IMPRESS: Improving Engagement in Software Engineering Courses through Gamification*

Tanja E.J. Vos[1], I.S.W.B. Prasetya[2][0000000234214635], Gordon Fraser[3], Ivan Martinez-Ortiz[4], Ivan Perez-Colado[4], Rui Prada[5], José Rocha[5], and António Rito Silva[5][0000−0001−9840−457X]

[1] Open Univeriteit Nederland [2] Utrecht University [3] Universität Passau
[4] Universidad Complutense de Madrid [5] INESC-ID and Instituto Superior Técnico, Universidade de Lisboa

Abstract. Software Engineering courses play an important role for preparing students with the right knowledge and attitude for software development in practice. The implication is far reaching, as the quality of the software that we use ultimately depends on the quality of the people that make them. Educating Software Engineering, however, is quite challenging, as the subject is not considered as most exciting by students, while teachers often have to deal with exploding number of students. The EU project IMPRESS seeks to explore the use of gamification in educating software engineering at the university level to improve students' engagement and hence their appreciation for the taught subjects. This paper presents the project, its objectives, and its current progress.

Keywords: software engineering education, gamification in education, gamification in software engineering education

1 Introduction

While our society increasingly depends on software for various aspects of civic, commercial and social life, software engineers struggle to ensure that software achieves the necessary high quality. The increasing complexity of modern software systems and the ever reducing time-to-marked further exacerbate the problem. Although the discipline of Software Engineering offers different techniques to ensure quality, programmers in practice are reluctant to engage with them, with detrimental effects on software quality. The root of this situation lies in how software developers are educated. The focus tends to lie on the creative aspects of design and coding, whereas the more laborious and less entertaining necessities to assure the software's quality are neglected. This disengagement carries over to practice. This has to change: tomorrow software engineers need to be raised with appreciation of software quality, and quality assurance techniques need to become a natural aspect of software development, rather than

* The IMPRESS project https://impress-project.eu/ is funded by EU Erasmus+ Programme, grant nr. 2017-1-NL01-KA203-035259. Duration: 2017-2020. Partners: Open Univ. (NL), Utrecht Univ. (NL), Univ. Complutense Madrid (SP), Univ. Passau (DE), INESC-ID Lisbon (PT). The project is also partially funded by the Fundacão para a Ciência e a Tecnologia (FCT) fund UID/CEC/50021/2019.

a niche topic. Implementing the change, however, is not easy, as teachers have to motivate students through materials already branded as uninteresting. To help teachers, the IMPRESS project seeks to explore the use of gamification, i.e., the application of game-design elements and game principles in non-gaming contexts, which has seen successful applications in other domains. This paper will present the project objective, the results so far, and a conclusion.

2 IMPRESS Expected Outcomes

Although *gamificaton* is known to improve users' engagement and appreciation [4], its application to Software Engineering is still limited. IMPRESS seeks to deliver innovations that would help improving students' engagement and enthusiasm on topics traditionally considered boring. It will focus on the following:

(1) Improving *in-class engagement* through gamified quizzes. Quizzes are an effective tool to set a course's pace. A cleverly setup quiz can trigger an engaging discussion, while gamification can stimulate wider engagement through competitive elements. A set of quizzes from selected topics will be developed within the project, along with tools to let others to develop more.

(2) Improving *out-class engagement* through educational games that can be played at home or in unguided lab sessions. We will focus on the subject of quality assurance —a key subject, as pointed out earlier—, in particular in two key competences: formalizing specifications and unit testing.

(3) Enhancing gamification with *story telling AI* for better emotional engagement and *advanded analytics* to provide insight on students' learning progress.

3 IMPRESS Innovations

This section presents the project progress so far.

Keeping students on the move with quizzes. Quizzes have great potential as teaching tools. They can enrich the presentation of a course's content, and foster participation in the class subject. Tools like Kahoot prospered because of this. Quizzes can be used in a class to raise attention to particular issues, e.g. by showing to the students what they do not know, hence, supporting self-awareness of knowledge and make students more receptive to new information. Quizzes can also be used to support revision of knowledge, for example, as a summary in the end of the class, and to evaluate students. Outside the class, quizzes can be a good self assessing tool for students and enhancing their learning process by supporting self-regulation of learning and providing quick feedback about their current state of readiness on their subjects.

We have developed a web-based tool to reduce teachers' effort in preparing quizzes. The tool, available in a GitHub repository: `https://github.com/socialsoftware/as-tutor`, allows users to search through a repository of questions and quizzes, and create new quizzes by re-using and re-purposing the materials they find. The tool also supports automatic generation of quizzes on students' (or teachers') requests, e.g. classified according to a set of topics. Produced quizzes can then be exported to gamified quiz tools, e.g. ARSnova, `https://arsnova.eu/`. The repository currently contains over 600 questions

and 80 quizzes, mostly on the subject of Software Architecture. A pilot in some of our courses is planned, after which the tool will be deployed open for the community. We plan to extend the tool with automatic classification of questions (for more accurate automatic quizz generation) and generation of post-quizz feedback for both students and teachers on the students' learning progress.

Training formalization skill with a game. Writing formal specifications is a skill that would greatly benefit students. Software with formal specifications can be verified, or at least tested, *automatically*, hence greatly improving its correctness assurance. Unfortunately, this skill is often left underdeveloped. The skill is not easy to master: it is easy to make mistakes, and training it can quickly become boring. In IMPRESS we experiment with a new game called FormalZ [13] to train the basic of writing formal specifications in the form of pre- and post-conditions. Unlike existing Software Engineering themed education games like Pex [15] and Train-Director-B [6], FormalZ takes a deeper gamification approach [1], where 'playing' is given a more central role. After all, what makes games so engaging is not merely the awarded scores and badges, but primarily the experience of playing them. Fig. 1 shows a screenshot of FormalZ.

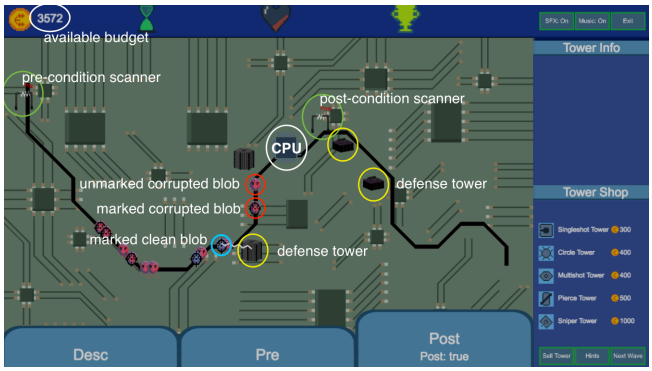

Fig. 1. *A screenshot of FormalZ. The game is to defend the CPU in the middle of the circuit board. The small red and blue blobs represent data coming to or leaving the CPU. Some of them might be corrupted. The user builds pre- and post conditions, and defense towers, trying to eliminate corrupted blobs. See also [13].*

FormalZ also takes a *Constructionism* approach [10]: just typing in formulas, which would be faster, is forbidden. Instead, the user constructs formulas by dragging and connecting blocks of electronic hardware components. The Constructionism theory believes that humans learn by *constructing* knowledge, rather than by simply copying it from the teacher. Framing the knowledge in terms of familiar physical objects, such as electronic components, plays a key role in this process, because the learner already has knowledge on how they work [5], which the learner then uses to construct the new knowledge in his mind. The theory was originally proposed by Papert and Harel [10] and was e.g. used in the programming language LOGO for teaching programming to children.

The initial reaction from our students have been encouraging [13], but more studies are needed to investigate the actual impact on the game's learning goal.

Teaching software testing through a competitive game. A further challenging activity in software engineering practice as well as education is testing a program for errors. In IMPRESS we explore improving the education of testing using Code Defenders, a game intended to engage students in the context of a Java object-oriented class under test and its test suite. In the game, *attackers* aim to introduce artificial bugs ("mutants") into the class under test that reveal weaknesses in the test suite, while *defenders* aim to improve the test suite by adding new tests. If a mutant program produces a different output for a test than the original program, then that mutant is detected by the test, and the defender who wrote the test scores points. If a mutant is not detected by any tests, then the attacker scores points. The number of points a mutant is worth depends on the number of tests it "survives", which further encourages players to create as subtle as possible mutants, and as strong as possible tests.

Code Defenders is implemented as a web-based game and is played by teams of students. The players are shown the source code of the Java class under test, with color highlighting to indicate the coverage of the defenders' test suite, and with bug-icons labelling the locations and status of the attackers' mutants. Attackers create mutants by editing the source code of the Java class, and defenders write JUnit tests using a code editor. A scoreboard breaks down the game's current score for each team and player.

We have studied player behavior in detail [14] and shown that players enjoy writing tests in the game more than as a regular developer activity. We have also applied Code Defenders in class and designed a software testing undergrad course around it [3]. Initial evaluation results suggest that Code Defenders supports students in achieving their learning objectives.

3.1 Advanced analytics

We have extended the analytics platform from the H2020 RAGE project[6] to adequate its functionalities to IMPRESS' needs, in particular to support different types of analytics generating educational activities [7]. These new developments allowed two approaches for analytics integration: *light* and *deep* integration.

Often, educational tools (like Kahoot!) provide a report that summarizes students interaction to some extent. In light integration the underlying educational tool it is not modified at all (e.g. because modification is not possible). RAGE Analytics is simply used on available analytics provided by the educational tool, e.g. to provide better or uniform visualisation across multiple tools.

In deep integration, the developers of the education tool need to integrate a tracker [11] into the tool, used to send out the user interaction information. As such, this approach can provide more fine grained analytics and to provide it live and is therefore the recommended integration approach. This was the approach selected for integration of the FormalZ game with RAGE Analytics, allowing us to collect all students interactions and to show them graphically to teachers, near real-time, in a single dashboard (Fig. 2). The analytics can also show how

[6] GitHub repository: https://github.com/e-ucm/rage-analytics

the students evolve their solutions, to give insight on their mental process in constructing the solutions.

Having all analytics in one place allowed us to provide an additional capability for teachers that want to have analytics of multiple heterogeneous activity (e.g. to track student progress during a longer period). This is facilitated through configurator to perform simple operations and weight of activities, so they can build new variables that can be included in class level dashboards [12].

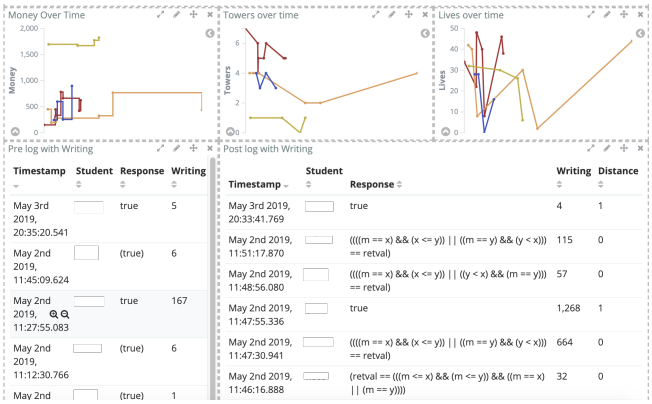

Fig. 2. *FormalZ analytics main dashboard.*

3.2 AI in IMPRESS

One of the use of AI for teaching is the generation and adaptation of learning content[2]. We are currently working on an AI module to create personalization features of the previously mentioned quiz tool we developed. It will work with the data that will be stored by the students performance on the quizzes to define student profiles and choose the best quizzes to enrich their learning experience.

AI can also improve the learning experience by adding a storytelling layer to the content. Stories are common in games and support meaning making and emotional engagement that foster learners motivation and learning[9]. We are developing storytelling components for the Code Defenders and FormalZ games by using the FAtiMA toolkit[7][8]. Our approach is to put the challenges presented by the games into a narrative, by including a character in the game that will talk to the players contextualizing the challenge that is given to the player(s) and presenting feedback on the performance. The toolkit facilitates the creation of such characters including mechanisms for the generation of personality and emotional responses, an authoring tool for character's behaviour, and integration through a REST API.

4 Conclusion

While the importance of Software Engineering courses is well acknowledged, creating engaging Software Engineering courses is very challenging. Much can

[7] https://fatima-toolkit.eu/

be improved through innovative use of modern technology. Along this line, IMPRESS has contributed innovations in gamification, and more can be expected before the project ends in 2020. Ultimately though, energizing Software Engineering education is not a challenge that a single project like IMPRESS can solve on its own. Community, and Industry, should also own the problem and commit to solving it.

References

1. Boyce, A.K.: Deep Gamification: Combining Game-based and Play-based Methods. Ph.D. thesis, North Carolina State Univ. (2014)
2. Brisson, A., Pereira, G., Prada, R., Paiva, A., Louchart, S., Suttie, N., Lim, T., Lopes, R.A., Bidarra, R., Bellotti, F., Kravcik, M., Oliveira, M.F.: Artificial intelligence and personalization opportunities for serious games. In: Proc. of the 8th Artificial Intelligence and Interactive Digital Entertainment Conf. (2012)
3. Fraser, G., Gambi, A., Kreis, M., Rojas, J.M.: Gamifying a software testing course with code defenders. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education. pp. 571–577. ACM (2019)
4. Hamari, J., Koivisto, J., Sarsa, H., et al.: Does gamification work? –a literature review of empirical studies on gamification. In: 47th Hawaii International Conference on System Sciences (2014)
5. Kafai, Y.B.: The Cambridge Handbook of the Learning Sciences, chap. Constructionism. Cambridge University Press (2005)
6. Korečko, Š., Sorád, J.: Using simulation games in teaching formal methods for software development. In: Innovative Teaching Strategies and New Learning Paradigms in Computer Programming, pp. 106–130. IGI Global (2015)
7. Martnez-Ortiz, I., Prez-Colado, I., Rotaru, D.C., Freire, M., Fernndez-Manjn, B.: From heterogeneous activities to unified analytics dashboards. In: IEEE Global Engineering Education Conference (EDUCON) (2019)
8. Mascarenhas, S., Guimarães, M., Prada, R., Dias, J., Santos, P.A., Star, K., Hirsh, B., Spice, E., Kommeren, R.: A virtual agent toolkit for serious games developers. In: Proc. Conf. on Computational Intelligence and Games (CIG). IEEE (2018)
9. Ohler, J.B.: Digital Storytelling in the Classroom: New media pathways to literacy, learning, and creativity. Corwin Press (2013)
10. Papert, S., Harel, I.: Constructionism. Ablex Publishing (1991)
11. Perez-Colado, I., Alonso-Fernandez, C., Freire, M., Martinez-Ortiz, I., Fernandez-Manjon, B.: Game learning analytics is not informagic! In: 2018 IEEE Global Engineering Education Conference (EDUCON) (2018)
12. Perez-Colado, I.J., Rotaru, D.C., Freire-Moran, M., Martinez-Ortiz, I., Fernandez-Manjon, B.: Multi-level game learning analytics for serious games. In: 10th Int. Conf. on Virtual Worlds and Games for Serious Applications (VS-Games) (2018)
13. Prasetya, I.S.W.B., Leek, C.Q.H.D., Melkonian, O., Tusscher, J.t., van Bergen, J., Everink, J.M., van der Klis, T., Meijerink, R., Oosenbrug, R., Oostveen, J.J., van den Pol, T., van Zon, W.M.: Having fun in learning formal specifications. In: Proc. 41st Int. Conf. on Software Engineering (ICSE). IEEE (2019)
14. Rojas, J.M., White, T.D., Clegg, B.S., Fraser, G.: Code defenders: crowdsourcing effective tests and subtle mutants with a mutation testing game. In: Proc. 39th Int. Conf. on Software Engineering. IEEE Press (2017)
15. Tillmann, N., de Halleux, J., Xie, T.: Pex for fun: Engineering an automated testing tool for serious games in computer science. Tech. rep., MSR-TR-2011-41 (2011)