rewards	PS010, PS003, PS154
Challenges	PS001, PS03
Storytelling	PS010, PS007, PS154
Achievements	PS003
Time Pressure	PS007
Awards	PS001
Quests	PS010, PS154
Gifts & Sharing	PS003
Status	PS003
Badges	PS005
Feedback	PS007
Marketplace and Economies	PS007

studies, respectively). Competition is the most recurring game design principle identified (4 primary studies).

Dechieva et al. [28] stated that there is not a commonly agreed classification of game design elements. The analysis of the primary studies supported this claim, as we observed a lack of standard definitions of game elements. Deternding et al. [12] identified game elements in varying levels of abstraction and proposed a classification scheme based on 5 levels (from concrete to abstract): (i) “Game interface design patterns”; (ii) “Game design patterns and mechanics”; (iii) “Game design principles and heuristics”; (iv) “Game models”; and (v) “Game design methods”. Zichermann and Cunningham [29] categorized game elements into mechanics, dynamics, and aesthetics. Dechieva et al. [28] proposed a classification of game elements in gamified educational contexts organized in two levels: (i) “Game Mechanics” and (ii) “Design Principles”. The former is a combination of the first two levels of Deterding's classification and refers to the more concrete representation of game elements, such as leaderboards, badges, point, and levels. The latter is a combination of the third and fourth level of Deterding's classification and it is concerned with abstract elements used in games, such as Competition and Status. Therefore, given this lack of standardization in the primary studies, we did not use any of the aforementioned classification schemes. In Table 10, we mapped only the game elements as they were mentioned in the primary studies.

PS01 uses weekly challenges to motivate students on applying eXtreme Programming practices to their project. The students had to compete for a “challenge cup” award. PS010 exposes students to software testing using a game-like environment, HALO (Highly Addictive, sociaLly Optimized) Software Engineering. HALO uses MMORPG (Massively Multiplayer Online Role-Playing Game) motifs to create an engaging and collaborative development environment. Students performed software testing tasks as “quests” contextualized in a fictional storyline. When students complete “quests”, they gain “experience points” and they “level up”. Players gain Social rewards (titles and levels) when they complete achievements (such as successfully closing over 500 bugs). Students can track their progress and choose the quests they want to perform in any order. Quests can be individual or requiring a developer to form a team and work collaboratively towards their objective. HALO is also used in the hybrid approach described in PS154.

PS007 describes eight MMORPG elements incorporated in a project based software engineering capstone course: (i) Narrative Context (Epic Story); (ii) Feedback; (iii) Reputations; (iv) Rank, and Levels; (v) Marketplaces and Economies; (vi) Competition under explicit and enforced rules; (vii) Teams; and (viii) Time Pressure. In PS03, the authors provided in depth details of the setup for a gamified classroom for the subject of software engineering. The authors developed a software platform to support game elements in the classroom, such as, Paths, Purpose, Autonomy, Levels, Progress Bar, Points, Heroes, Peers, Interaction, Collaboration, and Personas. The authors did not introduce Gamification to the software engineering topics being taught, but to the classroom structure, where students could choose (Autonomy) which area of study (Paths and Purpose) they want to take, and keep track of their progress (Progress Bars). When students complete tasks, they gain “experience points” (Points) and can advance to a next level in a given area of study. Students can interact with each other (Peers, Interaction, and Collaboration) and are rewarded for providing help on completing tasks (Heroes/Altruism). The experience in PS03 was a failure and the authors redesigned the course in PS08, maintaining Gamification elements, but not emphasizing them.

PS002 proposes the use of Gamification to increase students’ utilization of tools during educational software development. They used a plugin for the project management tool “Redmine” to support Points and Leaderboards. PS005 describes the experiment of implementing Gamification techniques into software engineering and service-oriented architecture courses. In first course, the authors used Points and Leaderboards and promoted competition. In the later course, the authors used Points, Leaderboards, and Badges. Additionally, the authors adopted a physical representation for Points, in the form of Poker Chips.

PS006 describes an experiment to encourage students to make more frequent commits to version control system, in a software project course. The authors proposed a leaderboard based on the number of commits from each students and established milestones or thresholds that would trigger messages congratulating students and teams that reached the specified number of commits. PS004 proposes the incorporation of over 20 Gamification elements in modern software engineering courses. The authors organize Gamification elements in three categories: i) Progression Gamification Techniques; ii) Feedback Gamification Techniques; and iii) Behavior Gamification Techniques. However, the authors did not provide enough detail about which elements they used in a pilot study briefly described in the study.

5. Discussion

This section discusses our main findings and insights about the use of game related methods in software engineering education. Section 5.1 discusses the use of GBL approaches in software engineering education. Section 5.2 discusses the use of GDBL approaches in software engineering education. Section 5.3 discusses the role of Gamification in software engineering education.

5.1. On the use of games as learning tools in software engineering education

The use of games as learning tools is not new. Our results are an evidence that serious games are a common approach to provide variety in teaching and learning approaches. However, many authors claim that this learning method is not a substitute for traditional classes, but a complimentary tool for addressing specific learning outcomes.

In the case of the serious game Problems and Programmers (PS021, PS022, PS057, PS070), Baker et al. [30] allege that, “when used in conjunction with lectures and projects, Problems and Programmers allows students to gain a thorough understanding of real-world lessons that might otherwise have been poorly understood or overlooked altogether”. In the case of SimSE (PS035, PS063, PS077, PS092, PS096), evaluations of the game indicated that it does successfully help students learn software process concepts, but critical lessons learned are the necessity of both providing students with proper instruction in playing the game, and providing students with a set of guiding questions to answer while playing the game [9].

Caulfield et al. [31] suggest “while games are useful pedagogical tools and are well-received by players, they are not sufficient in themselves and must be supplemented by other learning devices”. Heikkilä et al. [32] go further discussing that the efficiency of educational games in software engineering is debatable. The authors warn about the risks of using educational games: (i) Games introduce overhead, such as learning the rules and setting up the game and may require more time to complete than other methods; and (ii) gameplay mechanics may draw the players’ attention away from actual learning goals. Therefore, the efficiency of GBL approaches is not consensus, but there is a general thought that games are useful resources in software engineering education. The evaluation of educational games is a research topic that has been explored by several authors [33–36], but was out of the scope of this mapping study.

The proposal of serious games is often inspired by the necessity to motivate students using interesting, concrete, and convincing examples. For instance, the topic of global software development is difficult to exemplify using student projects for several limitations, but the use of simulation games may provide tools to contextualize the issues in globally distributed development processes, as described in PS054 and PS096. Oliveira et al. [37] also stated this problem in the motivation of eRisk (PS059, PS090). The authors claim that “traditional approaches usually adapt and simplify problems, thus, reducing their relevance”

and that "Problems are also usually linked to prefabricated solutions that do not help them to develop their own ideas to tackle problems".

Regarding the coverage of SE 2014 knowledge areas, 60 primary studies presenting GBL approaches support "Software Process". We believe that providing concrete examples/experiences of the varied aspects of software process and project management is a challenge, and GBL approaches help educators in addressing this issue. SE 2014 [1] affirms that "process is difficult to motivate until students understand central challenges such as scale, complexity, and human communication that motivate all of software engineering", we add to that the limitations of typical classroom assignments and capstone projects: they are often limited by time, scope and size of the teams. Consequently, it is difficult to expose students to the consequences of poor application of software process concepts in short periods imposed by the format of courses and student projects. Therefore, games might be useful to provide abstractions that emphasize specific aspects related to software process, and to demonstrate the benefits of applying the concepts being taught and the undesirable consequences of failing to apply them.

Additionally, SE 2014 proposes that "software process should be central to the curriculum organization and to students' understanding of software engineering practice" [1]. This guideline suggests that software process is both a focal and a crosscutting topic in software engineering education. Therefore, given these observations, the great emphasis on "Software Process" is understandable.

5.2. On the use of game development as a learning context for software engineering education

The context of game development for software engineering education takes advantage of two characteristics: (i) it addresses the need of practical experiences in software engineering education; and (ii) it relies on the popularity of games, both as a domain familiar to learners, and as an engaging domain for learning.

The necessity of providing real world experience of software development to students is a recurring theme on SE 2014, and several of its guidelines address this matter [1]. Curriculum Guideline 5 suggests that "students also need practical material to be taught early so they can gain maturity by participating in real-world development experiences (…)" [1]. Curriculum Guideline 10 discusses the multiple dimensions of the problem-solving aspect of software engineering, and suggests that "problem solving is better learning through practice and taught by example" [1]. Curriculum Guideline 17 suggests the need of using interesting, concrete and convincing examples to motivate students. Finally, Curriculum Guideline 14 objectively declares "the curriculum should have a significant real-world basis" [1].

The use of game development as a method to introduce software engineering provides students with hands-on experience and exposes the learners to practical issues of real-world software development. The results of RQ2 (Section 4.3) show that educators and researchers have found this method particularly useful to support the practice of software design topics. The use of the game development domain also helps students in developing an understanding of a specific software application domain. SE 2014 suggest that it is desirable to develop skills in at least one application domain, but warns that great care must be taken to define good balance in a way that depth is not sacrificed in either software engineering or the domain explored [1]. This issue was discussed in Section 4.5.

Our results also revealed a relevant number of studies on the use of game development frameworks to support students in GDBL approaches in software engineering education. This perspective is adherent to SE 2014 Curriculum Guideline 12 [1]: "The curriculum must be taught so that students gain experience using appropriate up-to-date tools, even though tool details are not the focus of the learning". While game development frameworks are useful to support learners in developing games, giving them opportunity to focus on software engineering issues rather than in technical aspects of game development, they also provide students with experience in using tools from professional game development industry.

As discussed in Section 4.3, GDBL approaches usually cover more topics at once than GBL approaches which usually focus on fewer learning topics. We observed that GDBL approaches use game development projects to expose students to the many activities of software development life cycle. SE 2014 suggests that "important efficiencies and synergies can be achieved by designing curricula so that several types of knowledge are learned at the same time". In our results, for instance, we observed that modeling (in the context of the knowledge area "Software Modeling and Analysis") was mostly covered in GDBL approaches, given the fact that in the primary studies PS116, PS120, PS131, PS134, PS138, PS140, PS142, PS148, and PS152 used a problem-based approach to motivate students in applying UML.

5.3. On the use of Gamification in software engineering education

In comparison to GBL and GDBL approaches, few researchers explored Gamification in software engineering education, which still is an open field for more experiments and proposals. This is supported by the small number of primary studies found that address the use of Gamification in this context.

It is important to understand that, different from GBL and GDBL approaches, by definition, Gamification is a technique that requires a non-game context (*e.g.* the classroom activities, capstone projects, or the use of a tool) in which game elements are introduced. Therefore, in the context of software engineering education, it is not a "stand-alone" educational tool. Consequently, the purpose of Gamification in the primary studies was not to objectively make students learn or directly supporting the development of new skill. Gamification was used as a device to motivate students in conforming to desired behaviors, such as the more frequent use of specific tools, acquiring the habit of applying specific techniques, or being more participative in the classroom. Gamification was also used as a strategy to induce learners to use specific software engineering abilities or practices, by promoting competition or systematically rewarding learners as they perform expected actions or show expected behaviors. Therefore, Gamification is a relevant strategy to support students in developing an appreciation of the importance of continued learning and in acquiring habits for professional software development.

The studies PS002, PS003, PS007, and PS08 are concerned with transforming the classroom experience, motivating and engaging students. The studies PS001, PS002, PS006, PS010, and PS154 uses Gamification elements directly on software engineering practices, aiming to promote the development of new skills or incorporation of good behaviors in recurring activities. These proposals are aligned with the goals of studies proposing the use of Gamification in professional software development [13], as they use game elements to increase the use of tools, best practices, and expected behaviors in daily activities related to software engineering.

Regarding the use of game elements, Points, Levels, and Leaderboards are the most used game elements in the primary studies. Authors should be aware of the risk of adopting Gamification as a simple "pointification system", as the technique has more to offer [38]. The primary studies show that Competition is an important game design principle for engaging people. Points and Leaderboards are effective when used to promote competition among learners.

A recurring concern identified in the primary studies is the difficulty of adapting Gamification to each context. The primary studies show that each context require great effort from the educators to setup game elements appropriately, and still there is a chance of failure, as shown in PS03. Besides that, Gamification has its own risks, such as the chance of students trying to "game the system", i.e. students might become more engaged in exploiting the rules to "win the game", than following the expected flow of activities. This phenomenon is discussed in PS006 and PS007. Additionally, assessing the impact of Gamification in

software engineering education is still a challenge, as the technique is more related to motivation and engagement toward specific behaviors than in learning properly.

6. Threats to validity

The validity of the results is a key issue to be considered when performing of a systematic mapping study. This section presents and discusses the different validity threats related to this study with respect to the four groups of common threats to validity [39]: internal validity, external validity, construct validity, and conclusion validity. We also discuss how the threats were addressed to minimize the likelihood of their impact on our results.

Internal validity. Internal validity concerns the question whether the effect is caused by the independent variables or by other factors [39]. In this sense, a limitation of this mapping study concerns the reliability of its results. The reliability has been addressed as much as possible by involving three researchers, and by having a strict protocol which was piloted and hence evaluated. If the study is replicated by another set of researchers, it is possible that some studies that were removed in this review could be included. Similarly, some studies we selected could be excluded by others. However, in general we believe that the internal validity of this study is high given the use of a systematic procedure, consultation with the researchers in the field, involvement, and discussion between three researchers.

External validity. External validity concerns the ability to generalize the results to other environments, such as to actual classrooms [39]. A major external validity to this mapping study was the identification of primary studies. The search for the primary studies was conducted in four large scientific databases, namely ACM Digital Library, IEEE Xplore, EI Compendex, and Scopus, in order to capture as many relevant studies as possible and to avoid all sorts of bias. However, the quality of search engines could have influenced the completeness of the identified primary studies. For instance, our search may have missed those studies whose authors have used different terms to specify their game-based approach for Software Engineering education. In addition, we search for relevant terms only in the title, abstract, and keywords of their papers. Regarding the possibility to replicate this study, the study selection process was clear enough to be executed by two researchers, leading to a low number of conflicts (less than 7% of the selected studies). Those conflicts were resolved with the support of a third researcher in the second selection phase. Additionally, the research protocol was piloted in a previous study [5], and, therefore, we believe it is clear enough to be reproduced.

Construct validity. Construct validity reflects to what extent the operational measures that are studied really represent what the researcher has in mind and what is investigated according to the research questions [39]. Four researchers of this study are experts in software engineering education and three have experience in research on game-based education. We are not aware of any bias we could have had during the construction of the study protocol. However, from the selection perspective, a construct validity threat could be biased judgment. In this study, the decision of which studies to include or to exclude and how to categorize them could have been biased and thus pose a threat. To minimize this threat, both the processes of inclusion and exclusion were piloted by at least three researchers. Furthermore, potentially relevant studies that were excluded were documented for further verification.

Conclusion validity. Threats to conclusion validity are related with issues that affect the ability to draw the correct conclusions from the study [39]. From the reviewers' perspective, a potential threat to conclusion validity is the reliability of the data extraction from the primary studies, since not all information was obvious to answer the research questions and some data had to be inferred. Therefore, in order to ensure the validity, sometimes cross-discussions among the paper authors took place to reach a common agreement. Furthermore, in the event of a disagreement between the two researchers, a third reviewer acted as an arbitrator to ensure a position to be reached.

7. Related work

This section describes the related research on the use of game-related methods for software engineering education. We focus our discussion on similar secondary studies on the relation of game-related learning methods and software engineering education. We found several studies analyzing specific game-related methods or different approaches in software engineering education. However, to the extent of our research, our study differentiates from previous study providing an in-depth discussion on the classification of game-related methods in SE education, their differences, and how these methods have been used to support the specific SE education knowledge areas.

Marques et al. [3] identified 173 primary studies describing practical approaches for software engineering education. They only found 12 primary studies describing games for SE education. However, some of the teaching approaches found, are also mentioned in the GDBL primary studies we found in our study: learn by doing, problem based learning and simulation (projects that simulate a part of a real-life project). The authors conclude, and we have also discussed, that bridging the gap between theory and practice is still a major challenge in SE education. We add that game-related educational methods are a relevant tool for addressing this challenge.

Malik and Zafar [4] performed a similar analysis, mapping 70 primary studies on SE education to three reference curricula; (i) SE 2004, (ii) GSwERC and (iii) SWEBOK. Their study identified that knowledge areas related to software engineering management is the most recurring topic addressed by the primary studies. However, the authors did not identify which educational methods are used in each primary study. In SE 2014 [1], software management topics are included in "Software Process" knowledge area, the most supported topic in our study. Therefore, we can affirm that the findings of Malik and Zafar [4] are similar to ours considering the specific scope of game-related educational methods.

Regarding the classification of game-related approaches for software engineering education, Kosa et al. [7] identified 53 primary studies between 2003 and 2015. The authors classified primary studies in five categories: (i) Games that learners/students play; (ii) Games that learners/students develop as projects; (iii) Curriculum proposals; (iv) Developing/Coming up with new approaches, tools, frameworks or suggestions; and (v) others. In our study, we analyzed a greater number of primary studies and we focused in the classification of the game-related methods and mapping these methods to knowledge areas. Additionally, we describe "Gamification" as a recent method used in SE Education. The findings that both studies have in common is the need for need of additional empirical data, and the lack of clear guidelines for the design and evaluation of game-related approaches.

Considering the use of games as learning tools (GBL, in our study), Caulfield et al. [6] identified 36 primary studies on the use of games and simulations for educational or training purposes in software engineering or software project management. The authors mapped the primary studies to SWEBOK areas and their results show that most of the primary studies address software engineering management and development processes areas. Our results are similar in the sense that primary studies on game-related educational methods have great emphasis on SE 2014 "Software Process" knowledge area (which includes both process and project management). However, specific methods have been used with different focus, as the case of GDBL studies with greater focus on "Software Design" topics. Another similarity in our findings is that the study of Caulfield et al. [6] has shown that most studies followed a non-experimental design, and many had very small sample sizes. The authors conclude that enough evidence exists to say that educators could include serious games in their courses as a useful and interesting supplement to other teaching methods.

A systematic literature review was performed on the use of GDFs in education to summarize a guideline on how to use GDBL on a curriculum [10]. The authors selected 34 studies and analyzed then according to three aspects: (i) pedagogical context and teaching process, (ii) technical aspects, and (iii) evaluation results in relation to the objective of their study. Their findings showed that GDBL has the potential power to help students to learn different curriculums, and they provided a guideline for integrating a GDF in learning with teaching strategies. Our study also showed that using GDFs and GDBL as teaching aids could help motivate students in software engineering education, and that GDF is a relevant investigation topic in GDBL.

Regarding Gamification, Pedreira et al. [13] identified 29 primary studies describing the use of this technique in the professional context of software engineering. Despite the fact that the authors excluded studies focusing on education and training, results showed that software requirements, software development and software testing are the areas which attracted the greatest interest in the field of Gamification. Similar to what we found in the context of education, the authors suggest that the existing research on Gamification applied to SE is very preliminary or even immature, with a majority of publications in workshops or conferences, and few of them offer empirical evidence on the impact of their proposals on user engagement and performance. The authors point the necessity of further research providing empirical results about the effect of Gamification. Therefore, these results reinforce our position that Gamification is still an open field for research, especially considering education and training, given the decreased number of studies found.

8. Conclusion and future work

This study presented a systematic mapping study on the use of game-related methods to support software engineering education. We mined four scientific databases (IEEE Xplore, ACM Digital Library, Scopus, and EI Compendex) and retrieved 156 primary studies, published between 1974 and June 2016. We classified the approaches in three types: (i) game-based learning (GBL), as the use of games as learning tools for software engineering education (88 primary studies); (ii) game development based learning (GDBL), as the use of game development as a domain for software engineering education (60 primary studies); and Gamification, as the use of game elements, rather than full games, in the context of software engineering education (11 primary studies).

It is important to notice that these educational methods are not specific to software engineering education. That is, game-related educational methods have been used to address educational problems in other fields of computing education [40,41]. However, this paper focuses on the specific problems of software engineering education that have been addressed by such approaches. Some recurring issues in software engineering education that motivate the use of game-related approaches are related to the applied nature of software engineering, that requires learners to experience real-world issues of software development to acquire appreciation for software engineering concepts. It is difficult to provide convincing examples of some aspects of software engineering concepts in traditional lectures and students' projects, given the limitations of these formats. Game-related approaches have been used to overcome some of these limitations. Gamification adds a new possibility in this context: developing behaviors in students towards the application of software engineering concepts and good practices during the learning process.

Mapping the learning goals of the primary studies to software engineering knowledge areas of SE 2014, we found that "Software Process", "Software Design", and "Professional Practice" are the most the most recurring topics covered, with respectively 92, 60 and 48 primary studies supporting them direct or indirectly. When we analyzed how the game related methods support each knowledge area, we observed that "Software Process" is more supported by GBL approaches, while "Software Design" is more supported by GDBL approaches. "Professional Practice" is mostly supported as a secondary objective. In contrast, "Software Modeling and Analysis" and "Software Quality" are the least supported knowledge areas, regarding game related methods.

We observed that research on Gamification in software engineering education is still preliminary. Nonetheless, the primary studies analyzed show it is a promising area for further research, and software development activities are suitable scenarios for gamified experiences. More empirical studies are necessary to assess the relevancy of this technique in software engineering education.

The analysis of the primary studies in relation to the curriculum guidelines proposed in SE 2014, shows that game-related approaches are useful to support software engineering education, but they are often described as complimentary resources rather than stand-alone learning methods replacing traditional lectures. GBL approaches are useful in providing students with additional resources to understand or apply knowledge in topics that would be difficult to exemplify or simulate in traditional lecture formats. GDBL approaches provide hands on experience in software development, where students can apply software engineering concepts in a domain they are familiar with. Gamification is still a novel approach useful in imbuing specific behaviors and motivating students in applying software engineering concepts.

The main challenge in the execution of this systematic study was the lack of standardization in the studies of this research field. A considerable number of studies lack clear, structured learning goals. Few studies refer to curriculum standards such as SE 2014. In the context of GBL approaches, there was little classification standard in relation to game genres. Finally, the game elements in Gamification also lack standardization. Therefore, we advise researchers in this field to seek standardization.

We expect to provide educators, learners, and researchers with an understanding of the role of game-related methods in software engineering education. We have also found some open challenges and opportunities for future research. First, there is few empirical data regarding the evaluation of game-related methods in software engineering education. Therefore, the community should strive for systematic and replicable evaluation methods. Second, Gamification is new and, so, there is a special need for large scale experiments on the use of this technique in software engineering education. Further analysis on its usefulness in this context in needed. Third, game development frameworks are useful resources in GDBL approaches and, therefore, they may be relevant to further explore the analysis and development of GDF for educational purposes. Finally; the use of game-related methods for some knowledge areas (such as, software quality, and software modeling and analysis) can still be further explored. As future work, we are working on evaluating the use of Gamification to support the development of professional skills in software engineering students [19].

Acknowledgments

This research was partially supported by Brazilian funding agencies: CNPq (grant 424340/2016-0 and 142022/2017-9), CAPES, and FAPEMIG (grant PPM-00651-17).

Appendix A. List of selected primary studies

[PS001] B. S. Akpolat and W. Slany, "Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification", IEEE Conference on Software Engineering Education and Training, 2014

[PS002] A. L. D. Buisman and M. C. J. D.Van Eekelen, "Gamification in educational software development", Computer Science Education Research Conference, 2014

[PS003] K. Berkling and C. Thomas, "Gamification of a software engineering course and a detailed analysis of the factors that lead

to it's failure", International Conference on Interactive Collaborative Learning, 2013
[PS004] V. Uskov and B. Sekar, "Gamification of software engineering curriculum", Frontiers In Education Conference, 2014
[PS005] M. Laskowski, "Implementing gamification techniques into university study path - A case study", IEEE Global Engineering Education Conference, 2015
[PS006] L. Singer and K. Schneider "It was a bit of a race: Gamification of version control", International Workshop on Games and Software Engineering, 2012
[PS007] J.N. Long and L. S. Young, "Multiplayer on-line role playing game style grading in a project based software engineering technology capstone sequence", ASEE Annual Conference and Exposition, 2011
[PS008] C. Thomas and K. Berkling, "Redesign of a gamified Software Engineering course", International Conference on Interactive Collaborative Learning, 2013
[PS009] W. Q. Qu, Y. F. Zhao, M. Wang and B. Liu, "Research on teaching gamification of software engineering", International Conference on Computer Science Education, 2014
[PS010] J. Bell, S. Sheth and G. Kaiser, "Secret Ninja Testing with HALO Software Engineering", International Workshop on Social Software Engineering, 2011
[PS011] C.-H.a Su and C.-H.b Cheng, "3D game-based learning system for improving learning achievement in software engineering curriculum", Turkish Online Journal of Educational Technology, 2013
[PS012] J. Srinivasan and K. Lundqvist, "A constructivist approach to teaching software processes", International Conference on Software Engineering, 2007
[PS013] E. Knauss, K. Schneider and K. Stapel, "A Game for Taking Requirements Engineering More Seriously", International Workshop on Multimedia and Enjoyable Requirements Engineering, 2008
[PS014] L. Ohlsson and C. Johansson, "A Practice Driven Approach to Software Engineering Education", IEEE Transactions on Education, 1995
[PS015] S. Zuppiroli, P. Ciancarini and M. Gabbrielli, "A role-playing game for a software engineering lab: Developing a product line", IEEE Conference on Software Engineering Education and Training, 2012
[PS016] T. A. B. Galvao, F. M. M. Neto, M. F. Bonates and M. T. Campos "A serious game for supporting training in risk management through project-based learning", Communications in Computer and Information Science, 2012
[PS017] T. Wang and Q. Zhu, "A software engineering education game in a 3-D online virtual environment", International Workshop on Education Technology and Computer Science, 2009
[PS018] A. Hoffmann, "A Trainer's Guideline to Teaching Soft Skills Using Improvisation Theater: A Workshop Format Exemplified on a Requirements Engineering Game", European Conference on Pattern Languages of Programs, 2012
[PS019] T. D. Lynch, M. Herold, J. Bolinger, S. Deshpande, T. Bihari, J. Ramanathan and R. Ramnath, "An agile boot camp: Using a LEGO®-based active game to ground agile development principles", Frontiers In Education Conference, 2011
[PS020] T. M. Connolly, M. Stansfield and T. Hainey, "An application of games-based learning within software engineering", British Journal of Educational Technology, 2007
[PS021] A. Baker, E. O. Navarro and A. van der Hoek, "An experimental card game for teaching software engineering", IEEE Conference on Software Engineering Education and Training, 2003
[PS022] A. Baker, E. Oh Navarro and A. Van Der Hoek "An experimental card game for teaching software engineering processes", Journal of Systems and Software, 2005
[PS023] A. Y. K. Chua and R. S. Balkunje, "An exploratory study of game-based m-learning for software project management", Journal of Universal Computer Science, 2012
[PS024] R. Atal and A. A. Sureka "A software engineering simulation game for teaching practical decision making in peer code review", CEUR Workshop Proceedings, 2015
[PS025] G. Rong, He Zhang and D. Shao, "Applying competitive bidding games in software process education", IEEE Conference on Software Engineering Education and Training, 2013
[PS026] T. M. Connolly, M. Stansfield and T. Hainey "Applying games-based learning to teach software engineering concepts" Webist 2007 - 3rd International Conference on Web Information Systems and Technologies, 2007
[PS027] P. Sonchan and S. Ramingwong, "ARMI 2.0: An online risk management simulation" International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2015
[PS028] S. Ramingwong and L. Ramingwong "ARMI: A risk management incorporation", International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2014
[PS029] L. Ganesh, "Board game as a tool to teach software engineering concept - Technical debt", IEEE International Conference on Technology for Education, 2014
[PS030] A. Calderón and M. Ruiz, "Bringing real-life practice in software project management training through a simulation-based serious game", International Conference on Computer Supported Education, 2014
[PS031] O. Alsaedi, Z. Toups and J. Cook, "Can a team coordination game help student software project teams?", International Workshop o Cooperative and Human Aspects of Software Engineering, 2016
[PS032] C. M. Z. Jaramillo, "Communication and traceability game: A way to improve requirements elicitation process teaching", Revista Facultad de Ingenieria, 2010
[PS033] N. Tillmann, J. de Halleux, T. Xie and J. Bishop, "Constructing Coding Duels in Pex4Fun and Code Hunt", International Symposium on Software Testing and Analysis, 2014
[PS034] C. G.a Von Wangenheim, R. Savi and A. F.a Borgatto, "DELIVER! - An educational game for teaching Earned Value Management in computing courses", Information and Software Technology, 2012
[PS035] E. Oh Navarro and A. Van Der Hoek, "Design and evaluation of an educational software process simulation environment and associated model", IEEE Conference on Software Engineering Education and Training, 2005
[PS036] Y.-W.a Wu, S.-H.b Hsu, S.-L.a Li, W.-H.a Wu and Y.-M.b Huand, "Digital game as a learning approach to enhance practice lesson in software engineering course", International Conference on Computers in Education, 2009
[PS037] J. Pieper, "Discovering the essence of Software Engineering an integrated game-based approach based on the SEMAT Essence specification", IEEE Global Engineering Education Conference, 2015
[PS038] T. Xie, N. Tillmann and J. De Halleux, "Educational software engineering: Where software engineering, education, and gaming meet", International Workshop on Games and Software Engineering, 2013
[PS039] S.-T. Huang, W.-H. Lin and M.-C Hsu, "Embracing business context in pedagogical simulation games - A case with process disciplined project management" IEEE-CS Conference on Software Engineering Education and Training Workshop, 2008
[PS040] C.a b Gresse Von Wangenheim, M.a Thiry and D.a Kochanski, "Empirical evaluation of an educational game on software measurement", Empirical Software Engineering, 2009

[PS041] C. Liu and B. Wu, “Enabling collaborative learning with an educational MMORPG”, IEEE International Games Innovation Conference, 2011
[PS042] K. Shaw and J. Dermoudy, “Engendering an Empathy for Software Engineering”, Conferences in Research and Practice in Information Technology Series, 2005
[PS043] E. Ye, C. Liu, J. A. Polack-Wahl, “Enhancing software engineering education using teaching aids in 3-D online virtual worlds”, Frontiers In Education Conference, 2007
[PS044] Y.a Wautelet and M.b Kolp, “E-SPM: An online software project management game”, “International Journal of Engineering Education”, 2012
[PS045] C. Szabo, “Evaluating GameDevTycoon for teaching Software Engineering”, ACM Technical Symposium on Computer Science Education, 2014
[PS046] R. Van Solingen, K. Dullemond and B. Van Gameren, “Evaluating the effectiveness of board game usage to teach GSE dynamics”, IEEE International Conference on Global Software Engineering, 2011
[PS047] T. Hainey, T. M. Connolly, M. Stansfield and E. A. Boyle “Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level”, Computers and Education, 2011
[PS048] F. B. V. Benitti, and L. Sommariva, “Evaluation of a Game Used to Teach Usability to Undergraduate Students in Computer Science”, Journal of Usability Studies, 2015
[PS049] R. O. Chaves, C. G. Von Wangenheim, J. C. C. Furtado, S. R. B. Oliveira, A. Santos and E. L. Favero, “Experimental evaluation of a serious game for teaching software process modeling”, IEEE Transactions on Education, 2015
[PS050] J. Bergin and F. Grossman, “Extreme construction: Making agile accessible”, AGILE Conference, 2006
[PS051] R. Smith and O. Gotel, “Gameplay to Introduce and Reinforce Requirements Engineering Practices”, IEEE International Requirements Engineering Conference, 2008
[PS052] B. Scharlau, “Games for teaching software development”, Annual Conference on Innovation and Technology in Computer Science Education, 2013
[PS053] J. Beatty and M. Alexander, “Games-Based Requirements Engineering Training: An Initial Experience Report”, IEEE International Requirements Engineering Conference, 2008
[PS054] J. Noll, A. Butterfield, K. Farrell, T. Mason, M. McGuire and R. McKinley, “GSD Sim: A Global Software Development Game”, IEEE International Conference on Global Software Engineeering Workshops, 2014
[PS055] H. Potter, M. Schots, L. Duboc and V. Werneck, “InspectorX: A game for software inspection training and learning”, IEEE Conference on Software Engineering Education and Training, 2014
[PS056] A. Rusu, R. Russell, R. Cocco and S. DiNicolantonio, “Introducing object oriented design patterns through a puzzle-based serious computer game”, Frontiers In Education Conference, 2011
[PS057] D. Carrington, A. Baker and A. van der Hoek, “It's All in the Game: Teaching Software Process Concepts”, Frontiers In Education Conference, 2005
[PS058] J. H. Andrews, “Killer App: A Eurogame about software quality”, IEEE Conference on Software Engineering Education and Training, 2013
[PS059] C. D. C.a De Oliveira, M. E.b Cintra and F. M.b Mendes Neto, “Learning risk management in software projects with a serious game based on intelligent agents and fuzzy systems”, Conference of the European Society for Fuzzy Logic and Technology, 2013
[PS060] S. T. Huang, M. C. Hsu and W. H. Lin, “Management and Education on the Case-Based Complex e-Business Systems Based On Agent Centric Ontology and Simulation Games”, IEEE International Conference on e-Business Engineering, 2007
[PS061] D. Saito, A. Takebayashi and T. Yamaura, “Minecraft-based preparatory training for software development project”, IEEE International Professional Communication Conference, 2015
[PS062] M. D. O.b Barros, A. R.a Dantas, G. O.a Veronese and C. M. L.a Werner, “Model-driven game development: Experience and model enhancements in software project management education”, Software Process Improvement and Practice, 2006
[PS063] E. Navarro and A. van der Hoek, “Multi-site Evaluation of SimSE”, ACM Technical Symposium on Computer Science Education, 2009
[PS064] M. R. Woodward and K. C. Mander, “On Software Engineering Education: Experiences with the Software Hut Game”, IEEE Transactions on Education, 1982
[PS065] G. Jimenez-Diaz, M. Gomez-Albarran and P. A. Gonzalez-Calero, “Pass the ball: Game-based learning of software design”, Lecture Notes in Computer Science, 2007
[PS066] M. J. Lee and A. J. Ko, “Personifying Programming Tool Feedback Improves Novice Programmers' Learning”, International Workshop on Computing Education Research, 2011
[PS067] N. Tillmann, J. De Halleux, T. Xie and J. Bishop, “Pex4Fun: A web-based environment for educational gaming via automated test generation”, IEEE/ACM International Conference on Automated Software Engineering, 2013
[PS068] J. M. Fernandes and S. M. Sousa, “PlayScrum - A card game to learn the scrum agile method”, International Conference on Games and Virtual Worlds for Serious Applications, 2010
[PS069] R. W. C. Lui, H. K. N. Leung, V. T. Y. Ng and P. T. Y. Lee, “PMS – A simulation game for interactive learning of software project management”, Communications in Computer and Information Science, 2015
[PS070] A. Baker, E. O. Navarroand A. van der Hoek, “Problems and Programmers: An Educational Software Engineering Card Game”, International Conference on Software Engineering, 2003
[PS071] J. E. N. Lino, M. A. Paludo, F. V. Binder, S. Reinehr and A. Malucelli, “Project management game 2D (PMG-2D): A serious game to assist software project managers training”, Frontiers In Education Conference, 2015
[PS072] M. A. Miljanovic and J. S. Bradbury, “Robot on!: A Serious Game for Improving Programming Comprehension”, International Workshop on Games and Software Engineering, 2016
[PS073] G. Jimenez-Diaz, M. Gomez-Albarran and P. A. Gonzalez-Calero, “Role-play virtual environments: Recreational learning of software design”, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008
[PS074] T. C. Kohwalter, E. W. G. Clua and L. G. P. Murta, “SDM - An Educational Game for Software Engineering”, Brazilian Symposium on Games and Digital Entertainment, 2011
[PS075] J. Ludewig, Th. Bassler, M. Deininger, K. Schneider and J. Schwille, “SESAM - Simulating software projects”, International Conference on Software Engineering and Knowledge Engineering, 1992
[PS076] C.a Caulfield, S P.a Maj, J. C.b Xia and D.a Veal, “Shall we play a game?”, Modern Applied Science, 2012
[PS077] E. O. Navarro and A. Van Der Hoek, “SimSE: An interactive simulation game for software engineering education”, International Conference on Computers and Advanced Technology in Education, 2004
[PS078] A. Rusu, R. Russell and R. Cocco, “Simulating the software engineering interview process using a decision-based serious

computer game", International Conference on Computer Games, 2011
[PS079] A. Jain and B. Boehm, "SimVBSE: Developing a Game for Value-Based Software Engineering", IEEE Conference on Software Engineering Education and Training, 2006
[PS080] H. Cervantes, S. Haziyev, O. Hrytsay and R. Kazman, "Smart Decisions: An Architectural Design Game", International Conference on Software Engineering, 2016
[PS081] M. Latzina and B. Rummel, "Soft(ware) skills in context: corporate usability training aiming at cross-disciplinary collaboration", IEEE Conference on Software Engineering Education and Training, 2003
[PS082] J. J. Horning and D. B. Wortman, "Software Hut: A Computer Program Engineering Project in the Form of a Game", IEEE Transactions on Software Engineering, 1977
[PS083] N. Tillmann, J. de Halleux, T. Xie, S. Gulwani and J. Bishop, "Teaching and learning programming and software engineering via interactive gaming", International Conference on Software Engineering, 2013
[PS084] C. Caulfield, D. Veal and S. P. Maj, "Teaching software engineering project management-A novel approach for software engineering programs", Modern Applied Science, 2011
[PS085] M.a Ivanovic, Z.a Putnik, Z.a Budimac and K.b Bothe, "Teaching Software Project Management course-Seven years experience", IEEE Global Engineering Education Conference, 2012
[PS086] M. Paasivaara, V. Heikkila, C. Lassenius and T. Toivola, "Teaching Students Scrum Using LEGO Blocks", International Conference on Software Engineering, 2014
[PS087] V. T. Heikkila, M; Paasivaara and C.Lassenius, "Teaching University Students Kanban with a Collaborative Board Game", International Conference on Software Engineering, 2016
[PS088] Kilamo, Terhi "The Community Game: Learning Open Source Development Through Participatory Exercise", International Academic MindTrek Conference: Envisioning Future Media Environments, 2010
[PS089] W.-F.a Chen, W.-H.b Wu, T.-Y.c Chuang and P.-N.d Chou, "The effect of varied game-based learning systems in engineering education: An experimental study", International Journal of Engineering Education, 2011
[PS090] M. D. Ernst and J. Chapin, "The Groupthink Specification Exercise", International Conference on Software Engineering, 2005
[PS091] A. Wegmann, "Theory and practice behind the course designing enterprisewide IT systems", IEEE Transactions on Education, 2004
[PS092] E. Oh and A. van der Hoek, "Towards game-based simulation as a method of teaching software engineering", Frontiers In Education Conference, 2002
[PS093] K.a Vega, H.a Fuks and G.b Carvalho, "Training in requirements by collaboration: Branching stories in second life", Simposio Brasileiro de Sistemas Colaborativos, 2009
[PS094] G. Taran, "Using Games in Software Engineering Education to Teach Risk Management", IEEE Conference on Software Engineering Education and Training, 2007
[PS095] G. Jiménez-Díaz, P. González-Calero and M. Gómez-Albarrán, "Using role-play virtual environments to learn software design", European Conference on Games-Based Learning, 2007
[PS096] A. Zeid, "Using simulation games to teach global software engineering courses", Frontiers In Education Conference, 2015
[PS097] W. Xu and S. Frezza, "A case study: Integrating a game application-driven approach and social collaborations into software engineering education", International Conference on Enterprise Information Systems, 2011
[PS098] I. A. Zualkernan, "A course for teaching integrated system design to computer engineering students", IEEE Global Engineering Education Conference, 2014
[PS099] N. Nitta, Y. Takemura and I. Kume, "A practice of collaborative project-based learning for mutual edification between programming skill and artistic craftsmanship", Frontiers In Education Conference, 2009
[PS100] A. I. Wang and B. Wu, "An application of a game development framework in higher education", International Journal of Computer Games Technology, 2009
[PS101] B. Wu, A. I. Wang, J.-E Strom and T. B. Kvamme, "An evaluation of using a Game Development Framework in higher education", IEEE Conference on Software Engineering Education and Training, 2009
[PS102] U. Wolz and S. M. Pulimood, "An integrated approach to project management through classic CS III and video game development", ACM Technical Symposium on Computer Science Education, 2007
[PS103] N. Ahmadi and M. Jazayeri, "Analyzing the Learning Process in Online Educational Game Design: A Case Study", Australian Software Engineering Conference, 2014
[PS104] T. Bay, M. Pedroni and B. Meyer, "By students, for students: A production-quality multimedia library and its application to game-based teaching", Journal of Object Technology, 2008
[PS105] B. Wu and A. I. Wang, "Comparison of learning software architecture by developing social applications versus games on the android platform", International Journal of Computer Games Technology, 2012
[PS106] T. Goulding and R. DiTrolio, "Complex Game Development by Freshman Computer Science Majors", ACM Technical Symposium on Computer Science Education, 2007
[PS107] T. Goulding, "Complex Game Development Throughout the College Curriculum", ACM Technical Symposium on Computer Science Education, 2008
[PS108] P. V. Gestwicki, "Computer Games As Motivation for Design Patterns", ACM Technical Symposium on Computer Science Education, 2007
[PS109] V. Isomöttönen and V. Lappalainen, "CS1 with games and an emphasis on TDD and unit testing: Piling a trend upon a trend", ACM Inroads, 2012
[PS110] C. S. Longstreet and K. Cooper, "Experience report: A sustainable serious educational game capstone project", International Conference on Computer Games, 2013
[PS111] S. Krusche, B. Reichart, P. Tolstoi and B. Bruegge, "Experiences from an experiential learning course on games development", ACM Technical Symposium on Computer Science Education, 2016
[PS112] E. Keenan and A. Steele, "Exploring Game Architecture Best-practices with Classic Space Invaders", International Conference on Software Engineering, 2011
[PS113] B. Wu, A. I. Wang, A. H. Ruud and W. Z. Zhang, "Extending Google Android's application as an educational tool", IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning, 2010
[PS114] A. I. Wang, "Extensive Evaluation of Using a Game Project in a Software Architecture Course", ACM Transactions on Computing Education, 2011
[PS115] H. Schoenau-Fog, L. Reng and L. B. Kofoed, "Fabrication of games and learning: A purposive game production", European Conference on Games-Based Learning, 2015
[PS116] E. Sweedyk and R. M. Keller, "Fun and Games: A New Software Engineering Course", ACM Technical Symposium on Computer Science Education, 2005
[PS117] B. Wu and A. I. Wang, "Game development frameworks for SE education", IEEE International Games Innovation Conference, 2011
[PS118] O.a Denninger and J.b Schimmel, "Game programming and

XNA in software engineering education", Computer Games and Allied Technology, 2008
[PS119] E. Sweedyk, "How middle school teachers solved our SE project problems", IEEE Conference on Software Engineering Education and Training, 2011
[PS120] S. Acharya and D. Burke, "Incorporating gaming in software engineering projects: Case of RMU monopoly", International Multi-Conference on Society, Cybernetics and Informatics, 2008
[PS121] R. Burns, L. Pollock and T. Harvey, "Integrating hard and soft skills: Software engineers serving middle school teachers", ACM Technical Symposium on Computer Science Education, 2012
[PS122] L. B. Sherrell and D. L. Mills, "Introducing Software Engineering Processes via Games and Simulations: A Tri-P-LETS Initiative", Journal of Computing Sciences in Colleges, 2008
[PS123] M. Yampolsky and W. Scacchi, "Learning Game Design and Software Engineering through a Game Prototyping Experience: A Case Study", International Conference on Software Engineering, 2016
[PS124] M. S. El-Nasr and B. K. Smith, "Learning through game modding", Computers in Entertainment, 2006
[PS125] D. Giordano and F. Maiorana, "Object Oriented Design through game development in XNA", Interdisciplinary Engineering Design Education Conference, 2013
[PS126] Z.a Li, L.b O'Brien, S.c Flint and R.c Sankaranarayana, "Object-oriented Sokoban solver: A serious game project for OOAD and AI education", Frontiers In Education Conference, 2015
[PS127] A. I. Wang, "Post-mortem analysis of student game projects in a software architecture course: Successes and challenges in student software architecture game projects", International IEEE Consumer Electronics Society's Games Innovations Conference, 2009
[PS128] T. H. Laine and E. Sutinen, "Refreshing contextualised IT curriculum with a pervasive game project in Tanzania", International Conference on Computing Education Research, 2011
[PS129] B.a b Maxim, "Serious games as software engineering capstone projects", ASEE Annual Conference and Exposition, 2008
[PS130] R. Dorner and U. Spierling, "Serious Games Development As a Vehicle for Teaching Entertainment Technology and Interdisciplinary Teamwork: Perspectives and Pitfalls", ACM International Workshop on Serious Games, 2014
[PS131] R. Kerbs, "Student teamwork: A capstone course in game programming", Frontiers In Education Conference, 2007
[PS132] T. M. Rao and S. Mitra, "Synergizing AI and OOSE: Enhancing interest in computer science through game-playing and puzzle-solving", AAAI Spring Symposium - Technical Report, 2008
[PS133] M. A. Gomez-Martin, G. Jimenez-Diaz and J. Arroyo, "Teaching design patterns using a family of games", Conference on Integrating Technology into Computer Science Education, 2009
[PS134] P. Gestwicki, F.-S Sun and B. Dean, "Teaching Game Design and Game Programming Through Interdisciplinary Courses", Journal of Computing Sciences in Colleges, 2008
[PS135] P. Gestwicki, "Teaching Game Programming with PlayN", Journal of Computing Sciences in Colleges, 2015
[PS136] J. E. Sims-Knight and R. L. Upchurch, "Teaching Object-Oriented Design Without Programming: A Progress Report", Computer Science Education, 1993
[PS137] H. Huni and I. Metz, "Teaching Object-oriented Software Architecture by Example: The Games Factory", ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, 1992
[PS138] J. Ryoo, F. Fonseca and D. S. Janzen, "Teaching Object-Oriented Software Engineering through Problem-Based Learning in the Context of Game Design", IEEE Conference on Software Engineering Education and Training, 2008
[PS139] A. Whitley, D. Dave, J. Fenwick, B. T. McDaniel, N. Radford and G. T. Loftis, "Teaching Software Design Principles to Undergraduates by Creating Software Inspired by the Board Game Castle Panic", Journal of Computing Sciences in Colleges, 2014
[PS140] N. E. Cagiltay, "Teaching software engineering by means of computer-game development: Challenges and opportunities", British Journal of Educational Technology, 2007
[PS141] M. C. Johnson and Y.-H. Lu, "Teaching software engineering through competition and collaboration", ASEE Annual Conference and Exposition, 2006
[PS142] K. Claypool and M. Claypool, "Teaching Software Engineering Through Game Design", ACM Technical Symposium on Computer Science Education, 2005
[PS143] T. W. S. Plum and G. M. Weinberg, "Teaching Structured Programming Attitudes, Even in APL, by Example", ACM Technical Symposium on Computer Science Education, 1974
[PS144] P. Gestwicki, "The Entity System Architecture and Its Application in an Undergraduate Game Development Studio", International Conference on the Foundations of Digital Games, 2012
[PS145] Y. Rankin, A. Gooch and B. Gooch, "The Impact of Game Design on Students' Interest in CS", International Conference on Game Development in Computer Science Education, 2008
[PS146] J. Pirker, A. Kultima and C. Gutl, "The value of game prototyping projects for students and industry", International Conference on Game Jams, Hackathons, and Game Creation Events, 2016
[PS147] I.a Yoon and E.-Y. E.b Kang, "Transforming experience of computer science software development through developing a usable multiplayer online game in one semester", International Conference on Computer Supported Education, 2014
[PS148] B. R. Maxim and B. Ridgway, "Use of interdisciplinary teams in game development", Frontiers In Education Conference, 2007
[PS149] J. C. McKim and H. J. C. Ellis, "Using a multiple term project to teach object oriented programming and design", IEEE Conference on Software Engineering Education and Training, 2004
[PS150] A. I. Wang and B. Wu, "Using game development to teach software architecture", International Journal of Computer Games Technology, 2011
[PS151] A. Emam and M. G. Mostafa, "Using game level design as an applied method for Software Engineering education", International Conference on Computer Games, 2012
[PS152] J. Edgington and S. Leutenegger, "Using the Ancient Game of Rogue in CS1", Journal of Computing Sciences in Colleges, 2008
[PS153] B. Wu, J. E. Strom, A. I. Wang and T. B. Kvamme, "XQUEST used in software architecture education", International IEEE Consumer Electronics Society's Games Innovations Conference, 2009
[PS154] S. Sheth, J. Bell and G. Kaiser, "A competitive-collaborative approach for introducing software engineering in a CS2 class", IEEE Conference on Software Engineering Education and Training, 2013
[PS155] A. Rusu, R. Russell, J. Robinson and A. Rusu, "Learning software engineering basic concepts using a five-phase game", Frontiers In Education Conference, 2010
[PS156] O. Shabalina, N. Sadovnikova and A. Kravets, "Methodology of teaching software engineering: Game-based learning cycle", IEEE Eastern European Regional Conference on the Engineering of Computer Based Systems, 2013

References

[1] IEEE & ACM JTFCC, Software engineering 2014: curriculum guidelines for undergraduate degree programs in software engineering, in: IEEE & ACM; The Joint Task Force on Computing Curricula, November 23, 2015.

[2] C.G.V. Wangenheim, F. Shull, To game or not to game? IEEE Softw. 26 (2) (2009) 92–94.

[3] M.R. Marques, A. Quispe, F.S. Ochoa, A systematic mapping study on practical approaches to teaching software engineering, Frontiers in Education Conference (FIE), 2014.

[4] B. Malik, S. Zafar, A Systematic Mapping Study on Software Engineering Education, in: World Academy of Science, Engineering and Technology, 2012.

[5] M.R.A. Souza, L.F. Veado, R. Moreira, E. Figueiredo, H. Costa, Games for learning: bridging game-related education methods to software engineering knowledge areas, International Conference on Software Engineering (ICSE) Software Engineering Education and Training Track (SEET), Buenos Aires, 2017.

[6] C. Caulfield, J.C. Xia, D. Veal, S.P. Maj, A systematic survey of games used for software engineering education, Mod. Appl. Sci. 5 (6) (2011) 28.

[7] M. Kosa, M. Yilmaz, R.V. O'Connor, P.M. Clarke, Software engineering education and games: a systematic literature review, J. Universal Comput. Sci. 22 (12) (2016) 1558–1574.

[8] A. Baker, E.O. Navarro, A. van der Hoek, Problems and Programmers: an educational software engineering card game, 25th International Conference on Software Engineering (ICSE), 2003, pp. 614–619.

[9] E. Navarro, A. van der Hoek, Multi-site evaluation of SimSE, in: ACM SIGCSE Bull. 41 (1) (2009).

[10] B. Wu, A.I. Wang, A guideline for game development-based learning: a literature review, Int. J. Comput. Games Technol. (January) (2012).

[11] S. Krusche, B. Reichart, P. Tolstoi, B. Bruegge, Experiences from an experiential learning course on games development, ACM Technical Symposium on Computing Science Education (SIGCSE), 2016.

[12] S. Deterding, D. Dixon, Gamification : using game design elements in non-gaming contexts, Extended Abstracts on Human Factors in Computing Systems (CHI), 2011.

[13] O. Pedreira, F. García, N. Brisaboa, M. Piattini, Gamification in software engineering – a systematic mapping, Inf. Softw. Technol. 57 (2015) 157–168.

[14] D.J. Dubois, G. Tamburrell, Understanding gamification mechanisms for software development, Joint Meeting on Foundations of Software Engineering, Saint Petersburg, ESEC/FSE'13, 2013.

[15] E.B. Passos, D.B. Medeiros, P.A. Neto, E.W. Clua, Turning real-world software development into a game, Brazilian Symposium on Games and Digital Entertainment (SBGAMES), 2011.

[16] E. Herranz, R.C. Palacios, A. de Amescua Seco, M. Yilmaz, Gamification as a disruptive factor in software process improvement initiatives, J. Universal Comput. Sci. 20 (2014) 885–906.

[17] B.S. Akpolat, W. Slany, Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification, IEEE Conference on Software Engineering Education and Training, CSEE & T, 2014.

[18] V. Uskov, B. Sekar, Gamification of software engineering curriculum, Frontiers in Education Conference (FIE), 2014.

[19] M.R.A. Souza, K. Constantino, L. Veado, E. Figueiredo, Gamification in software engineering education: an empirical study, IEEE 30th Conference on Software Engineering Education & Training (CSEE & T), 2017.

[20] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2007.

[21] H. Cervantes, S. Haziyev, O. Hrytsay, R. Kazman, Smart decisions: an architectural design game, Proceedings of the 38th International Conference on Software Engineering Companion, 2016.

[22] N. Tillmann, J. De Halleux, T. Xie, J. Bishop, Constructing coding duels in Pex4Fun and code hunt, Proceedings of the 2014 International Symposium on Software Testing and Analysis (ISSTA), 2014.

[23] B. Wu, A.I. Wang, J.E. Strøm, T.B. Kvamme, An evaluation of using a game development framework in higher education, 22nd Conference on Software Engineering Education and Training (CSEET), 2009.

[24] A.I. Wang, B. Wu, An application of a game development framework in higher education, Int. J. Comput. Games Technol. 2009 (2009) 1–12.

[25] B. Wu, A.I. Wang, A.H. Ruud, W.Z. Zhang, Extending Google android's application as an educational tool, Third IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), 2010.

[26] O. Denninger, J. Schimmel, Game programming and XNA in software engineering education, Proceedings of Computer Games and Allied Technology (CGAT), 2008.

[27] B. Wu, A.I. Wang, Game development frameworks for SE education, IEEE International Games Innovation Conference (IGIC), 2011.

[28] D. Dicheva, C. Dichev, G. Agre, G. Angelova, Gamification in Education: A Systematic Mapping Study, Journal of Educational Technology & Society 18 (3) (2015) 75–88.

[29] G. Zichermann, C. Cunningham, Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps, O'Reilly Media, Inc., 2011.

[30] A. Baker, E.O. Navarro, A. Van Der Hoek, An experimental card game for teaching software engineering processes, J. Syst. Softw. 75 (2005) 3–16.

[31] C. Caulfield, D. Veal, S.P. Maj, Teaching software engineering project management–a novel approach for software engineering programs, Mod. Appl. Sci. 5 (5) (2011) 87–104.

[32] V.T. Heikkilä, M. Paasivaara, C. Lassenius, Teaching university students Kanban with a collaborative board game, Proceedings of the 38th International Conference on Software Engineering Companion, 2016.

[33] D.C.C. Peixoto, R.F. Resende, C.I.P.S. Pádua, Evaluating software engineering simulation games: the UGALCO framework, IEEE Frontiers in Education Conference (FIE), 2014.

[34] N. Andriano, M.G. Moyano, C. Bertoni, D. Rubio, A quantitative assessment method for simulation-based e-learnings, IEEE-CS Conference on Software Engineering Education and Training (CSEE & T), 2011.

[35] E.O. Navarro, A. Van Der Hoek, Comprehensive evaluation of an educational software engineering simulation environment, Conference on Software Engineering Education & Training (CSEE & T), 2009.

[36] R. Savi, A.F. Borgatto, C.G. von Wangenheim, A model for the evaluation of educational games for teaching software engineering, 25th Brazilian Symposium on Software Engineering (SBES), 2011.

[37] C. Oliveira, M. Cintra, F. Mendes Neto, Learning risk management in software projects with a serious game based on intelligent agents and fuzzy systems, 8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT), 2013.

[38] K. Werbach, D. Hunter, For the Win: How Game Thinking Can Revolutionize Your Business, Wharton Digital Press, 2012.

[39] C. Wohlin, M. Host, P. Runeson, M. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering, Springer Science & Business Media, 2012.

[40] P. Baristela, C.V. Wangenheim, Games for Teaching Computing in Higher Education–A Systematic Review, IEEE Technology and Engineering Education 9 (1) (March 2016) 8–30.

[41] G. Petri, C.G. von Wangenheim, How Games for Computing Education are Evaluated? A Systematic Literature Review, Computers & Education 107 (January 2017) 68–90.