Games for Learning

Bridging Game-related Education Methods to Software Engineering Knowledge Areas

Mauricio R A Souza[1], Lucas Furtini[1], Renata Teles Moreira[2], Eduardo Figueiredo[1], Heitor Costa[2]

[1] Computer Science Department, Federal University of Minas Gerais, Belo Horizonte, Brazil
[2] Computer Science Department, Federal University of Lavras, Belo Horizonte, Brazil

{mrasouza,lfurtini,emagno}@dcc.ufmg.br, {renata, heitor}@dcc.ufla.br

***Abstract*— 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. 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. The systematic mapping study identified 106 primary studies describing the use of serious games, gamification and game development in software engineering education. Based on this mapping, we aimed to track the learning goals of each primary study to the knowledge areas defined in ACM/IEEE curricular recommendations. As a result, we observed that "Software Process", "Software Design" and "Profession Practice" are the most recurring knowledge areas explored by game related approaches in software engineering education. We also uncover possible research opportunities for game-related education methods.**

Keywords-software engineering education; game-based learning; gamification; game development based learning.

I. INTRODUCTION

The term "software engineering" has been used since the 1960s and, as a discipline, it has evolved over these decades in reflection of the evolution of the technology and the complexity of the problems to be solved by computers, requiring more complex software solutions. Therefore, the challenges of the educating new software engineers is challengehas also evolved: since software engineering is more than just programming, it includes attention to details such as quality, schedule, and economic goals (SEEK, 2014).

An initial challenge in SEET arises from the dual nature of the software engineering discipline, which study and practice is influenced both by its roots in computer science and its emergence as an engineering discipline (SEEK, 2014). This characteristic has a direct influence in the amount of material instructors must cover in software engineering classrooms, and the fact that Software professionals are required to not only understand technical challenges but also be up-to-speed on nontechnical issues, including management, communication, and teamwork (Wangenheim and Shull, 2009).

Additionally, SEET should include student experiences with the professional practice of software engineering in such way that they can understand which practices and techniques are useful in various situations [1]. The nature of the assignments and projects proposed in the academy are limited in scope and time, increasing the challenge finding a good balance between theory and practice. Wangenheim and Shull [1] also warn about losing the ability to learn by failure—that is, the important educational insight that comes from applying a suboptimal approach.ame-related educational methods have been used to minimize this gap between theory and practice. For instance, serious games have been recurrent tools for improving learning process and student engagement in software engineering [3] [4] [5]. In serious games, the main purpose of the class is to learn while students are playing a game. In addition to the use of serious games, game-related educational methods can also include gamification [6] [7] and game development based learning [8] [9]. Gamification is the adoption of games elements and mechanics in the learning process. Unlike serious game, gamification is not a game, but it includes elements of games, such as leaderboard and badges [10]. In the case of game development based learning, the class assignment or project is the developing of games as a domain. Therefore, students experience software engineering methods and techniques in game developing.

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

In this context, the goal of this paper is to identify and discuss game related methods in the light of the nine knowledge areas defined in ACM/IEEE Curriculum Guidelines. First, we aim to investigate which specific knowledge areas these types of educational methods cover. In this case, a systematic mapping study was applied to survey the related literature and the game related educational methods found were categorized regarding their nature and their support for knowledge areas. Another expected contribution of this study is to provide a comparative analysis and discussion of the different game related methods. In particular, we provide an analysis of how these education methods support software engineering education focusing on the knowledge areas.

The remainder of this paper is organized as follows: Section II provides the necessary theoretical foundation for the study; in Section III we describe the study planning; in Section IV we present the results of the study; in Section V we discuss the threats to the validity of the study; in Section VI we discuss the related work; in Section VII we present the final discussion and future work.

II. BACKGROUND

This section defines important concepts used in this study.

A. 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 about what every software engineering graduate must know: The Software Engineering Education Knowledge.

SE 2014 is organized in "knowledge areas", representing particular sub disciplines of software engineering that are generally recognized as a significant part of the software engineering knowledge that an undergraduate should know. The se 2014 knowledge areas are [1]: "Software Modeling and Analysis", "Requirements Analysis and Specification", "Software Design", "Software Verification and Validation", "Software Process", "Software Quality", "Professional Practice", "Computing Essentials", "Mathematical and Engineering Fundamentals", and "Security". The knowledge areas considered in this study are described in Table 1.

In this paper, we use SE 2014 knowledge areas as reference for identifying what software engineering disciplines are supported by game related approaches. Although we acknowledge the relevance of the knowledge areas "Computing Essentials", and "Security", they were not considered 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.

B. Game Related Approaches

In context of this study, the term "game related approaches" refers to any approach that uses games or game elements for supporting learning and teaching in the scope of software engineering education. Therefore, it is not limited only to (serious) games used for the purpose of teaching or training specific knowledge and abilities in software engineering: It also includes the use of game elements and mechanics and the adoption of the domain of game development as learning instruments.

Game Based Learning (GBL, henceforth) deals with game applications that have defined learning outcomes [2]. Games are "any contest (play) among adversaries (players) operating under constraints (rules) for an objective (winning, victory, or pay-off)". Therefore, 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 do not limit this concept to digital games, or to the use of serious games: commercial games (games with the pure intent of entertainment) used in the context of learning software engineering are also considered.

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 [10]. The goal of gamification is to

TABLE 1 KNOWLEDGE AREAS FROM SE2014 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.

use the philosophy, elements, and mechanics 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 [11].

Games can also be used as an engaging domain for learning software engineering. Game Development Based Learning (GDBL, henceforth) is the practice of developing games as a hands-on learning experience [12]. First, games are a well-known domain for students, therefore facilitating the understanding of requirements. Second, games are appealing, motivating and engaging students in their authoring process. Finally, the varied complexity of games can provide educators with a large set of options for classroom assignments and projects.

III. Study Design

This section presents the goals of this study and the methodology adopted.

A. Goal

The goal of this study is to: Identify and analyze *game related approaches* from the purpose of *mapping their learning objectives to software engineering knowledge areas* from the perspective of *researchers, educators, and learners* in the context of *software engineering education*.

B. Method

To achieve this goal, a systematic mapping study was conducted. A systematic mapping study (SMS, henceforth) is a secondary study method that systematically (i.e. based on a structured and repeatable process or protocol) explores and categorize studies in a given research field, and provides a structure of the type of research reports and results that have been published [13]. 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.

C. Systematic Mapping Process

We have conducted the SMS in the period of June/2016 to September/2016, following four process steps adapted from [13] (see Figure 1): definition of research questions, conducting the search for relevant papers, study selection, and classification and data extraction. Each process steps have an outcome, converging for the systematic map as the final outcome of the process. 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.

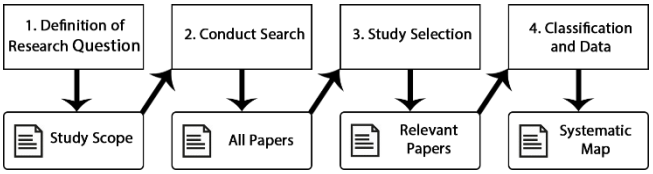

FIGURE 1. SYSTEMATIC MAPPING PROCESS, ADAPTED FROM PETTERSEN ET AL. [13]

D. Research Questions

In order to achieve the goal of this study, we established two research questions:

RQ1. What game related approaches have been proposed for supporting software engineering education? – The expected outcome is a categorized list of approaches.

RQ2. What software engineering education knowledge areas are supported by the existing game related approaches? – The expected outcome is a mapping between learning objectives from the approaches identified from RQ1 and the software engineering knowledge areas from IEEE/ACM curricular recommendations.

E. 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: education, software engineering, 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 requirement, requirement 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 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 selected scientific databases, using (and adapting when necessary) the search string. The databases used were ACM Digital Library[1] and IEEE Xplore[2], considering the number of relevant conferences and journals indexed by them.

[1] http://portal.acm.org/dl.cfm
[2] http://ieeexplore.ieee.org

F. Study Selection

In this step, we filtered the studies retrieved from the automatic searches through a series of 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 the proposal, usage, discussion or evaluation of game related approaches in the context of software engineering education.
- Exclusion Criteria: Papers not written in English; Papers not fully available download; Studies formatted as short papers (2 or less pages), workshops, tutorials, demos, books, book chapters, and similar; secondary studies; and duplicates.

It is important to highlight that we are not interested in studies focusing on general approaches for software engineering education that do not explicitly advocate or discuss the use of game related approaches. We are also not interested in game related approaches for education that are not focused specifically in the context of software engineering education, or that simply provides an arbitrary example of application in the context of software engineering. Finally, studies focusing on software engineering and games but out of the scope of education and training were not included. The study selection process is described in Table 2.

TABLE 2. STUDY SELECTION PROCESS

Study Selection Steps	*ACM DL*	*IEEE Xplore*
Step 1: Retrieval and organization of studies from automatic search databases	1204	558
Step 2: Remaining studies after reading titles and abstracts and excluding studies not in compliance to Inclusion Criteria	248	155
Step 3: Remaining studies after downloading papers, and reading introduction and conclusion for applying Exclusion Criteria	42	69
Total Primary Studies Selected (no duplicates)	106	

Two researchers performed the first step, seeking consensus. In case of doubt or divergence in opinion regarding the inclusion of a given study, the study was kept for the next step. In the second step divergences or doubts about the exclusion of a study was resolved with a voting with the other three researchers. In case of different papers reporting the same study (e.g., journal and conference papers with the same title), only the most recent and/or most complete was kept in the final list of primary studies.

G. Study Classification and Data Extraction

The resulting list of primary studies was analyzed in order to extract information identifying: the presented approach, the type of approach, and its learning objectives or expected developed skills.

We classified each game related approach in the following categories:

- Gamification: Approaches using gamification strategies to support software engineering education.
- GBL: Approaches using proper games as instruments to support software engineering education.
- GDBL: Approaches where the domain of game development is used as scenario for supporting software engineering education.

To propose a classification scheme for the learning goals, we identified the expected learning outcomes and mapped them to the purpose and related topics of the knowledge areas from ACM/IEEE curricular recommendations for software engineering.

IV. RESULTS

The selection of primary studies resulted in 106 primary studies. Table 3 summarizes the must recurring publication venues in our study and the publication venues respective counting of selected primary studies. We listed the publication venues that have three or more primary studies selected in this study.

TABLE 3 RELEVANT PUBLICATION VENUES FOR GAME RELATED APPROACHES IN SOFTWARE ENGINEERING EDUCATION

# Studies	*Publication Venues*
14	IEEE Conference on Software Engineering Education and Training (CSEE&T)
12	Frontiers In Education Conference (FIE)
9	International Conference on Software Engineering (ICSE)
7	ACM Technical Symposium on Computer Science Education (SIGCSE)
5	Journal of Computing Sciences in Colleges
4	IEEE Transactions on Education
3	International Conference on Computer Games (CGAMES)
3	IEEE Global Engineering Education Conference (EDUCON)
3	International Workshop on Games and Software Engineering (GAS)

Figure 2**Erro! Fonte de referência não encontrada.** presents a histogram of papers from 1974 to 2016, with different colors representing the types of approaches found (discussed in details in Subsection A). This result suggests that software engineering education has been a challenge since the early years of software engineering in the 1970's, and that the use of game related approaches is not a novelty. Nonetheless, considering only the two databases searched, since 2002 there is a constant and growing presence of studies about game related approaches for software engineering education.

The complete list of primary studies can be found in http://goo.gl/9jQq9Q. The remaining of the study cites primary studies using the id "PSxxx".

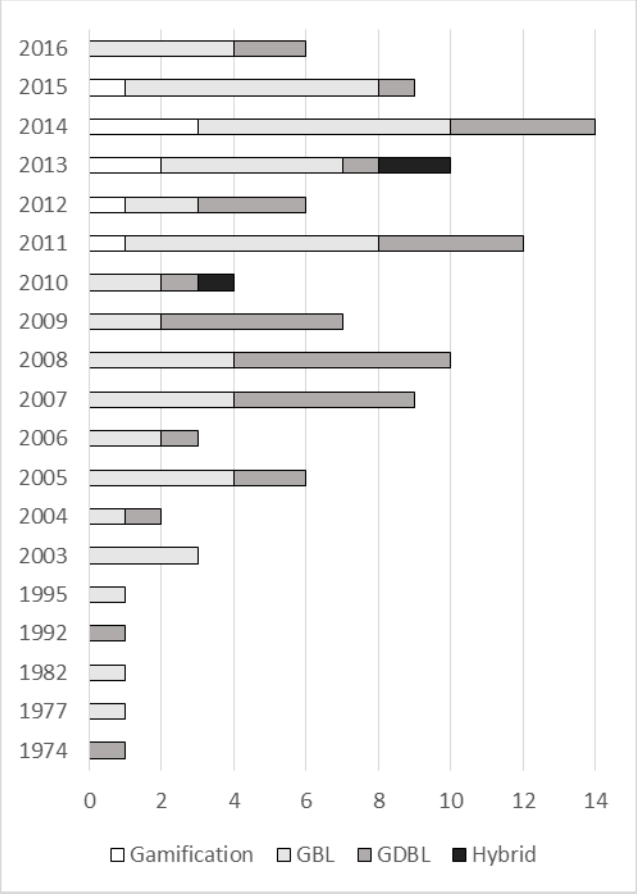

FIGURE 2 TIMELINE OF PRIMARY STUDIES

A. RQ1 Game related approaches

In this subsection we discuss the results for the research question "*RQ1. What game related approaches have been proposed for supporting software engineering education?"*.

A total of 57 primary studies presented GBL approaches (for a total of 48 different games), 38 presented GDBL approaches, 8 presented Gamification approaches, and 3 presented hybrid approaches. The mapping of primary studies and respective approach type is presented in Table 4.

TABLE 4. MAPPING OF CATEGORIES OF GAME RELATED APPROACHES A AND PRIMARY STUDIES

Approaches	Primary Studies
Gamification	PS001 – PS008
GBL	PS009 – PS065
GDBL	PS066 – PS103
Hybrid	PS104 – PS106

Although the term "Gamification" dates from 2011 in the academic scenario [10], the adoption of this approach in software engineering education has been reported since 2011 (see **Erro! Fonte de referência não encontrada.**). The gamification approaches found are usually applied as an innovative learning methodology more focused in engaging and motivating learners. However, gamification approaches are usually not associated to a specific learning goal or to the development of a specific software engineering skill. Nonetheless, we found approaches that use gamification to promote specific skills: Bell *et al* [14] (*PS008*) introduce software testing in the form of "quests", and Singer and Schneider [7] (*PS006*) use game elements to promote a competition in order to induce students to use good practices of version control. There is still an open opportunity to apply and evaluate gamification techniques in software engineering education to promote good practices.GBL and GDBL were the most recurring approaches found in this study. The GBL approaches found are varied in genres and formats. We found digital games such as simulations, puzzle-based games, and commercial games. In equal proportion, we found non-digital games such as card-games, board games, role-playing games (RPG), and group activities. SimSE [15] (*PS023, PS044, PS065*), Problems and Programmers [5] (*PS016, PS040, PS049*), and Pex4Fun [4] (*PS025, PS047, PS058*) were the most recurring GBL approaches found.

GDBL approaches are divided in two types: assignments on developing game products and the adoption of game development frameworks. The first rely on hands on experience in the domain of game development to introduce students the challenges of software engineering. The second advocates the use of game development frameworks as learning instruments in order to simplify certain aspects of development and/or allowing students to concentrate effort on specific learning goals.

Some studies (*PS104, PS105, PS106*) mix the previous approaches in order to achieve better learning results (hybrid approaches). Two studies (*PS105, PS106*) discuss the advantages of learning cycles where students play and develop serious games. One study (*PS104)* propose a hybrid approach consisting of developing games focusing on good design, and applying a gamified strategy for testing it.

B. RQ2 Mapping learning goals to SEEK Knowledge Areas

In this subsection we discuss the results for the research question "*RQ2. What software engineering education knowledge areas are supported by the existing game related approaches?".*

We identified and tracked the learning goals of each primary study to the SE 2014 knowledge. In the case of recurring games, such as Software Hut (*PS045, PS059*), SimSE [15] (*PS023, PS044, PS065*), Problems and Programmers [5] (*PS016, PS040, PS049*), and Pex4Fun [4] (*PS025, PS047, PS058*) we mapped the learning goals of only one primary study for each of them. This mapping is presented in Table 5, and the studies with repeated games were marked with "*". Figure 3 summarizes this mapping, presenting a quantitative relationship between different game approaches and the coverage of each SE 2014 knowledge area.

The results show that "Software Process" is the knowledge area with the greater number of game related approaches supporting it (54), followed by "Software Design" (42), "Professional Practice" (42). "Requirement Analysis and Specification" (28) and "Software Verification and Validation" (23). The least supported knowledge areas are

"Software Quality" and "Software Modelling and Analysis", supported by 6 and 8 game related approaches, respectively.

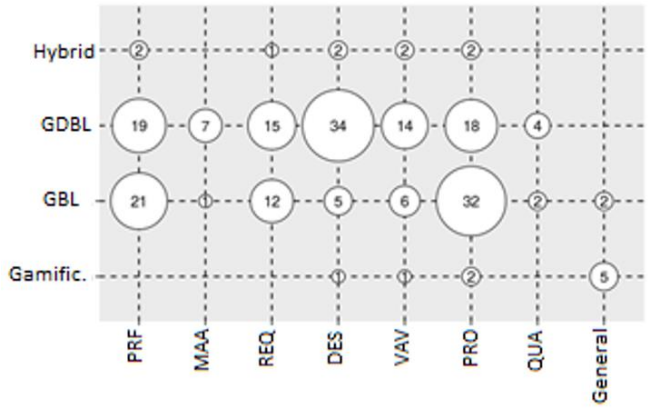

FIGURE 3. COVERAGE OF SOFTWARE ENGINEERING KNOWLEDGE AREAS

TABLE 5 MAPPING OF KNOWLEDGE AREAS (K.A.) FROM SE 2014 AND PRIMARY STUDIES

K.A.	Primary Studies
PRF	*PS009, PS010, PS011, PS012, PS013, PS014, PS017, PS021, PS029, PS030, PS031, PS034, PS035, PS042, PS043, PS045, PS051, PS056, PS057* PS059, PS062, PS063, PS067, PS069, PS071, PS072, PS074, PS075, PS081, PS082, PS085, PS087, PS090, PS091, PS094, PS096, PS097, PS098, PS099, PS101, PS102, PS105, PS106.*
MAA	*PS042, PS079, PS088, PS090, PS093, PS095, PS099, PS102.*
REQ	*PS010, PS011, PS012, PS013, PS029, PS032, PS035, PS036, PS042, PS053, PS059, PS062, PS068, PS072, PS074, PS078, PS079, PS081, PS082, PS083, PS084, PS085, PS087, PS090, PS095, PS098, PS099, PS105.*
DES	*PS005, PS012, PS032, PS039, PS055, PS056, PS066, PS067, PS068, PS070, PS072, PS073, PS074, PS075, PS076, PS077, PS078, PS079, PS080, PS081, PS082, PS083, PS084, PS085, PS086, PS087, PS089, PS090, PS091, PS092, PS093, PS094, PS095, PS096, PS097, PS098, PS099, PS100, PS102, PS103, PS104, PS105.*
VAV	*PS008, PS012, PS022, PS025, PS032, PS038, PS046, PS047*, PS058*, PS072, PS074, PS078, PS079, PS081, PS082, PS083, PS084, PS090, PS095, PS097, PS098, PS099, PS102, PS104, PS105.*
PRO	*PS001, PS006, PS009, PS011, PS012, PS013, PS015, PS016*, PS017, PS018, PS019, PS020, PS023, PS024, PS026, PS028, PS029, PS030, PS031, PS033, PS034, PS037, PS040, PS041, PS043, PS044*, PS045, PS048, PS049*, PS050, PS052, PS054, PS056, PS057*, PS059, PS060, PS061, PS064, PS065*, PS069, PS072, PS074, PS075, PS079, PS081, PS082, PS083, PS085, PS087, PS088, PS090, PS091, PS095, PS097, PS098, PS099, PS101, PS105, PS106.*
QUA	*PS011, PS041, PS074, PS079, PS095, PS099.*

Some approaches did not address any knowledge area specifically, we classified those as "general", since they were still related to software engineering education, but focused on broader concepts. It is important to notice that a single approach can support any number of knowledge areas.

We noticed that the two most recurring knowledge areas, "Software Process" and "Software Design", are each predominantly supported by a specific type of approach. GBL is predominant for "Software Process". On the other hand, GDBL is predominant for "Software Design". The remaining knowledge areas are almost equally supported by GBL and GDBL. In Section V we provide an in-depth analysis of how the knowledge areas are supported by each approach category.

V. DISCUSSION

In this section we discuss how the results presented in previous section overlap. We observed that GDBL and GBL have distinct characteristics that makes each one more adequate to a specific context or objective. The most significant difference identified is that GBL are usually constrained in a shorter duration and usually provide simplifications that encourage the perception of specific skills or details in a given topic. This characteristic makes GBL more suitable for specific learning goals and for giving a deeper understanding of specific topics. For instance, most of the GBL approaches supports one or two knowledge areas. Examples include games focusing specifically on software process modelling, risk management, teaching scrum, or interview process of requirement engineering.

On the other hand, GDBL takes advantage of problem based learning, presenting learners with challenges where they usually have to apply software engineering techniques. By applying software engineering best practices, learners discover the inherent problems and challenges of the discipline by themselves. Therefore, GDBL is more suitable for extensive learning goals and for developing skills by practicing. Several of the GDBL approaches cover three or more knowledge areas, because they introduce students to various aspects of software development life cycle in one assignment.

Gamification plays a different role in the study sample we analyzed. The studies found can be split in two types: the first one focused in reimagining the classroom or lesson layout, where game elements promote engagement and fun to keep students motivated to learn The second focus on gamifying the learning topic itself, inducing students to apply or develop new skills to achieve better performance at the meta-game or to move the narrative forward, therefore promoting a change in behaviors. The first type was more recurrent in our sample, while the second is still a good opportunity of research on software engineering education.

In the following subsections we discuss how game related approaches support each knowledge area individually.

A. *"Software Process"*

For this area there is a predominance of GBL approaches (such as simulations, board and card games, and role-playing games). We observed that authors often discuss the difficulty to provide students with scenarios that really justify the need of software process and management good practices using the classic classroom format and assignments. However, there is a relevant number of GDBL approaches supporting this knowledge area, which usually focus on presenting learners to

a development life cycle or a predefined process model they have to follow in order to develop a game product.

Software life cycles and process models are the most recurring topic related to this knowledge area, it is largely supported by GBL (*PS009, PS028 , PS044, PS049, PS013, PS016, PS017, PS023, PS024, PS029, PS040, PS041, PS012*), GDBL approaches (*PS072, PS083, PS090, PS069, PS079, PS082, PS085, PS095, PS087, PS098, PS074, PS101, PS099, PS088*), including hybrid approaches using both strategies (*PS106, PS105*) and by gamification (PS001).

SimSE [15] (*PS023, PS044, PS065*), for instance, is an educational, interactive, graphical environment for building and simulating software engineering processes in a game-like setting. SimSE has been also used as base for other process simulation games, such as the approach proposed by Wang et al [16] (*PS013*), MO-SEProcess [17] (*PS029*) (a multiplayer take on SimSE), and SimVBSE [18] (*PS054*) (an adaptation of SimSE for value-based software engineering). Additionally, KillerApp [19] (*PS041*) and Problems and Programmers [5] (*PS016, PS040, PS049*) are two examples of non-digital games that also explore software life cycle topic.

Related topics also includes user centered development process (*PS056*), open source software development (*PS061*), technical debt (*PS020*), and software process modelling (*PS033*).

Software project management is another recurring key topic in PRO knowledge area, supported by GBL (*PS028, PS030, PS026, PS050, PS011, PS019, PS018, PS064, PS037, PS031, PS065*), GDBL (*PS069, PS079, PS085, PS087, PS044, PS099, PS091, PS072, PS081*) and by one hybrid approach (*PS106*).

For instance, GameDev Tycoon [20] (*PS030*), SimJavaSP [21] (*PS028*), PMG-2D [22] (*PS050*), and SDM [23] (*PS052*) are four simulation games that explore various aspects of management of software development projects.

In smaller scale, we have also found approaches covering agile development methods (*PS090, PS097, PS048, PS099, PS015, PS034, PS060*), configuration management and version control (*PS075, PS082, PS006*), and value based software engineering (*PS054*).

B. "Software Design"

GDBL are predominant for supporting the area. Authors noted better learning results on teaching software engineering techniques regarding the good design of software (e.g. design patterns and software, architecture, usability) using hands-on and problem based approaches, hence the use of GDBL. Recurring topics found include: design patterns, software architecture, and object oriented design.

The ability to apply those principles are usually explored in the context of designing or modifying a real software (or prototype). The advantage of adopting the domain of games in these assignments is that they are familiar to students. For instance, Keenan and Steele [9] (*PS076*) promote the classic arcade game Space Invaders, it provides an ideal environment for students to learn about best practices in game software, highlighting the experiment with the challenges and tradeoffs inherent in any software design. Yampolsky & Scacchi [8] (*PS084*) describe the experience that students with no prior experience in software development or programming must take on the task of learning how to make a game, and along the way, learn about many common challenges in modern SE practice through personal discovery and experience. The authors suggest that this project may be used for lowering the barriers to entry into game software development in specific, and into SE more generally.

Another innovative application of GDBL is the use of game development frameworks (toolkits used to develop games) to develop or modify games for learning software engineering skills. XNA, for instance, is a game development framework adopted in the approaches of three primary studies with focus on supporting the education of software architecture (*PS068, PS080, PS103*). In this scenario, learners can focus on software engineering practices and concepts rather than coding, therefore lowering technical barriers for SE education.

C. "Professional Practice"

The area skills are usually treated as secondary goals, i.e. usually the approaches have a different learning goal and passively develops skills such as communication, teamwork, collaboration, and other soft skills along the learning process. For instance, "Extreme Construction" [24] (*PS034*) is a game activity that teaches the fundamentals of any agile methodology while helping develop teamwork skills.

In all GDBL approaches that support this knowledge area, students working in teams had to develop team management, coordination, communication, collaboration, interpretation and other soft skills for achieving success in their assignments.

In GBL approaches there are some games that communication and teamwork skills are the core of the gameplay. For instance, in "Groupthink" [3] (*PS062*) the goal is each team improve an incomplete specification, agree upon the specification, and understand what they have agreed upon. The teams are asked questions that each member has to answer individually, and for each divergent answer, the team score is penalized. Alsaedi et al. [25] (*PS021*) adopted a research training game named *TeC*, developed to improve team coordination in disaster response teams, and hypothesized that it might help student software teams in developing team coordination and communication.

D. "Requirement Analysis and Specification"

All GDBL that support this area promote the hands-on practice on the specification of game products. Similarly, some GBL use role-playing games develop requirement analysis and specification skills by asking learners to define the requirements for specific products (*PS012, PS011, PS042*).

Among the GBL approaches we found serious games created specifically to develop requirement engineering skills. The "Software Quantum Metaphor" [26] (*PS010*) is a simulated software project game that help students visualize a main challenge of RE: building the right system within available time. "Groupthink" [3] (*PS062*) is a fun group activity, in the style of a game show, that teaches students about specifications (the difficulty of writing them, techniques for getting them right, and criteria for evaluating them). Later,

Ye et al. [17] (*PS029*) adapted Groupthink as a game in Second Life. RE-O-POLY [27] (*PS035*) is a board game similar to "Monopoly" developed to introduce and reinforce a fundamental set of established requirement engineering for good practices. Finally, Rusu et al. [28] (*PS053*) simulates the interview process of requirement analysis using a decision-based serious computer game.

No gamification approach was found for supporting this knowledge area.

E. "Software Verification and Validation"

Similar to the previous knowledge area, all GDBL approaches found promotes the hands-on practice for verification and validation on the games developed in student's projects.

Pex4Fun [4] (*PS025, PS047, PS058*) is the most recurring serious game in the area. The game (and its successor Code Hunt [29] (*PS022*)) is based on challenges where students implement a code segment based only on unit testing results. Therefore, the students have to continuously debug their code in order to complete the challenges. InspectorX [30] (*PS038*) is a serious game focused on develop software inspection abilities.

One gamification approach was found. It focus on software testing: HALO (Highly Addictive, sociaLly Optimized) [14] (*PS008*) uses quests and storyline to provide software engineering tasks to students. Bell et al [14] used HALO for making testing more attractive to students (leading to fewer bugs in their code) and also lowering technical barriers of testing for new programmers.

F. "Software Quality" and "Software Modelling and Analysis"

No game was found to specifically address the knowledge area of "Software Quality". Even with the vast popularity of quality models, only one of the games found (PS011) adopted assignments in the form of a roleplaying game, in which in a given phase students had to develop a quality plan for the project, starting from a quality standard in accordance to the principles of IS09000. Other approaches would impart students with the perception of quality criteria for products.

Similarly, "Software Modelling and Analysis" was approached directly by the studies. Usually this knowledge area was covered by introducing learners to UML modelling language for documenting various aspects of the software applications they were working on.

These two knowledge areas are still interesting research objects in the context of game related approaches for software engineering education.

VI. THREATS TO VALIDITY

In this section we leverage some threats to the validity of this study, as presented next.

Construct validity: to minimize the risk that study was not able to address the research questions, a pilot study was performed, and the research questions and search strategy were carefully refined in order to achieve the research goals.

Internal validity: during the extraction process, the studies were classified based on our judgment. However, despite double checking, some studies could have been classified incorrectly, especially regarding the mapping of the knowledge areas from SE 2014. Sometimes it was difficult to classify the studies according to research areas and topics. In order to mitigate this threat, the classification process was repeated by each author of this study and the results were jointly discussed to reach a consensus.

External validity: it is possible that the systematic mapping study did not return all relevant studies on game related approaches for software engineering education. We adopted international standards (such as SE 2014) and compared the keyword to identify only the relevant for the search procedure. Additionally, we compared our search process and results with other secondary studies to assure the relevance of our primary studies.

Conclusion: to ensure conclusion validity of our study, we presented throughout Sections IV and V graphs and tables exposing results generated directly from the data, and discussed the explicit observations and trends. This ensures a high degree of traceability between the data and conclusions. In addition, our corpus of studies is available for other researchers.

VII. RELATED WORK

This section presents studies that provide similar and complementary objectives and results to the ones that we. The related work is concerned with methods for software engineering education, the adoption of game related approaches for software engineering, and the support to software engineering knowledge areas.

Concerning teaching methods for software engineering education, Marques et al. [31] present a systematic mapping study that characterizes the approaches available to address the practical experiences in software engineering education. The authors mapped 173 primary studies. The study shows that games approach was used in 12 studies (7%), and simulation in 9 studies (5%). Learning by doing approach is the most used, with 93 studies (54%). The results indicate that universities have realized the value of including practical experiences as part of the software engineering teaching process. There is a clear concern for teaching software engineering involving practical experiences, and there are several initiatives exploring how to do sit. However, few proposals indicate how to address this challenge. In comparison, our results corroborated the perspective that game related approaches also impart practical experiences in software engineering education. Our study discussed that a key characteristic of Game Development Based Learning approaches is the focus on hands-on experience in building application in the domain of digital games for developing software engineering skills by practical experiences.

Considering games, Caulfield et al [32] conducted a systematic review of the literature looking for games or simulations used for educational or training purposes in software engineering or software project management across any of the Software Engineering Body of Knowledge (SWEBOK) areas. The authors mapped 36 papers, describing 26 studies, and the results showed that games were mainly used in the SWEBOK of software engineering management

and development processes. Relating to our results, these knowledge areas are equivalent to "Software Process" from SE 2014 [1]. Therefore, our results endorse and expand the results from Caulfield et al, showing that not only educational games, but game related approaches in general still have a greater focus on topics of software process and management.

The study of Caulfield et al [32] has shown that, as a pedagogical device, games are becoming more common and students enjoy playing them and feel that they get some value from the experience. Despite 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.

Malik and Zafar [33] have performed a mapping study to give an overview of the current state of software engineering, by identifying the latest advancements taken place to improve the state of the area. The authors mapped 70 primary studies on three reference curriculums; SE 2004 (Software Engineering Curriculum Guidelines for Undergraduate, 2004), GSwERC (Graduate Software Engineering Reference Curriculum) and SWEBOK (Software Engineering Body of Knowledge). They found that the knowledge studies related to the areas of software engineering management in the three reference curricula are the most recurring. Therefore, our results are convergent, as we have discussed, the greater focus of game related approaches are on "Software Process" from SE 2014 [1], where topics related to software engineering management are grouped.

Concerning the adoption of game related approaches, Pedreira et al [11] conducted a systematic mapping study to characterize the state of the art of gamification in software engineering in the context of software development. Thus, it does not include those pieces of work focusing on teaching or training. The 29 selected studies were classified in terms of software engineering processes that have been the object of gamification, according to the software process defined in the ISO 12007 standard. The results showed that software requirements, software development and software testing are the areas which attracted the greatest interest in the field of gamification. 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. Comparing with our results, the research on the adoption of gamification in education is still in earlier stages, therefore reinforcing our position that gamification is still an open field for research.

VIII. Conclusion and Future Work

This paper identifies and analyzes the relationship between game related approaches and software engineering education knowledge areas. In order to do this, we conducted a systematic mapping study to survey the relevant literature. We found 106 studies published in major conferences and journals of the area, since 1974. We classified the proposed approaches in four categories: Game Based Learning (GBL), Game Development Based Learning (GDBL), Gamification, and Hybrid approaches.

We found that the GBL and GDBL approaches were the most prevalent, with 57 and 38 studies respectively. There were also some authors who took advantage of these two approaches and created innovative teaching methods, that we called "hybrid" approaches. Gamification was the least used approach in software engineering education (8 primary studies), what is expected due to the novelty of gamification. We believe Gamification is a relevant approach to explore in the context of software engineering education.

The analysis of the selected studies revealed that the most covered SEEK areas were "Software Process" with 54 approaches, "Software Design" with 42 approaches and "Professional Practice" with 42 approaches. For Software Process education, GBL is the most used approach, while GDBL is largely used to support Software Design education. Professional practice education is often tackled as a secondary goal. We found few approaches supporting "Software Quality" and "Software Modelling and Analysis" knowledge areas.

We advise researchers interested in proposing game related approaches for software engineering education to seek for adherence to curricular guidelines, such as SE 2014. This is important for software engineering education to thrive for standardization.

We also noted the lack of a common vocabulary for categorizing the GBL approaches found. Despite some recurring and more famous game types (e.g.: simulation, role-playing game, card games, board games) many studies did not provide a game genre or type for categorizing the approaches. Similarly, Blooms taxonomy [34] was used in some studies for categorizing learning goals in relation to the expected level of skill development, but its adoption was not consensus. We advise the research community to seek for a common vocabulary for game related approaches in software engineering education, in order to solidify knowledge and to facilitate secondary studies in summarizing results.

We hope the results of this work provide an overview about game based approaches in software engineering education for educators, learners, and researchers in Software Engineering Education. We also hope to encourage researchers to develop new software engineering educational approaches focusing on the support of knowledge areas, specially supporting the least supported knowledge areas "Software Quality" and "Software Modelling and Analysis".

For future work, we plan to consolidate the results of this systematic mapping and providing an in-depth classification of the available game approaches and support for software engineering education knowledge areas.

IX. Acknowledgement

This work was partially supported by CAPES, CNPq (grant 424340/2016-0), and FAPEMIG (grant PPM-00382-14).

REFERENCES

[1] IEEE & ACM JTFCC, "Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering," in EEE & ACM;The Joint Task Force on Computing Curricula, November 23 2015.

[2] C. G. V. Wangenheim and F. Shull, "To game or not to game?," IEEE Software, vol. 26, no. 2, pp. 92-94, 2009.

[3] M. D. Ernst and J. Chapin, "The groupthink specification exercise," in Proceedings. 27th International Conference on Software Engineering (ICSE), pp. 617–618, 2005.

[4] N. Tillmann, J. d. Halleux, T. Xie, S. Gulwani and J. Bishop, "Teaching and learning programming and software engineering via interactive gaming," in 35th International Conference on Software Engineering (ICSE), 2013.

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

[6] B. S. Akpolat and W. Slany, "Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification," in 27th Conference on Software Engineering Education and Training (CSEE&T), 2014.

[7] L. Singer and K. Schneider, "It was a bit of a race: Gamification of version control," in 2nd International Workshop on Games and Software Engineering (GAS), 2012.

[8] M. b. Yampolsky and W. Scacchi, "Learning game design and software engineering through a game prototyping experience: A Case Study," in Proceedings of the International Conference on Software Engineering (ICSE), p 15–21, 2016.

[9] E. Keenan and A. Steele, "Exploring game architecture best-practices with classic space invaders," in Proceedings of the 1st International Workshop on Games and Software Engineering (GAS), pp. 21–24, 2011.

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

[11] O. Pedreira, F. García, N. Brisaboa and M. Piattini, "Gamification in software engineering – A systematic mapping," in Information and Software Technology, vol. 57, pp., 2015.

[12] B. Wu and A. I. Wang., "A guideline for game development-based learning: a literature review," International Journal of Computer Games Technology, p. 8, 2012.

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

[14] J. Bell, S. Sheth and G. Kaiser, "Secret ninja testing with HALO software engineering," in Proceedings of the 4th International Workshop on Social Software Engineering (SSE), 2011.

[15] E. O. Navarro and A. V. D. Hoek, "Design and evaluation of an educational software process simulation environment and associated model," in 18th Conference on Software Engineering Education Training (CSEE&T), pp. 25–32, 2005.

[16] T. Wang and Q. Zhu, "A software engineering education game in a 3-D online virtual environment," in First International Workshop on Education Technology and Computer Science (ETCS) '09. vol. 2, pp. 708–710., 2009.

[17] E. Ye, C. Liu and J. A. Polack-Wahl, "Enhancing software engineering education using teaching aids in 3-D online virtual worlds," in 7th Annual Frontiers In Education Conference (FIE), 2007.

[18] A. Jain and B. Boehm, "SimVBSE: Developing a game for value-based software engineering," in 19th Conference on Software Engineering Education Training (CSEET), pp. 103–114, 2006.

[19] J. H. Andrews, "Killer App: A Eurogame about software quality," in 26th International Conference on Software Engineering Education and Training (CSEE&T), pp. 319–323, 2013.

[20] C. Szabo, "Evaluating GameDevTycoon for teaching Software Engineering," in Proceedings of the 45th Technical Symposium on Computer Science Education (SIGCSE), pp. 403–408, 2014.

[21] K. Shaw and J. Dermoudy, "Engendering an empathy for software engineering," in Conferences in Research and Practice in Information Technology Series, 2005.

[22] 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," in Frontiers in Education Conference (FIE), pp. 1–8., 2015.

[23] T. C. Kohwalter, E. W. G. Clua and L. G. P. Murta, "SDM - An educational game for software engineering," in Brazilian Symposium on Games and Digital Entertainment (SBGAMES), 2011.

[24] J. Bergin and F. Grossman, "Extreme construction: making agile accessible," in AGILE 2006 (AGILE),, 2006.

[25] O. AlsaedI, Z. Toups and J. Cook, "Can a team coordination game help student software project teams?," in Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2016.

[26] E. Knauss, K. Schneider and K. Stapel, "A Game for taking requirements engineering more seriously," in Third International Workshop on Multimedia and Enjoyable Requirements Engineering - Beyond Mere Descriptions and with More Fun and Games, pp. 22–26, 2008.

[27] R. Smith and O. Gotel, "Gameplay to introduce and reinforce requirements engineering practices," in 16th International Requirements Engineering Conference (RE), pp. 95–104, 2008.

[28] A. Rusu, R. Russell and R. Cocco, "Simulating the software engineering interview process using a decision-based serious computer game," in 16th International Conference on Computer Games (CGAMES), 2011.

[29] N. Tillmann, J. D. Halleux, T. Xie and J. Bishop, "Constructing coding duels in pex4fun and code hunt," in International Symposium on Software Testing and Analysis (ISSTA), 2014.

[30] H. Potter, M. Schots, L. Duboc and V. Werneck, "InspectorX: A game for software inspection training and learning," in IEEE 27th Conference on Software Engineering Education and Training (CSEE&T'), 2014.

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

[32] C. Caulfield, J. C. Xia, D. Veal and S. P. Maj, "A systematic survey of games used for software engineering education," in Modern Applied Science, 5(6), 28, 2011.

[33] B. Malik and S. Zafar, "A systematic mapping study on software engineering education," in Proceedings of World Academy of Science, Engineering and Technology (No. 71, p. 2061), 2012.

[34] R. Britto and M. Usman, "Bloom's taxonomy in software engineering education: A systematic mapping study," in Frontiers in Education Conference (FIE), 2015.