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 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 can be used to systematically and semi-automatically drive 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?

2 Related Work

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 investigte 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. They are the

works of Groenewegen et al. [19], Stikkolorum et al. [20], Richardsen[21], and Ionita et al. [22]. 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 +CITATION, coupling and cohesion +CITATION, and enterprise architectures +CITATION. Most of the works also cover pedagogical aspect superficially or not at alland 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.

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.

4.1 Design Lenses

Deterding et al. [24] define various design lenses, to focus analysis of the game requirements. Here, we outline those relevant to our gamification design, and outline how they are applied to the modelling tasks.

Challenge Lens. The design of the levels of a 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.

Goal and Motivation Lens. The design implements interim goals and intrinsic rewards to motivate learners to engage with the artefact. For software modelling, every form of modelling (e.g. object modelling, collaboration modelling) has several stories. A story represents a specific case study to introduce learners to certain problems in specific domains. Every story consists of several levels, and every level has one or more objectives that a student needs to accomplish to complete the level. The level also might be a continuation of its previous level, thus give the learners a sense of step-by-step progression to complete the domain problems. In every story or level, new concepts — related to their type of modelling—or a combination between the old and new concepts might be introduced to learners. Therefore, as the learners progressing, their competence in modelling are being developed — the intrinsic rewards for the learners.

Action and Object Lens. A real world problem can be very exhaustive and time-consuming. Thus, the extraneous activities that are not relevant to the core concepts that are being taught should be removed. As a result, learners will be more focused on the main concepts. For that reason, game elements like bite-sized actions (e.g. drag and drop), limited choices (i.e. only limited items can be dragged), and microflow (i.e. put the right element to its right place) are selected to facilitate learners in performing the core activities. Likewise, an element of underdetermination is used to provoke learners' creativity since most of the time there is no single correct model to represent a model; the models can vary. Attractive design is also significant since it draws learners' attention to interact with the artefact.

Feedback. The system should give immediate, glanceable, actionable, and graspable feedback to keep learners on track and monitor the state of their games. Moreover, the feedback should be designedmto be interesting and varied, and to appeal to the learners' motives, in order to keep their attention.

4.2 Intrinsic Skill Atoms

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. The main motivation for engaging with our game artefact is to master software modelling, which means that they will be able to construct models and solve model-related problems. Forinstance, students should be able to build an object modelfor a sign-in screen. Moreover, The goal of their modelling activities is to meet the given objectives whilst obeying the rules. For example, learners are asked to construct an object diagram of a button that when pressed will clear the text of a textbox; there will be two objects (a button object and a textbox object), but the diagram will not be complete if there is no link that connects them to represent their relationship.

As in most games, learners are required to take some actions to manipulate some objects in order to achieve the modelling goals. The artefact requires learners to perform dragging and dropping and connecting diagram elements

(e.g. objects and links) to construct a model. There are also draggable items — containing the names or values of slots, actions, and classes of objects — that must be associated to the appropriate diagram elements. The learners are confronted with some challenges in the form of problem-solving cases to make the modelling activity more exciting. For example, an animation that shows how a login page works is displayed to the learners, and they are asked to model it.

All the models created have to adhere to the rules that are inherent in the types of the models. However, the game does not always have to be strict with the rules since it could cause the learners demotivated. As an illustration, a link in an object diagram should connect two objects, one on each end. However, this rule can be neglected as long as the rule is not a crucial part of the topic that is being addressed. Finally, feedbackis required to help the learners keep track of every action they made, to inform them of their progress, and to motivate them in their ups and downs.

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 andso 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 arfefacts and framework will use controlled experiments. The respondents 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 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.

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+CITATION 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.

Comment (FACP): this is a nice evaluation of a game... but what about an evaluation of the gamification framework... wont this require getting developers to create new games for different modelling languages?

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 evaluation plan have been explained. Nevertheless, some interesting aspects of this research have not been covered in this paper due to the limitation of space, such as the architecture of the artefact and the validation methods applied to evaluate models created by learners. 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.

Acknowledgments. Thanks to York Masters students who participated in our preliminary surveys. This research is supported by *Lembaga Pengelola Dana Pendidikan Indonesia* (Indonesia Endowment Fund for Education).

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 - ITiCSE-WGR '12*, (New York, New York, USA), p. 39, ACM Press, jul 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," 2006.
4. L. Saitta and J.-D. Zucker, "02 - Abstraction in Different Disciplines," in *Abstraction in Artificial Intelligence and Complex Systems*, ch. 2, pp. 11–47, Springer New York, 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: Envisioning future media environments*, pp. 9–15, ACM, 2011.
6. D. R. Michael and S. L. Chen, "Serious Games: Games That Educate, Train, and Inform," *Education*, vol. October 31, pp. 1–95, 2005.
7. A. Yohannis and Y. Prabowo, "Sort Attack: Visualization and Gamification of Sorting Algorithm Learning," in *VS-Games 2015 - 7th International Conference on Games and Virtual Worlds for Serious Applications*, 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, M. Brandsteidl, B. Demuth, K. Hisazumi, T. C. Lethbridge, P. Stevens, and D. R. Stikkolorum, "Tool use in software modelling education: state of the art and research directions," in *the Educators' Symposium co-located with ACM/IEEE 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013)* (T. C. Lethbridge and P. Stevens, eds.), (Miami, U.S.A.), 2013.
11. M. Brandsteidl, K. Wieland, and C. Huemer, "Novel communication channels in software modeling education," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6627 LNCS, pp. 40–54, 2011.
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. Szmurlo and M. Śmialek, "Teaching Software Modeling in a Simulated Project Environment," *Lecture Notes in Computer Science*, vol. 4364, pp. 301–310, 2007.
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," *CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13*, p. 3263, 2013.
18. O. Pedreira, F. García, N. Brisaboa, and M. Piattini, "Gamification in software engineering - A systematic mapping," in *Information and Software Technology*, vol. 57, pp. 157–168, Elsevier, 2015.
19. J. Groenewegen, S. Hoppenbrouwers, and E. Proper, "Playing ArchiMate models," 2010.
20. D. R. Stikkolorum, M. R. V. Chaudron, and O. de Bruin, "The Art of Software Design, a Video Game for Learning Software Design Principles," jan 2014.
21. O. Richardsen, "Learning Modeling Languages Using Strategies from Gaming," 2014.
22. D. B. Ionita, R. Wieringa, J.-w. Bullee, and A. Vasenev, "Tangible Modelling to Elicit Domain Knowledge: An Experiment and Focus Group," vol. 9381, pp. 558–565, 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.

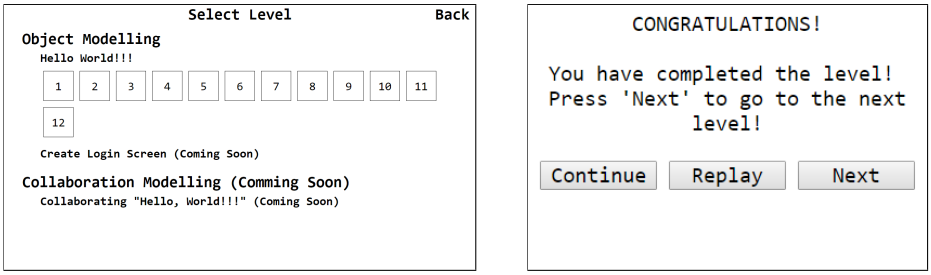
8 Appendix A: Tables and Images

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

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



(a) Level Selection (b) Positive Reinforcement

Fig. 1: Game and learning elements embedded into the artifact.

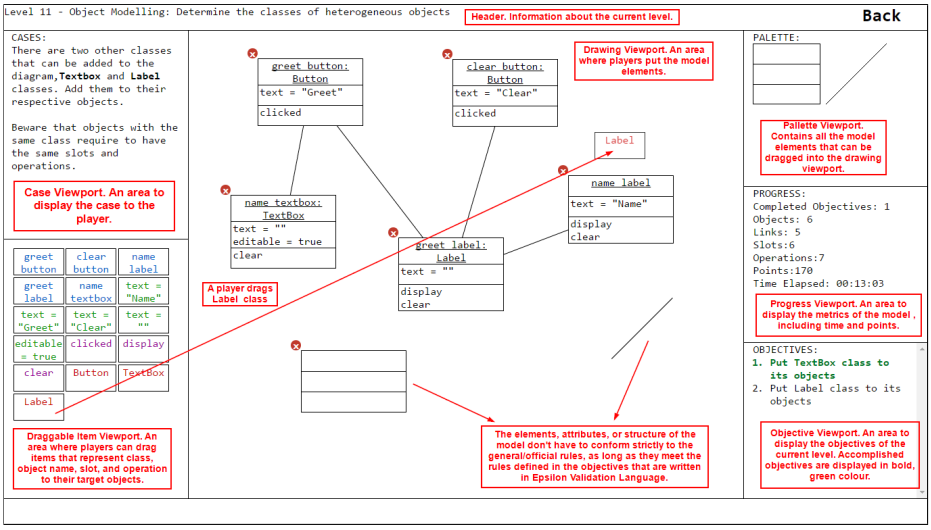


Fig. 2: The game's display.

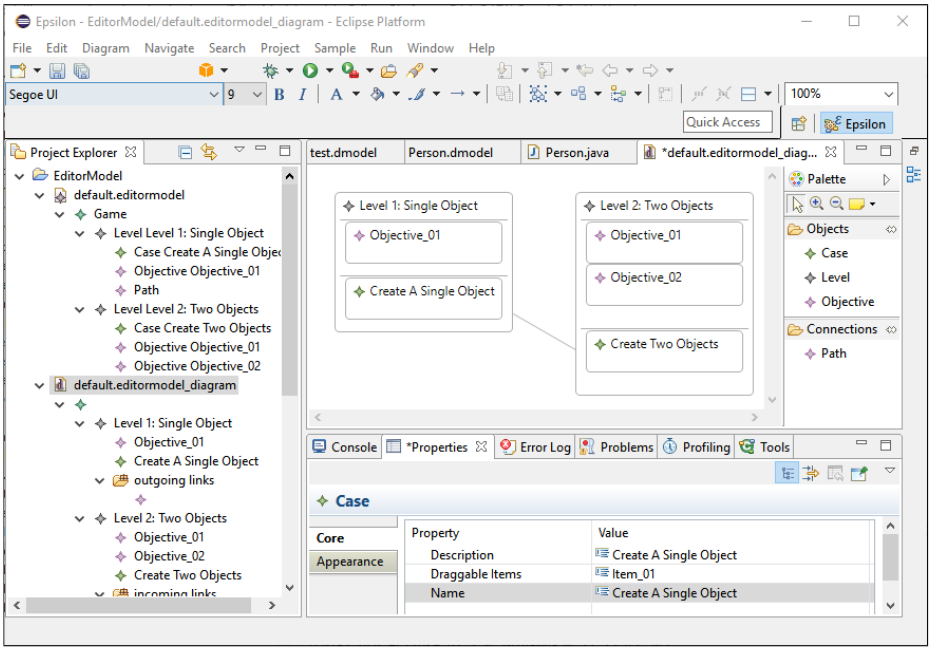


Fig. 3: Game editor to automatically generate the game.