Teaching Software Engineering with Gamification Elements

https://doi.org/10.3991/ijac.v11i1.9169

Sigrid Schefer-Wenzl[1] and Igor Miladinovic[1]
[1] University of Applied Sciences, Computer Science and Digital Communications, Vienna, Austria

***Abstract*—Students of software engineering courses in higher education often experience a lack of motivation, partly caused by traditional teaching methods. In our study program we introduced a novel blended learning concept with threefold gamification elements for teaching software engineering. In this paper we present the teaching method mix with particular focus on the integration of three gamification elements to increase students' engagement.**

***Index Terms*—experiential learning, gamification, software engineering education.**

I. Introduction

Learning is known to be most effective when it is active, experiential, situated, problem-based, and provides immediate feedback [4]. These are properties that traditional software engineering courses in higher education often do not meet. A typical software engineering course consists of the following two elements: a lecture where concepts and theories are taught as well as small programming projects enabling students to apply this knowledge. However, lectures only allow passive learning and projects are very constrained, structured and well-defined, which does not prepare students adequately for their future jobs [11]. Moreover, using this setting, teachers of software engineering courses are often confronted with students' lack of motivation to continuously practice their programming skills (see, e.g., [5, 8]), which leads to high dropout rates as well as high failure rates in software engineering courses.

One promising approach to increase students' motivation is the introduction of gamification elements into software engineering courses. Gamification is defined as the application of game design elements to non-game activities with the goal to increase user experience and engagement [16]. Using gamification in different educational contexts is known to be one way to enhance learner motivation and to improve learning outcomes by capturing the interest of learners and inspiring them to continue learning [9, 10].

Our University of Applied Sciences Campus Vienna offers a Bachelor program in Computer Science and Digital Communications, with key skills focusing on a solid understanding of software development as well as comprehension of modern software engineering principles. One of the key lectures for those skills is "Software Engineering" in the third semester. Until now this course was taught using traditional methods, such as frontal lecture and home assignments. These methods were not able to support diverse entry levels of students frequently resulting in failing to achieve the learning objectives for students with lower entry levels. To address this issue we developed a new threefold gaming approach for teaching software engineering, supported by different gamification elements. In this paper we present this concept and discuss impacts of gamification on the achievement of learning objectives.

The remainder of this paper is structured as follows. Section II presents related work on gamification in software engineering courses. In Section III, we introduce our course design for a Bachelor-level course. In Section IV, we focus on the gamification concepts we employed in our course. Finally, Section V concludes this paper.

II. Related Work

In recent years, several approaches have been proposed to motivate students and facilitate students' learning in the area of software development (see, e.g. [1, 2, 10]), some of them by using different kinds of game-based elements. Many authors discussing gamified software engineering courses state that students report improved content comprehension, retention and recap (see, e.g. [3]). This is mainly due to the practical application of the taught concepts in a game-based course.

Combefis et al. [8] analyzed several game-based online programming platforms. Amongst others they conclude that successful educational game platforms need to provide feedback and assessment, game elements need to be fun, and collaborative games and contests raise participation rates. According to Nah et al. [6], game design elements that are often used in an educational or learning context are experience points, levels/stages, badges, leaderboards, prizes/rewards, progress bars, storyline, and feedback.

Applying gamification to educational contexts has produced several promising results. However, a game-based approach is also associated with several risks. For example, Berkling and Thomas [5] introduced a gamification platform to teach a software engineering course. However, the results showed that students found the gamified environment as not being helpful, as they wanted to focus on relevant material for the exam.

Existing research in the field of applying gamification elements in software engineering education is still preliminary. More research on defining a systematic approach to design gamified learning activities, on relating learning goals with game-related methods as well as on evaluating the impact of gamification in software engineering education is needed.

III. COURSE DESIGN

A. *Learning Objectives*

We defined the following learning outcomes for our "Software Engineering" course:

- Understand and apply an effective software engineering process, based on knowledge of widely used software process models.
- Employ team working skills including organized planning, time management and inter-group negotiation.
- Capture, document and analyze requirements.
- Translate a requirements specification into an implementable design, following a structured process.
- Make effective usage of software design strategies.
- Design a testing strategy for a software system, employing techniques such as unit testing, test driven development and functional testing.
- Evaluate the final projects by checking compliance with the requirements, and analyze the design and implementation.

B. *Course overview*

The course consists of a lecture part and a tutorial part. The lecture part comprises the following four modules:

(1) Software engineering activities,
(2) Unified Modelling Language,
(3) process models, and
(4) invited lectures.

The software engineering activities module provides an overview of the main tasks of software engineering, such as requirements engineering, high level design, low level design, development and testing. The second module introduces Unified Modelling Language (UML) [12], enabling the students to graphically visualize the design of their future software projects. In the third part, different kinds of process models that structure the software development process are investigated and applied on smaller examples. The lecture part is completed with short lectures of four invited experts from several partner companies, who are involved in the software engineering process in their daily business.

During the lecture there are two example projects to illustrate how the learning content can be applied in practice. The first project is provided and presented by the lecturers demonstrating all aspects necessary for the implementation of the students' projects. The second project is elaborated by the students during the lecture to apply what they have learned on a simple project.

In the tutorial part, students work on a larger industry-like software gaming project, where they go through all software engineering phases according to a particular process model. Students are supported by the lecturers via regular coaching meetings.

C. *Applied Teaching Methods*

Figure 1 lists the applied methods for both, lecture and tutorial part. In the lecture we used blended learning to address different entry levels and learning patterns. The students prepare themselves for each lecture module with the provided learning material and generate control questions and corresponding answers in advance. The lecturers evaluate these inputs and select the most important questions to be discussed in the class. In-class sessions start with a recap of the learning content for this module, followed by a discussion on the selected questions. Afterwards, the obtained knowledge is applied on exercises in the context of the students' lecture projects.

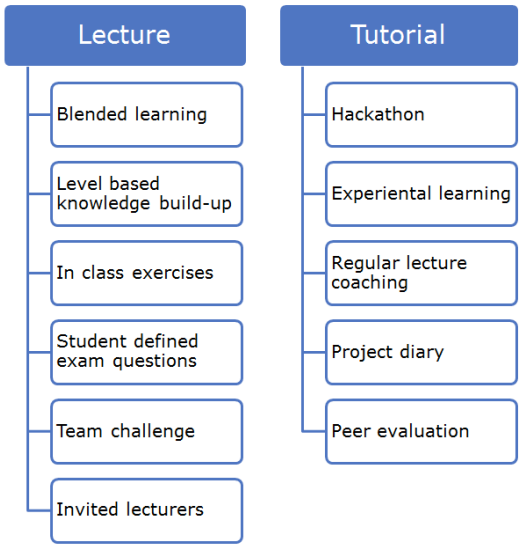

Figure 1. Method mix

In the tutorial part experiential learning via simulation of a real gaming software engineering project takes place. The whole tutorial is accompanied by regular lecturer coaching. The project starts with a Hackathon [13, 14] event, which is a six hours event where the students are brainstorming, discussing and evaluating their projects ideas. After the individual brainstorming phase there is a voting for each project idea by all students. One third of the ideas – those with the highest voting – are selected for the implementation. Each implementation is done in a group of three students with predefined roles: programmer, coordinator and designer. These roles have specified tasks and should reflect the roles of a real software project:

- The coordinator plans and documents the project, schedules regular meetings, keeps track of important decisions, talks with the customer, i.e. the lecturers, observes the whole development process and checks that all deliverables are provided in time.
- The designer defines the software architecture and the graphical user interface.
- The programmer is responsible for implementing and testing the project.

Students have to document their project progress in a project diary where they keep record of their decisions,

agreements, findings, problems and successes. These project diary blogs are readable for all course members. Each course member can comment on the blog entries and give suggestions for improvements. Constructive comments are awarded with bonus points.

The final projects are evaluated in a peer assessment manner, where the compliance with the requirements specification and the high-level design is checked.

IV. Threefold Gamification

To foster the motivation of students even more, we introduced three gamification concepts in our lecture: a challenge, a level based progress tracking and experiential learning through own game projects.

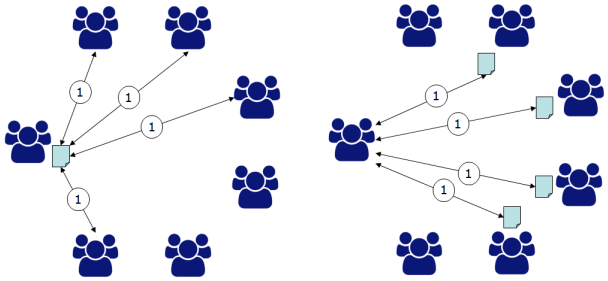

Figure 2. Team challenge

A. Challenge

The challenge is organized as a kind of competition between the development teams, where they compete for common resources, which are points for the grading in our lecture. The challenge applies for the requirements engineering and the high level design exercises of the lecture. Each group specifies these two documents for a given project (same for all the groups) and shares these specifications with a certain number of other groups (n). These groups challenge shared documentations and look for weaknesses in them. If no weaknesses are found, the group which owns the documents gets one grading point. Otherwise, the group which challenged the document gets that point. Figure 2 illustrates this process.

The total number of grading points which are achievable by this challenge can be adjusted by the number of groups which challenge the documentation. For each document – and there are two in our course – the maximum number of points is 2n-2.

B. Gaming Levels

The second gamification concept is based on knowledge levels according to Bloom's Taxonomy [15]. For our course we adapted the six original categories into four levels, as shown in Figure 3.

For each level we defined clear tasks and achievements to accomplish. The first level is Remember and Understand and the achievements are determined by a written examination. The Applied level is driven by lecture exercises deepening the theoretical knowledge and by self-created questions and answers about the lecture contents. The described challenge covers the Analyze and Evaluate level. Finally, the design and implementation of the own project in the tutorial part enable the students to move to the Create level.

When the students obtain at least half of the possible achievements from one level, they are promoted into the next level. This promotion is a precondition for achievements in the next level.

C. Experiential Learning

Simulating real-world projects is aimed to provide students with the experience in order to unify the practical understanding with the theoretical knowledge [10]. In our course, students are supposed to learn by experiencing all phases of a typical industry-like software project. They create a project idea and follow it until successful implementation. Therefore, students are likely to identify themselves with their projects resulting in an increased motivation. Applying the gamification concept here we intend to foster student engagement even more.

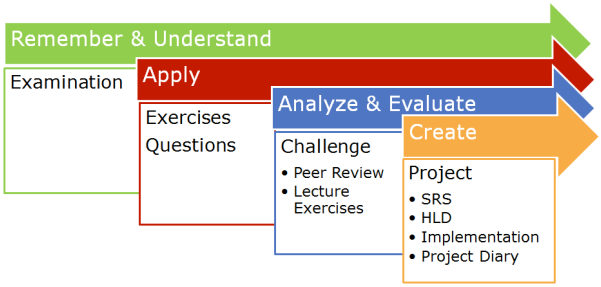

Figure 3. Knowledge levels

During the Hackathon event, students have to propose ideas for a project, which should be a gaming application. The best ideas are selected for implementation as described in Section III.C. Gaming applications have several advantages: A game needs clearly defined rules which constitute the requirements of the application. In addition, in a game functionality, design and performance are all important motivating the students not to neglect one of them. Usually games also frequently need a local database and cloud architecture which increases learning effects.

V. Conclusion

Different studies show that gamification can increase students' engagement in higher education. We designed a bachelor level software engineering course by combining a method mix with threefold gamification elements, *challenge*, *gaming levels* and *experiential learning*. With this concept we gradually address all the knowledge levels according to Bloom's Taxonomy. In future research, we will apply and evaluate this course concept in different student settings.

References

[1] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, "A survey of literature on the teaching of introductory programming", In: *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education, ITiCSE-WGR '07*, pp. 204–223, Dundee, Scotland. ACM, 2007. https://doi.org/10.1145/1345443.1345441

[2] A. Vihavainen, J. Airaksinen, and C. Watson, "A systematic review of approaches for teaching introductory programming and their influence on success", In: *Proceedings of the Tenth Annual Conference on International Computing Education Research, ICER '14*, pp. 19–26, Glasgow, United Kingdom. ACM, 2014. https://doi.org/10.1145/2632320.2632349

[3] P. G. F. Matsubara, C. L. Correa da Silva, "Game elements in a software engineering study group: a case study", In: *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track, ICSE-SEET '17*, pp. 160-169, Buenos Aires, Argentina, IEEE Press, 2017. https://doi.org/10.1109/ICSE-SEET.2017.8

[4] E. A. Boyle, T. M. Connolly, T. Hainey, "The role of psychology in understanding the impact of computer games", In: *Entertainment Computing*, vol. 2, nr. 2, 69–74, 2011. https://doi.org/10.1016/j.entcom.2010.12.002

[5] K. Berkling, C. Thomas, Gamification of a Software Engineering Course and a detailed analysis of the factors that lead to it's failure. In: *Proc. Of the International Conference on Interactive Collaborative Learning*, pp. 525–530, Kazan, Russia, IEEE Press, October 2013. https://doi.org/10.1109/ICL.2013.6644642

[6] F. F.-H. Nah, Q. Zeng, V. R. Telaprolu, A. P. Ayyappa, B. Eschenbrenner, "Gamification of Education: A Review of Literature", In: *Proc. of the International Conference on HCI in Business, HCIB 2014,* Heraklion, Crete, Greece, Lecture Notes in Computer Science, vol 8527, Springer, pp. 401-409, June 2014. https://doi.org/10.1007/978-3-319-07293-7_39

[7] D. J. Frailey, "The times, they are changing", In: *Software Engineering Education and Training*, pp.1-2, Klagenfurt, Austria, 2014. https://doi.org/10.1109/CSEET.2014.6816774

[8] S Combéfis, G. Beresnevicius, V. Dagiene, "Learning Programming through Games and Contests: Overview, Characterisation and Discussion", In: *International Olympiad in Informatics*, vol. 10, pp. 39–60. Vilnius, Lithuania, 2016. https://doi.org/10.15388/ioi.2016.03

[9] G. Barata, S. Gama, J. Jorge, D. Goncalves, "Improving Participation and Learning with Gamification", In: *Proc. of the 1st International Conference on Gameful Design, Research, and Applications*, Toronto, Ontario, Canada, 2013. https://doi.org/10.1145/2583008.2583010

[10] M. Kosa, M. Yilmaz, R. O'Connor, P. Clarke, "Software Engineering Education and Games: A Systematic Literature Review", In: *Journal of Universal Computer Science*, vol. 22, no. 12, pp. 1558-1574, 2016.

[11] Souza, Mauricio R. et al., "A systematic mapping study on game-related methods for software engineering education", In: Information and software technology, vol. 95, March 2018. https://doi.org/10.1016/j.infsof.2017.09.014

[12] Object Management Group, "Unified Modeling Language (UML). Version 2.5.1". available at: https://www.omg.org/spec/UML/2.5.1/, Dec. 2017.

[13] B. Rosell, S. Kumar, and J. Shepherd, "Unleashing innovation through internal hackathons", In: *IEEE Innovations in Technology Conference*, pp. 1–8, May 2014. https://doi.org/10.1109/InnoTek.2014.6877369

[14] E. H. Trainer, A. Kalyanasundaram, C. Chaihirunkarn, and J. D. Herbsleb, "How to hackathon: Socio-technical tradeoffs in brief, intensive collocation", In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW'16, pp. 1118–1130, New York, NY, USA, 2016. https://doi.org/10.1145/2818048.2819946

[15] B. S. Bloom, "Taxonomy of educational objectives, handbook I: Cognitive domain.", New York: David McKay, 1956.

[16] S. Deterding, D. Dixon, R. Khaled, L. Nacke, "From game design elements to gamefulness: defining 'gamification'". In: *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, Tampere, Finland, September 2011. https://doi.org/10.1145/2181037.2181040

AUTHORS

Sigrid Schefer-Wenzl is a senior researcher and lecturer at the University of Applied Sciences Campus Vienna, Austria (e-mail: sigrid.schefer-wenzl@fh-campuswien.ac.at).

Igor Miladinovic is head of the degree program Information Technologies and Telecommunication at the University of Applied Science Campus Vienna, Austria (e-mail: igor.miladinovic@fh-campuswien.ac.at).

Manuscript received 15 March 2018.

Published as submitted by the author(s).