Gamification of Software Modelling Learning

Alfa Yohannis

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, this subject is relatively perceived difficult since learners requires to develop abstraction skill to master the subject. On the other side, gamification has been flourishing as a new popular strategy to engage learners in learning activities. This research attempts to exploit gameful design as an innovative solution to reinforce learners to master software modelling by developing their abstraction skills. Drawing out the gameful design from the Design Lenses and Intrinsic Skill Atoms notions, the pedagogical design from several learning theories and models, as well as being governed by the Design Science Research Methodology and Model-Driven Engineering learning best practices, this research has been developing an early prototype for its software modelling learning gamification. For evaluation, the effects of the artefact are going to be measured using longitudinal controlled experiments.

Keywords: software modelling, gamification, learning, abstraction

1 Introduction

Software modelling is commonly perceived as an difficult subject since it requires a great deal of abstractions [1]. However, this subject has a fundamental and crucial role in software engineering. Failure to master this subject will affect the students abstraction capability in analysing and designing a real-world software. Less proficiency in software modelling will likely cause software engineering students facing difficulties completing their degrees, as most of the software engineering related subjects have a sense of intrinsic abstraction problems [2]. Their perception of software modelling will affect their attitude towards software engineering today and their career paths in the future.

For new software engineering students, software modelling is commonly perceived as subjects that are difficult to learn. Software, by its nature, is abstract and complex that demand loads of abstraction to model it. This problem is similar to problems in mathematics where much of the concepts can only be accessed through symbolical representations [3]. Abstraction also means requiring the students to perform information hiding, generalization, approximation or reformulation, leaving out the irrelevant aspects but keeping the relevant ones, or separation from the concrete reality [4]. In order to overcome these challenges, we need to put more efforts on delivering software modelling in a more concrete and motivating presentation which can engage and ease students to learn it.

On the other hand, the use of games or game elements for serious purposes other than leisure has drawn lots of attentions. Gamification [5] and Serious Games [6] have been viewed as solutions to solve motivational problems that emerge when users engage in boring, irrelevant, difficult activities, 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 drive gamification design to produce better-design software modelling games. More precisely, this research aims to answer the following research questions that are derived from the purpose of this research:

1. Which processes, aspects, principles, or components of software modelling and their teaching and learning practices that are essential to take into account?
2. What types of game elements and their roles that can deliver software modelling at best?
3. What kind of framework that orchestrates, design the interaction between, software modelling and game elements to produce a better software modelling gamification?
4. To what extend gamification of software modelling better engage, motivate, and improve learners performances?

2 Related Works

Several approaches have been conducted to bring software modelling into a more concrete presentation that can be easily understood by learners, ranging from didactic learning [9], modeling 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 to impact greatly on learning [16]. This condition then elevates game as one approach, a.k.a. game-based learning, to learn or teach software modelling. This approach provides students 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]. Regardless gamification design is still an ongoing challenge [17], it is an opportunity for research that up today there is still 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.

Most of the gamification studies available are dominantly relates to software engineering in a larger context or 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]. All of the works do not address software modelling learning in general—how learners can learn abstraction in modelling—and they only address very specific topics or area in software modelling, such as activity diagram, coupling and cohesion, and enterprise architectures. Most of the works also only touch the pedagogical aspect superficially or not at all and are validated in a very limited number of respondents.

3 Research Methods

Since the output of this work is design 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 artifacts. The six activities are (1) problem identification and motivation activity, (2) define objectives for a solution activity, (3) define objectives for a solution activity, (4) design and development activity, (5) demonstration and evaluation activities, and (6) communication.

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

4 Gamfication Design

In order to deliver a gameful experience while playing with the artefact, game elements have to 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

Challenge Lens. Figure 1a shows the level design of the artefact. The design of levels has a gradually-increased difficulty as well as variety in its challenges to expose learners to different kinds of domains, models, and diagrams. Orboarding 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 design will also utilise templates to help learners building models without having to start from scratch. The foundation model already given. They only need to continue the model the meet certain objectives.

Goal and Motivation Lens. To motivate learners engage with the artefact, the design implements interim goals and intrinsic rewards. Every type of modelling (object modelling, collaboration modelling, etc.) has several stories. A story represent a specific case study to introduce learners to certain problems in specific domains. Every story consist of several levels and every level has one or more objectives that a learner need to accomplish in order 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. This is the intrinsic rewards for the learners; building their competence in modelling.

Action and Object Lens. Modelling 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 to perform the core activities. Likewise, underdetermination element is also selected to provoke learners' creativity since most of the time there is no single correct model to represent a model; the models can be vary. Attractive design are also significant since it draws learners attention to interact with the artefact.

Feedbacks. To keep learner' keep on track and monitor the state of their games, the system should give immediate, glanceable, actionable, and graspable. Moreover, the feedbacks should be designed interesting, varied, and appeal to motives to draw their attentions, not being boring, and motivating.

4.2 Intrinsic Skill Atoms

The design of the cycle of the game mechanics that the learners will be playing during engaging with the artefact is derived from Intrinsic Skill Atoms, which have six components that should be defined: motivation, goals, actions and objects, challenges, rules, and feedbacks. The main motivation engaging with artefact is to master software modelling, which means that they will be able to construct models and solve model-related problems. For an instance, able to construct object modelling of a sign-in screen problem. Moreover, The goal of their modelling activities is to meet the given objectives but still in the corridor of the rules. As an example, learners are asked to construct an object diagram of an button that when pressed will clear the text of a textbox. Of course there will be two objects (button object and textbox object), but the diagram will not be complete if there is no link that connects them to represent their relationship.

In order to reach the goals, learners are required to do some actions to manipulate some objects like many other games. The artefact requires learners to perform dragging, dropping, and connecting diagram elements (e.g. objects

and links) to construct a model. They are also required to drag and drop some draggable items—contains the names or values of slots, actions, and classes of objects—into the appropriate diagram elements. To make the modelling activity more exciting, the learners are confronted with some challenges in the form of problem-solving cases. For an 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 generally have to adhere with the rules that are inherent to the types of the models. However, the rules do not always have to be strict with the rules since it could causes the learners demotivated. As an illustration, a link in an object diagram should connect two objects, one on each ends. However, this rule can be neglected as long as the rule is not a crucial part of the topic that are being addressed. Finally, feedbacks are required to keep the learners keep track on every actions they made, to be informed of their progression, and to motivate them in their ups and downs.

4.3 Modelling Editor

5 Evaluation

Controlled experiment will be applied to evaluate the effect of the design artefact. The respondents be divided into two groups control control group and experimental group. Control will learn software modelling using traditional method while experimental group will learn with the support from the artefact. After some time learning, they will be given problems to be solved. Their performance will be measured by their ability to solve the problems. In order to anticipate the order effect, they will be asked to change their roles, from control group to experimental group and vice versa, and then their ability will be tested again using similar problems. After the experiment, the significance of their performance to will be calculated. To evaluate the generality of the artefact's effect, longitudinal experiment is also considered as well as undergoing the experiment in different countries and universities.

Nonetheless, the controlled experiment is limited only to measure the significance of the design artefact and no understanding is developed 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 also an option if measuring the effects of the identified underlying factors is required. The other alternative method to gain understanding of underlying variables or processes is through investigating the artefact's event logs using data mining or machine learning techniques.

6 Conclusion

Acknowledgments. We would like to thank our respondents that participated in our preliminary interview. 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.

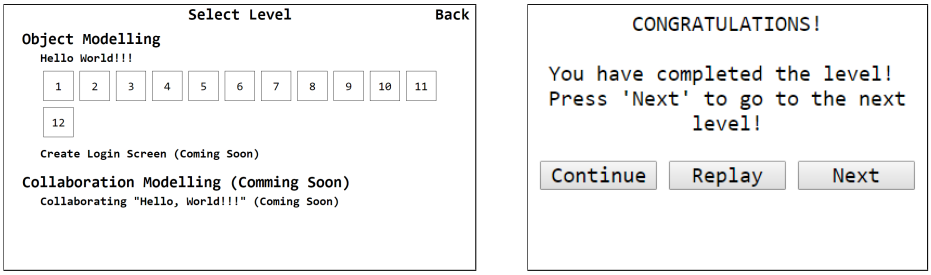
7 Appendix A: Tables and Images

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.

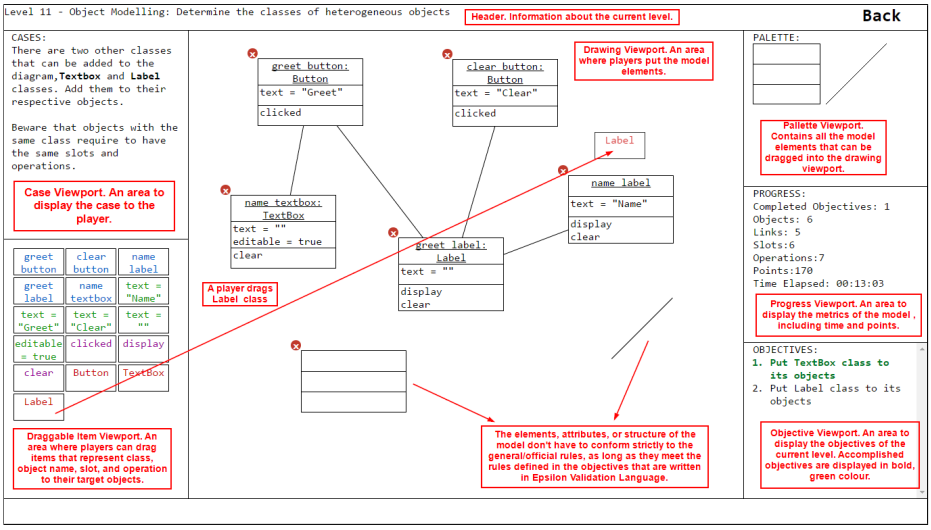


Fig. 2: The game's display.

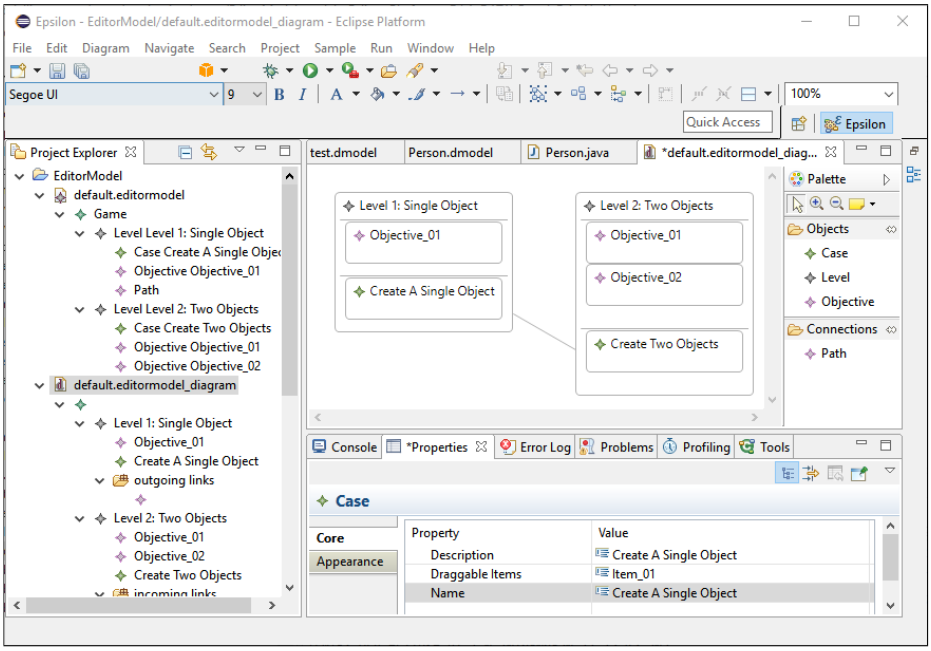


Fig. 3: Game editor to automatically generate the game.