Playing the Project: Incorporating Gamification into Project-based Approaches for Software Engineering Education

Maurício R. de A. Souza[1], Renata Teles Moreira[2], Eduardo Figueiredo[3]

[1,3]Computer Science Department – Federal University of Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

[2]Computer Science Department – Federal University of Lavras (UFLA)
Lavras – MG – Brazil

{mrasouza, figueiredo}@dcc.ufmg.br, renata@dcc.ufla.br

Abstract. *In Software Engineering higher education, besides learning theory and acquiring technical skills, students need to develop the ability to apply, evolve, and practice those skills throughout their lifetime. From 2016 to 2018, we had been adapting an introductory Software Engineering course from a theory-oriented course to a more practical experience for students. Therefore, the goal of this paper is to describe our experience incorporating Project-based Learning (PBL) principles and gamification elements in practical assignments for SE education. To achieve this goal, we present the structure of PBL projects we have been using and how we adapted game elements to the context of SE education in order to create a gamified course. Based on our experience, we defined 17 guidelines that instructors may follow when defining a practical assignment for SE education. While PBL was helpful to create a practice-oriented teaching experience, gamification was not only useful in improving the engagement of students, but also in creating a roadmap of activities and a structure for students to self-assess their progress in the project.*

1. Introduction

In Software Engineering education, besides learning theory and acquiring technical skills, students need to develop the ability to apply, evolve, and practice those skills throughout their academic life (Gary, 2015). Additionally, soft skills, such as leadership, teamwork, decision-making, negotiation, and self-reflection, are important abilities for software engineering practice, since software development also involves several human and social aspects (Marques et al., 2014). Nevertheless, the development of these crosscutting capabilities is usually less supported in Computer Science programs (Marques et al., 2014).

Traditional approaches (e.g., expository lectures, exams, and complimentary assignments) are still largely used by lecturers (Marques et al., 2014; Bessa et al., 2012; Sancho-Thomas et al., 2009). However, these teacher-centered educational methods may not support the practical development of competences (Sancho-Thomas et al., 2009) and may have limited learning efficiency (Prikladnicki et al., 2009). Therefore, student-centered approaches have shown to be more suited for allowing the development of competences, in learn-by-doing, with a higher motivation from the learner, a more active role in learning process, and better learning in the application level (Prikladnicki et al., 2009).

ACM/IEEE Curriculum Guidelines for Software Engineering programs – SE 2014 (IEEE/ACM, 2015) – recommend including team-based projects into the Software Engineering and Computer Science curriculum. The needs of providing real world experience of software development to students are a recurring theme in several guidelines of SE 2014.

Project-based Learning (PBL) is one of the main successful student-centered educational approaches broadly used in computing science, information systems and software engineering courses to introduce practice in the learning process (Marques et al., 2018; Jazayeri, 2015). PBL is a comprehensive perspective focused on teaching by engaging students in investigation, in which students pursue solutions to nontrivial problems by a series of actions that result in a product (Blumenfeld, 1991). In addition to that, gamification has also been used as an educational resource in software engineering education to systematically motivate students in practicing specific skills or professional practices (Souza et al., 2018). Gamification is the use of game elements in non-game contexts (Deterding et al., 2011).

Since 2016, we have been adapting an introductory software engineering course to move it from a theory-oriented course to a more practical experience for students. Therefore, the goal of this paper is to describe our experience incorporating PBL principles and gamification in practical assignments in software engineering education. To achieve this goal, we present the structure of PBL projects we have been using and how we adapted game elements to the context of software engineering education in order to create a gamification approach. We then define some guidelines that instructors may follow when defining a practical assignment for software engineering education. While PBL was helpful to create a practice-oriented teaching experience, gamification was not only useful in improving the engagement of students, but also in creating a roadmap of activities and a structure for students to self-assess their progress in the project.

The remainder of this paper is structured as follows. Section 2 describes the theoretical background for PBL and gamification. Section 3 presents the details of the course where we introduced thee proposed approach. Sections 4 and 5 describe the format of the practical assignment, regarding the project structure and the gamification approach, respectively. Section 6 discusses how the assignment format evolved through five installments until the final structure. Section 7 presents lessons learned in the format of guidelines. Finally, Section 8 concludes the paper with final remarks.

2. Background

This section presents the concepts for the two educational resources used in our intervention: PBL and gamification.

2.1. Project-based Learning (PBL)

PBL is a comprehensive approach to classroom teaching and learning where students are engaged in the investigation of realistic problems, as they learn by working on an open-ended project, discovering problems and finding solutions as they go along (Blumenfeld et al., 1991; Jazayeri, 2015). Bender (2012) describes PBL as an instructional model based on having students confront real-world issues and problems that they find

meaningful, determine how to address them, and then act in a collaborative fashion to create problem solutions. In PBL, the instructor has a less central role, acting as a guide, and students take more responsibility for their own learning, which results in higher student involvement (Jazayeri, 2015).

Blumenfeld et al. (1991) describe five characteristics that define PBL projects: (1) the project should be driven by a question or problem without a predetermined solution; (2) the project should result in a series of artifacts that culminate in a tangible final product that addresses the driving question/problem; (3) the project should be realistic, grounded in real world problems; (4) the project should allow students to work collaboratively with peers and instructors to construct knowledge; and (5) the project should allow students to have active voice - i.e., room to decide over how to achieve their goals and negotiate some aspects of the project.

2.2. Gamification

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 (Deterding et al., 2011). 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 (Pedreira et al., 2015).

Gamification has been used in software engineering education as a strategy to engage and motivate students in performing specific behaviors, such as the more frequent use of specific tools, acquiring the habit of applying specific techniques, or being more participative in the classroom (Souza, et al, 2017; Singer and Schneider, 2012). Gamification has also been 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 (Laskowski, 2015). Therefore, it is a relevant strategy to support students in developing an appreciation for continued learning and in acquiring habits for professional software development (Singer and Schneider, 2012; Laskowski, 2015; Souza et al., 2018). Similar to PBL, a problem related to gamification is the difficulty to adapt it to each context, as there are few studies providing general guidelines to use this technique for software engineering education.

3. Details of the Course

The Software Engineering course (“SE course”, henceforth) is a 60 hours introductory course offered every semester at Federal University of Lavras (UFLA), included in the curriculum of the Information System Bachelor undergraduate program. The SE course aims to introduce students to the concepts and methods required for the development of large software intensive systems. The prerequisite for taking this course is the approval in the Object-Oriented Programming course. The course syllabus includes: software development process, agile methods, software requirements analysis and specification, software design, system implementation and testing, configuration management, and software quality.

In previous iterations of the SE course (before 2016), it was heavily theoretical, relying mostly on traditional lectures, exams and practical exercises where students had

to design a fictitious software through using UML diagrams. From 2016 to 2018, we adapted the course in order to introduce a more practical approach. We shifted from the teacher-centered method to PBL and introduced a practical assignment in the format of a project, equivalent to 60% of the course grades. Two exams and punctual exercises were responsible for the remaining grades.

Table 1 presents the overall learning goals of the practical assignment. The SE course covers not only technical skills related to specific software engineering topics, but the assignment was also meant to expose students to the opportunity of developing crosscutting skills described as important for software engineers in SE 2014 (IEEE/ACM, 2015), such as professional knowledge, teamwork, design of solutions in context, end-user awareness, and continuing professional development.

Table 1. Learning goals of the practical assignment

Question	How can we systematically design and develop a software product to meet customer needs?
Topics of Interest	Course Syllabus {software requirements analysis and specification, software design, system implementation and testing, configuration management}
Specific Skills	- Elicit customer needs and describe system requirements - Design software in accordance to functional and non-functional requirements - Develop software in accordance to design decisions - Test software using appropriate methods - Use appropriate version control tools to manage the evolution of the software
General Skills	**Professional Knowledge:** Show mastery of software engineering knowledge and skills and of the professional standards necessary to begin practice as a software engineer. **Teamwork:** Work both individually and as part of a team to develop and deliver quality software artifacts. **Design Solutions in Context:** Design appropriate solutions in one or more application domains using software engineering approaches that integrate ethical, social, legal, and economic concerns • Perform Trade-Offs: Reconcile conflicting project objectives, finding acceptable compromises within the limitations of cost, time, knowledge, existing systems, and organizations. **End-User Awareness:** Demonstrate an understanding and appreciation of the importance of negotiation, effective work habits, leadership, and good communication with stakeholders in a typical software development environment. **Continuing Professional Development:** Investigate and use appropriate up-to-date tools and technologies suitable for the execution of the project.

4. The Project Structure for the Practical Assignments

The project adopted some principles from PBL:

- **Project-based:** The software development project is a central part of the course. All classroom activities and lectures are driven by the progression of the project.
- **Driving questions:** All activities of the project are driven by meaningful questions that directs students into investigating and applying SE theory for the project.
- **Evidence based:** The students have to provide evidences of performing tasks (mostly related to software development life cycle). These tasks are based on the skills from the Software Engineering Competence Model – SWECOM (Ardis et al., 2014).

- **Realistic:** The project is based on real world problems, and the instructors play the role of customers during the course. The development of the project is organized in iterations, inspired by the Scrum sprints and ceremonies (Schwaber and Sutherland, 2016).
- **Balance between guidance and freedom of choice:** The instructors act at mentoring activities to guide students in the execution of the project. However, students do not follow a rigid process, as they have freedom to select the technologies and tools they prefer in order to achieve the goals of the project.
- **Teamwork:** The students have to work in teams, typically from 3 to 5 students.

An anchoring problem of the project is discussed in the first class of the semester. The students form teams of 3 to 5 members. The goal of the project is the development of a software product. Instructors act as stakeholders of the project or interfaces between students and external customers.

The project is developed in three iterations (as described in Figure 1), resulting in partial product artifacts: the first devoted to the definition of the product specification; the second focusing on the design and development of a first minimum viable product (MVP); and the third for the development of a second increment of the product and verification activities.

By the end of each iterations students present their results for the instructors for validation (similar to the Scrum Review ceremony) and then the entire class perform a reflection activity (similar to the Scrum Retrospective ceremony). These activities are intended to allow instructors to evaluate students, and to foster reflection and inquiry on the use of software engineering principles to solve problems and to achieve the goals of the iteration.

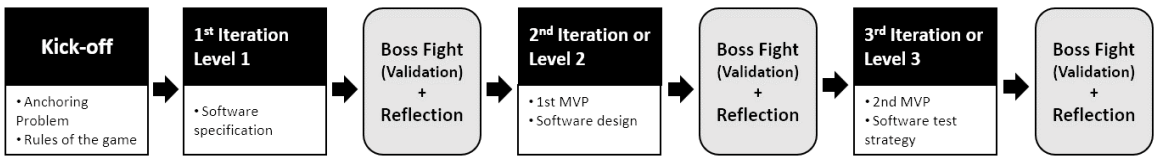

Figure 1. The lifecycle of the educational project

5. Introducing Gamification

In order to create a gamification approach, we selected some game elements and adapted them to the context of software engineering projects. The game elements we chose are: narrative, levels, goals, points, badges, leaderboards, feedback, time constraints, boss fights, prizes, narrative, teams. Details about the use of these elements are further explained in the following subsections.

5.1. Anchoring the Project as a Game (Narrative and Teams)

We use narrative to contextualize the project in an immersive environment. For instance, we usually introduce the anchoring problem as a customer running a selection process for a provider. The student teams are presented as very small software companies that have to create a prototype solution to compete.

5.2. The Process as a Game (Levels, Goals, Boss Battles, and Feedback)

The iterations of the project are structured as game levels. They have explicit objectives (e.g., defining the product specification). During each level, the instructors provide students with a set of tasks (Goals). These tasks are intended to guide students toward the accomplishment of the level, and to motivate the use of specific tools (such as version control systems and project management tools). The solutions for these tasks are not predefined, i.e., students have to reflect and plan how to accomplish them. In order to define those tasks, we used skills from SWECOM (Ardis et al., 2014) as reference.

To complete a task successfully, the teams have to provide evidences of accomplishing their objectives. The instructors evaluate these evidences and provide students with feedback for improvement. However, feedbacks are conditioned to finishing the tasks within specified deadlines (simulating the time pressure in a project), and, therefore, students have the opportunity to improve their evidences (chance to fail) as long as they respect the project chronogram (time constraints).

The validation activities are then performed as "boss battles", i.e., special challenges by the end of the level. In our approach, these boss battles serve as an opportunity for the instructors to evaluate students individually, and to observe the students' learning progress. In the boss battles, the students give an oral presentation of the artifacts they developed, and instructors may ask specific details.

5.3. Rewards (Badges, Points, Leaderboards, Hall of Fame, and Prizes)

The main goal of gamification in our approach is to motivate students in practicing software engineering skills, techniques and tools, by systematically rewarding and acknowledging students' actions that converge for the expected learning goals or that contribute for the project execution. Reward dynamics were introduced to our approach by using badges, points, and prizes. The teams are rewarded with badges for each task they successfully complete in time. These badges are presented as bronze, silver and gold trophies, according to the complexity of the tasks. In turn, these badges are worthy 3, 5 or 7 points, respectively. The teams are then ranked in a leaderboard, according to their score. The top three teams are displayed in the hall of fame and are the "winners" of the course (promoting social recognition of the more dedicated students). Additionally, we offer a collective challenge for the class, having a prize (some pizza) for the entire class if the teams are able to earn more than 80% of the trophies in total.

6. Discussion on the Evolution of the Course Assignment

The assignment format described in Sections 4 and 5 is a result of an experimentation process that occurred through five installments of the SE course of the Information System undergraduate program. Table 2 provides an overview of each course installment, with details about the projects executed, number of students and teams, and the use of PBL and Gamification. All installments had in common the learning goals, the format of three iterations, and the use of PBL. However, there were some issues, further discussed in this section, that motivated changes that culminated in the final structure of the assignment.

In the first installment of the course using PBL (2016.1), we used a real project and invited a real external stakeholder to act as requirement provider and for validating

the resulting product of the project. However, in the middle of the project, managing the commitment and availability of the external stakeholder became a problem. In this installment, all students work as a single team to develop the project. However, we observed that students divided themselves in sub teams specialized in specific tasks. This action had serious impact in the learning process of students, as the students practiced only specific phases of the development lifecycle.

Table 2. Details of the five installments of the course.

Class	#Students and Teams	Project	PBL?	Gamification?
2016.1	13 (1 team)	A software solution for the managing the lending of keys and equipments for lectures at UFLA	Yes	No
2016.2	18 (5 teams)	Teams were free to choose	Yes	No
2017.1	7 (1 team)	A software for the management of a fictitious hostel	Yes	Yes
2017.2	11 (2 teams)	A software for the management of a real small factory	Yes	Yes
2018.1	14 (4 teams)	A gamified system for the management of practical assignments.	Yes	Yes

In the second installment (2016.2), students were organized in small teams of 3 to 5 students, and each team had to choose their own projects to execute. We observed that students had trouble in assessing the viability of their proposed projects, resulting in changes in the scope of their projects in the middle of the course. In addition, as instructors we found difficult to provide mentoring for all teams because the projects were so different that it was difficult to promote discussions in the class.

In both 2016.1 and 2016.2 the course evaluation rubrics provided criteria that students could use as orientation about what to focus on the assignment execution, and classroom time was devoted to mentoring students in the project execution. However, we observed that students felt lost in the execution of the iterations. They were confused about what they should focus, and some were afraid that their limited development experience would affect their grade as they were concerned about not being able to develop a functional software product.

Starting from the 2017-1 course, we introduced gamification to the assignment format, in order to provide additional motivation and guidance for students in relation to the adoption of software engineering techniques, tools and good practices. Game elements were inserted to create a goal-oriented process for the project, by structuring goals and rewards that provide students with a roadmap of activities, with varying levels of difficulties, and that provide students with a clear mechanism to assess their progress in the project.

From 2017.1 on, we also opted to have the same project for all. In case of fictitious projects (as the case of class 2017.1), we created a narrative (as described in Section 4.1) to contextualize the project. In case of real projects, the instructors acted as an interface between students and the actual requirement providers, in order to achieve more control over the assignment. We observed that it was important to have a balance between realism and control, and that the realism could be achieved by using convincing methods that simulate professional environment rather than exclusively by relying on real projects with real customers. The fact that the teams were developing real working

products, was challenging and rewarding for students. In every installment of the course, all teams were capable of delivering functional prototypes, with varied levels of quality.

7. Guidelines for Defining Practical Assignments in SE Education

Based on our experiences reported in this paper, we defined a set of guidelines we use when planning practical activities for SE education. These guidelines rely on concepts from PBL and Gamification, adapted to the context of software engineering projects. As reported by Rodrigues et al. (2018), the lack of guidelines and support material may be a barrier for lecturers, regarding the use of games and gamification in SE education.

Table 3. PBL and gamification guidelines for practical assignments in SE education

ID	Guidelines
GL01	*Software engineering education needs to move beyond the lecture format and to consider a variety of teaching and learning approaches.*
GL02	*The assignment should have significant real-world basis, grounded in realistic problems or methods, in order to provide an interesting, concrete and convincing example of software engineering practice.*
GL03	*The assignment should use software process to contextualize software engineering practice and promote the use of appropriate and up-to-date tools, methods, and standards.*
GL04	*The assignment should promote the development not only of technical knowledge, but also expose students to personal and professional skills related to software engineering.*
GL05	*The design of the gamification approach should have software engineering as a central theme and should be tailored for specific contexts.*
GL06	*Gamification should systematically support the development of specific behaviors that converge to expected learning outcomes or contributes for the project execution.*
GL07	*The gamification approach should provide students with directions for the execution of the project and mechanism to track their progress.*
GL08	*The assignment should allow students to have significant autonomy of choice, and motivate continued and self-directed learning for the development of solutions in context.*
GL09	*The assignment should allow students to work collaboratively with peers and instructors to construct knowledge.*
GL10	*The assignment should be driven by a question or problem without a predetermined solution.*
GL11	*The assignment should result in a series of artifacts that culminate in a tangible final product that addresses the driving question/problem.*
GL12	*The project should balance realism (authenticity) with level of control and viability, in order to allow the achievement of learning goals while remaining significant.*
GL13	*Instructors should act more as knowledge facilitators than as knowledge transmitters.*
GL14	*The planning of project stakeholders should consider risks related to commitment, availability, and authenticity.*
GL15	*The gamification approach should promote social recognition of the students' efforts.*
GL16	*The assignment rubrics should provide clear indication of what students should focus on during the project execution and should consider some aspects of the gamification approach.*
GL17	*Instructors should provide sufficient guidance for students towards the conclusion of the project and the achievement of learning goals.*

8. Conclusion

This paper described our experience in evolving a strategy to apply PBL and gamification in an introductory software engineering course. We observed that PBL was

useful in creating a practical and immersive experience for students. In turn, gamification was useful to create a roadmap of activities to guide students in the execution of their projects, anchoring the project in a ludic and engaging format, and motivating students into applying best practices and tools that were not the primary focus of the assignment. We observed that the gamification approach did not push the students to compete with each other. Instead, students were more impacted by the possibility to measure their progress, being recognized for their efforts, and by working together to earn the collective prize.

We found the adoption of this approach satisfactory in terms of introducing practice in the learning process. The feedback from students has been most positive regarding the simulation of professional environment and escaping the traditional lecture format. Initially students complained about the need of more details about how they were going to be evaluated (i.e., what specifically they should do to obtain sufficient grades for approval). This problem was initially addressed by using classroom time for mentoring students about the next steps, and by providing students with a checklist with the course evaluation rubrics. The later was replaced when we introduced the gamified approach, by having the entire roadmap of activities disclosed by the beginning of the project, and by establishing a clear relationship between the completion of tasks and the course evaluation rubrics.

However, we observed that this course format is difficult to scale without proper tools. To support gamification, the mechanics and rules need to be sustained through the entire course, and a higher number of teams may be impracticable. Similarly, to provide the sufficient guidance for students in larger classes, instructors may need additional support of teaching assistants to help mentoring all teams.

The contributions of this experience report are a set of insights and guidelines that can be useful for educators in applying these resources for practical assignments. We are currently replicating this format of assignment and guidelines in varied courses of a different institution: Web development, Software Quality, and Software Processes. As a future work, we intend to propose a framework to support SE educators in systematically planning and executing SE related courses using PBL and gamification.

Acknowledgement

This work was partially supported by CAPES, CNPq (grants 424340/2016-0 and 142022/2017-9), and FAPEMIG (grant PPM-00651-17).

References

Ardis, M., Fairley, D., Hilburn, T., Nidiffer, K., Towhidnejad, M., and Willshire, M. (2014). The software engineering competency model (swecom). Technical report, Los Alamitos, CA, USA.

Bender, W. N. (2012) “Project-Based Learning: Differentiating Instruction for the 21st Century”. Corwin. ISBN 978-1-4129-9790-4.

Bessa, B., Cunha, M., and Furtado, F. (2012) “Engsoft: Ferramenta para simulação de ambientes reais para auxiliar o aprendizado baseado em problemas (pbl) no ensino de engenharia de software”. XX Workshop sobre Educação em Computação (WEI).

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., and Palincsar, A. (1991) “Motivating project-based learning: Sustaining the doing, supporting the learning”. Educational Psychologist, 26(3-4):369–398.

Deterding, S., Sicart, M., Nacke, L., O’Hara, K., and Dixon, D. (2011). Gamification. using game-design elements in non-gaming contexts. In CHI’11 Extended Abstracts on Human Factors in Computing Systems, pages 2425–2428.

Gary, K. (2015) “Project-based learning”. Computer, 48(9):98–100. ISSN 0018-9162.

IEEE/ACM, T. J. T. F. o. C. C. A. (2015) “Software engineering 2014: Curriculum guidelines for undergraduate degree programs in software engineering”. Technical report, New York, NY, USA.

Jazayeri, M. (2015) “Combining mastery learning with project-based learning in a first programming course: An experience report”. In IEEE/ACM 37th IEEE International Conference on Software Engineering, pages 315–318. ISSN 0270-5257.

Laskowski, M. (2015). Implementing gamification techniques into university study path – a case study. In 2015 IEEE Global Engineering Education Conference (EDUCON), pages 582–586. ISSN 2165-9559.

Marques, M. R., Quispe, A., and Ochoa, S. F. (2014) “A systematic mapping study on practical approaches to teaching software engineering”. IEEE Frontiers in Education Conference (FIE), pages 1–8. ISSN 0190-5848.

Pedreira, O., García, F., Brisaboa, N., and Piattini, M. (2015). Gamification in software engineering – a systematic mapping. Information and Software Technology, 57:157– 168. ISSN 0950-5849.

Prikladnicki, R., Albuquerque, A. B., von Wangenheim, C. G., and Cabral, R. (2009) “Ensino de engenharia de software: Desafios, estratégias de ensino e lições aprendidas”. Fórum de Educação em Engenharia de Software (FEES).

Rodrigues, P., Souza, M., & Figueiredo, E. (2018). “Games and Gamification in Software Engineering Education: A Survey with Educators”. In 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1-9).

Sancho-Thomas, P., Fuentes-Fernández, R., and Fernández-Manjón, B. (2009) “Learning teamwork skills in university programming courses”. Computers Education, 53(2):517–531. ISSN 0360-1315.

Schwaber, K. and Sutherland, J. (2016) “The scrum guide”. Available at: http://www. scrumguides. org/docs/scrumguide/ v1/scrum-guide-us.pdf.

Singer, L. and Schneider, K. (2012). It was a bit of a race: Gamification of version control. In 2012 Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS), pages 5–8.

Souza, M. R. A., Constantino, K., Veado, L., and Figueiredo, E. (2017). “Gamification in Software Engineering Education: An Empirical Study” in IEEE 30th Conference on Software Engineering Education & Training (CSEE&T).

Souza, M. R. A., Veado, L., Moreira, R. T., Figueiredo, E., and Costa, H. (2018). “A systematic mapping study on game-related methods for software engineering education”. Information and Software Technology, 95:201–218. ISSN 0950-5849.