Gamification of Software Modelling Learning

Alfa Yohannis, Dimitris Kolovos, and Fiona Polack

Department of Computer Science, University of York, York, United Kingdom
`ary506@york.ac.uk`

Abstract. Software modelling has a fundamental role in software engineering. However, it is perceived as relatively difficult for learners to develop the necessary abstraction skills to master the subject. Gamification is now flourishing as a popular strategy to engage learners. This research attempts to exploit gameful design as an innovative approach, used to create games that reinforce learners' mastery of software modelling by developing their abstraction skills. Our approach to gameful design brings together gamification development concepts such as design lenses and intrinsic skill atoms, and pedagogical design principles from several learning theories and models. The research follows the Design Science Research Methodology and exploits Model-Driven Engineering best practices. To date, the research has been developing an early prototype for learning software modelling through gamification. For evaluation, the effects of the artefact will be measured using longitudinal controlled experiments.

Keywords: software modelling, gamification, learning, abstraction

1 Introduction

Software modelling is commonly perceived as a difficult subject since it requires a mastery of abstraction [1]. However, this subject has a fundamental and crucial role in software engineering education and practice. Failure to master this topic will affect the students abstraction capability in analysing and designing a real-world software. Weak software modelling will likely cause software engineering students to face difficulties completing their degrees, as most of the software engineering related subjects have a sense of intrinsic abstraction problems [2]. Students' perception of software modelling will affect their attitude towards software engineering today and their career paths in the future.

The problems of learning appropriate abstraction skills for software modelling is similar to problems in mathematics, where most of the concepts can only be accessed through symbolical representations [3]. Abstraction also requires the students to perform information hiding, generalisation, approximation or reformulation, leaving out the irrelevant aspects but keeping the relevant ones, or separation from the concrete reality [4]. To overcome these challenges, we need to put more effort into designing delivering software modelling training, developing a more concrete and motivating presentation which can engage students and facilitate deep learning.

In recent years, the use of games or game elements for serious purposes other than leisure has drawn lots of attention. Gamification [5] and Serious Games [6] have been viewed as solutions to motivational problems that emerge when users are required to engage in activites that they perceive as boring, irrelevant or difficult, e.g. Learning sorting algorithms [7] or C-programming [8].

Therefore, the purpose of this research is to investigate and develop a gamification design framework that systematically and semi-automatically drives gamification design to produce better designed software modelling games. More precisely, this research aims to answer the following research questions that are derived from the purpose of this study:

1. Which processes, aspects, principles, or components of software modelling and their teaching and learning practices should be included?
2. What types of game elements and their roles that can deliver software modelling learning best?
3. What kind of orchestrating framework is needed to design the interaction between, software modelling and game elements to produce a better software modelling gamification?
4. To what extent does gamification of software modelling improve engagement and motivation and improve learners performance?

Due to the limitation of number of pages, this paper does not cover some interesting aspects of this research, such as the architecture of the artefact and the validation methods applied to evaluate models created by learners.

2   Related Works

Several approaches attempt to bring software modelling into a more concrete presentation that can be easily understood by learners, ranging from didactic learning [9], modelling tools utilization [10], alternative communication channels and the use of modelling language [11], immersive visual modelling through virtual environment [12], software design studio [13], project-based approach [14], to code generation investigation [15]. However, most of the approaches have weaknesses in motivating learners to engage continuously, frequently, and actively to learn software modelling, which is the important aspect impacting greatly on learning [16]. To address lack of engagement, we investigate game-based learning, to learn or teach software modelling. This method provides students with a new way of learning software modelling, which is not only interactive but also engaging enough to keep them learning continuously.

The use of game elements for a purpose other than leisure is called gamification [5]. Gamification design is still an ongoing challenge [17], and, to date, there is no gamification design framework that particularly structures the design of software modelling gamification; a framework that integrates game specific domain into software modelling. Hence, this research aims to develop a gamification design framework of software modelling learning.

Most of the software related gamification studies available are related to software engineering in a larger context or to other aspects of software engineering, such as software implementation and project management, rather than software modelling in particular [18]. After the literature exploration, only four works have been identified applied gamification for software modelling. None of these works addresses software modelling learning in general — how learners can learn abstraction in modelling. Instead, they address specific topics such as activity diagramming [19], coupling and cohesion [20], and enterprise architectures [21] [22]. Most of the works also cover pedagogical aspect superficially or not at all and validation is restricted to a very limited number of users.

3   Research Methods

Since the output of this work is designed artefacts, we decided to utilise the Design Science Research Methodology (DSRM) [23] as our umbrella methodology. DSRM is selected because it provides a comprehensive high-level conceptual framework how to undergo a full-cycle research process. It also provides six activity guidelines for understanding, developing, executing, and evaluating design artefacts. The six activities are (1) problem identification and motivation activity, (2) definition of objectives for a solution activity, (3) setting of targets for a solution activity, (4) design and development activity, (5) demonstration and evaluation activities, and (6) communication.

The high-level characteristics of DSRM mean that we can employ other research methods as the sub-methods in each activity. For example, we employ interviews, literature reviews, and discussion with experts as our methods in problem identification and motivation activity, as well as using Deterding's lens of intrinsic skill atoms [24] to produce a gameful design in the design and development activity.

Table 1: Design lenses (game elements) applied in the gamification design.

Lenses	Elements
Challenges	Onboarding, scaffolded challenge, varied challenge
Goals and Motivation	interim goals, intrinsic rewards
Actions and Object	bite sized actions, limited choices, microflow, underdetermination, sensual
Feedbacks	Immediate, juicy, actionable, appeal to motives, glanceable, varied, graspable progress

4   Gamfication Design

To deliver a gameful experience while playing with the artefact, game elements have to be embedded into the artefact's design. Thus, Design Lenses and Skill

Atoms [24] are utilised to determine the required game elements as well as the game mechanics. The design of the levels of the game has a gradually increased difficulty as well as variety in its challenges, to expose learners to different kinds of domains, models, and diagrams. The onboarding game element is also planned to be implemented into the artefact to help learners familiar with the control system and the flow of the game. The software modelling game design will also utilise templates to help learners building models without having to start from scratch. The foundation model is already given. They only need to continue the model the meets certain objectives. The design of the cycle of the game mechanics is derived from Intrinsic Skill Atoms [24], which have six components that should be defined: motivation, goals, actions and objects, challenges, rules, and feedbacks.

Table 2: Skill Atoms applied in the gamification design.

Atoms	Description
Motivation	Master the modelling (solve problem, able to construct model)
Goals	Create models that satisfy requirements
Actions and Objects	Construct model using diagrams, elements of a diagram, keywords, hints
Challenges	Satisfy requirements, meet objectives
Rules	Inherent rules in every model diagram, constraints, objectives
Feedbacks	completed objectives, model metrics, and motivating words

5   Visual Modelling Editor

To support developers in the design and customisation of gamification for software modelling learning — the integration of game elements into its learning activities — at a high level of abstraction and so as to automatically build the game generator, this research is developing a visual modelling editor using Eugenia, a GMF-based graphical model editors [25]. Currently, developers can use the editor to design the gamification of software modelling learning by defining its flows, levels, challenges, and objectives. In the future, this research plans to add more features, such as user-customised types of modelling and diagrams.

6   Evaluation

The evaluation of the game artefact will use controlled experiments. The respondents, software modelling students, will be divided into two groups, a control group and experimental group. The control group will learn software modelling using traditional methods while the experimental group will learn with support from the artefact. After some time learning, both groups will be given problems
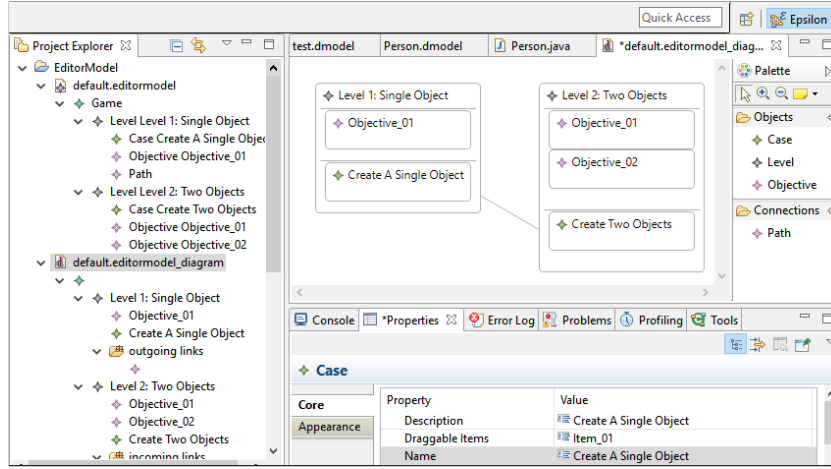
Fig. 1: Game editor to automatically generate the game.

to solve. Their performance will be measured by their ability to solve the problems. To anticipate the order effect, they will be asked to change their roles, from the control group to the experimental group and vice versa. After that, their ability will be tested again using similar problems. After the experiment, the significance of their performance will be calculated. To evaluate the generality of the effects of the artefact, a longitudinal experiment is also considered as well as undertaking the experiments in different countries and universities. Furthermore, The evaluation of the gamification design framework is similar to the game artefact's evaluation except for the respondents and problems will be software developers and software modelling gamification construction.

The controlled experiment is limited to measuring the significance of the design artefact. It cannot provide understanding to explain why learning software modelling supported with the artefact is better or worse than the traditional one. Consequently, surveying with questionnaires or interviews might be conducted to investigate the underlying variables or processes. Structural equation modelling [26] is also an option if measuring the effects of the identified underlying factors is required. An alternative method for understanding of underlying variables and processes is through investigating the artefact's event logs using data mining or machine learning techniques.

7   Conclusion

This paper explains our research motivation and problem statements, proposed solution and objectives, research methods, the current progress of the design and development of the artefact, and the evaluation plan. So far, this research is focusing its work on software modelling learning. In the future, this research plans to address metamodelling and model transformation learning.
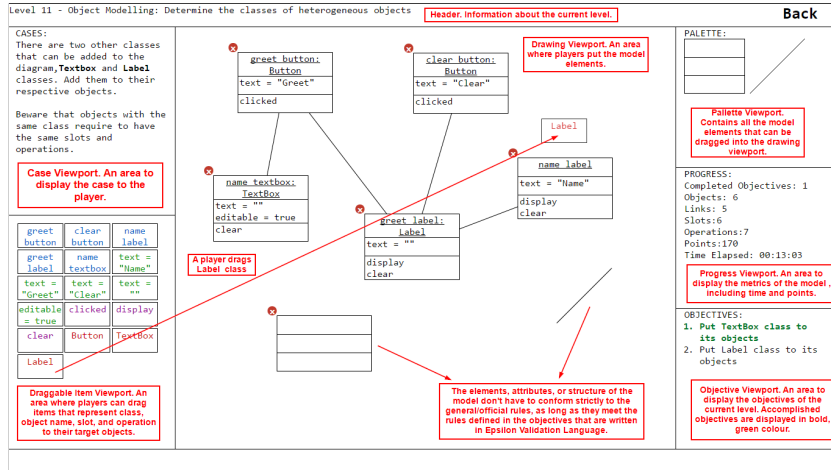
Fig. 2: The game's display.

References

1. J. Börstler, L. Kuzniarz, C. Alphonce, W. B. Sanders, and M. Smialek, "Teaching software modeling in computing curricula," in *Proceedings of the final reports on Innovation and technology in computer science education 2012 working groups*, pp. 39–50, ACM, 2012.
2. J. Kramer, "Is abstraction the key to computing?," *Communications of the ACM*, vol. 50, no. 4, pp. 36–42, 2007.
3. R. Duval, "A cognitive analysis of problems of comprehension in a learning of mathematics," *Educational studies in mathematics*, vol. 61, no. 1-2, pp. 103–131, 2006.
4. L. Saitta and J.-D. Zucker, *Abstraction in artificial intelligence and complex systems*, vol. 456. Springer, 2013.
5. S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: defining gamification," in *Proceedings of the 15th international academic MindTrek conference*, pp. 9–15, ACM, 2011.
6. D. R. Michael and S. L. Chen, *Serious games: Games that educate, train, and inform.* Muska & Lipman/Premier-Trade, 2005.
7. A. Yohannis and Y. Prabowo, "Sort attack: Visualization and gamification of sorting algorithm learning," in *the 7th VS-Games*, pp. 1–8, IEEE, 2015.
8. M.-B. Ibanez, A. Di-Serio, and C. Delgado-Kloos, "Gamification for engaging computer science students in learning activities: A case study," *IEEE Transactions on Learning Technologies*, vol. 7, no. 3, pp. 291–301, 2014.

9. S. Moisan and J.-P. Rigault, "Teaching object-oriented modeling and uml to various audiences," in *International Conference on Model Driven Engineering Languages and Systems*, pp. 40–54, Springer, 2009.

10. S. Akayama, B. Demuth, T. C. Lethbridge, M. Scholz, P. Stevens, and D. R. Stikkolorum, "Tool use in software modelling education.," in *EduSymp@ MoDELS*, 2013.

11. M. Brandsteidl, K. Wieland, and C. Huemer, "Novel communication channels in software modeling education," in *International Conference on Model Driven Engineering Languages and Systems*, pp. 40–54, Springer, 2010.

12. B. J. Neubauer and J. D. Harris, "Immersive visual modeling: potential use of virtual reality in teaching software design," *Journal of Computing Sciences in Colleges*, vol. 18, no. 6, pp. 142–150, 2003.

13. J. Whittle, C. N. Bull, J. Lee, and G. Kotonya, "Teaching in a software design studio: Implications for modeling education," in *CEUR Workshop Proceedings*, vol. 1346, pp. 12–21, CEUR-WS, 2014.

14. R. Szmurło and M. Śmiałek, "Teaching software modeling in a simulated project environment," in *International Conference on Model Driven Engineering Languages and Systems*, pp. 301–310, Springer, 2006.

15. A. Schmidt, D. Kimmig, K. Bittner, and M. Dickerhof, "Teaching model-driven software development: revealing the great miracle of code generation to students," in *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148*, pp. 97–104, Australian Computer Society, Inc., 2014.

16. T. L. Naps, "Jhavé: Supporting algorithm visualization," *IEEE Computer Graphics and Applications*, vol. 25, no. 5, pp. 49–55, 2005.

17. S. Deterding, S. L. Björk, L. E. Nacke, D. Dixon, and E. Lawley, "Designing gamification: creating gameful and playful experiences," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pp. 3263–3266, ACM, 2013.

18. O. Pedreira, F. García, N. Brisaboa, and M. Piattini, "Gamification in software engineering–a systematic mapping," *Information and Software Technology*, vol. 57, pp. 157–168, 2015.

19. O. Richardsen, "Learning modeling languages using strategies from gaming," Master's thesis, Norwegian University of Science and Technology, Norway, 2014.

20. D. R. Stikkolorum, M. R. Chaudron, and O. de Bruin, "The art of software design, a video game for learning software design principles," *arXiv preprint arXiv:1401.5111*, 2014.

21. J. Groenewegen, S. Hoppenbrouwers, and E. Proper, "Playing archimate models," in *Enterprise, Business-Process and Information Systems Modeling*, pp. 182–194, Springer, 2010.

22. D. Ionita, R. Wieringa, J.-W. Bullee, and A. Vasenv, "Tangible modelling to elicit domain knowledge: an experiment and focus group," in *International Conference on Conceptual Modeling*, pp. 558–565, Springer, 2015.

23. K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

24. S. Deterding, "The lens of intrinsic skill atoms: A method for gameful design," *Human–Computer Interaction*, vol. 30, no. 3-4, pp. 294–335, 2015.

25. D. S. Kolovos, A. García-Domínguez, L. M. Rose, and R. F. Paige, "Eugenia: towards disciplined and automated development of gmf-based graphical model editors," *Software & Systems Modeling*, pp. 1–27, 2015.

26. J. F. Hair Jr, G. T. M. Hult, C. Ringle, and M. Sarstedt, *A primer on partial least squares structural equation modeling (PLS-SEM)*. Sage Publications, 2016.