Gamification of Software Modelling Learning

Alfa Yohannis

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

Abstract. The abstract should summarize the contents of the paper and should contain at least 70 and at most 150 words. It should be written using the *abstract* environment.

Keywords: software modelling, gamification, learning

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] or the process of transforming a less gameful entity into a more gameful one [17] [18] [19]. Regardless gamification design is still an ongoing challenge [20], it is an opportunity for research that up today there is still no gamification design framework that particularly address how to guide the design of software modelling gamification; a framework that guides how to integrate game specific domain into software modelling. Therefore, 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 [21]. Several studies that apply gamification specifically for software modelling are the works of Stikkolorum et al. [22], Ionita et al. [23], Groenewegen et al. [24], and Richardsen [25]. These works are selected because they develop artefacts based on the gamification approach, apply them to software modelling, and validate the results.

4.4.1 Stikkolorum et al.(Stikkolorum et al., 2014) Contents. Cohesion, coupling, information hiding, and modularity in software design. Goals. Learning and applying software design principles. Techniques. Looking for a solution that provides balance between coupling and cohesion by using the toolbox to draw classes, methods, attributes, and relationships between them. Game Elements. Puzzle game, game levels, visual and audio feedbacks, progress indicator, level unlocking, choice of path, multiple solutions, scoring. Pedagogy. Blooms taxonomy is mentioned but never discussed. Outcomes. Users started to talk regarding classes, methods, and associations instead of boxes, blocks, and lines, indication unconscious learning. Challenges. Determining the scores because the there are more than one solutions. Coupling Between Classes (CBO) is used to determine the coupling. Cohesion is measured by comparing all itemsattributes, methods, class namein a class have similar keywords. Information hiding and modularity is evaluated using general design patterns. A script check if the user applied the elementsmethods, attributes, classesin a way that fits with design patterns. Validation. Conduct user test and using 'think aloud' method, asking users to tell their thoughts while using the game.

Groenewegen et al.(Groenewegen et al., 2010)[24] Contents. Enterprise architecture. Goals. Architecture model understanding and validation. Techniques. Exploring the model step by step, element by element according to the given rules is used. This method provides a progressive user experience. Therefore, it can improve user understanding. The game proposed is more playable, more freedom to try and explore, and no explicit rewards given. Game Elements. Cards, explorable board-game, rules. Pedagogy. No clear pedagogical aspect discussed. Outcomes. The user can understand the model better rather than by merely looking at the model so they can give argument whether the model is valid or not based on their existing knowledge. Challenges. Preparing the game by translating implicit knowledge to explicit knowledge of the modela gap of knowledge between the modeller and the reader. Domain knowledge is required. Lack of domain knowledge will make the model less understandable and the user cannot validate the model. Validation. Testing the model to 7 respondents and then interviewing them.

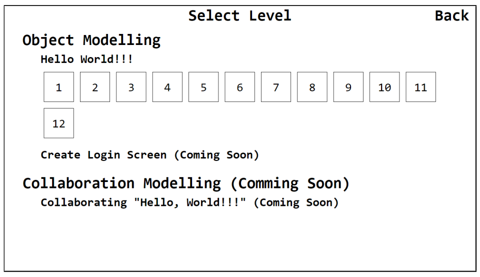
4.4.3 Ionita et al.(Ionita et al., 2015) Contents. Domain modelling (information security). Goals. Improve learnability and participation. Techniques. Develop a socio-technical modelling language (TREsPASS) and map them toward tangible representation. The tangible representations will increase the familiarity and understandability of the model, therefore increase awareness and involvement. Game Elements. Familiar, tangible representation, such as Lego characters, board-game metaphor, rules. Pedagogy. Theory of constructionism, cognitive load, cognitive fit. Outcomes. The experimental group performs better in learnability, efficiency, correctness, and satisfaction. Experts and professionals indicated that the tangible model might be useful for less technical domain experts and different types of stakeholders to be more participative and contributive in the early stages of architecture modelling. Challenges. Mapping the socio-technical modelling language to tangible model. Validation. Comparing performance between control and experimental group was performed. Conducting focus group with experts and professionals to assess its functionality.

4.4.4 Richardsen (Richardsen, 2014) Contents. Activity diagram. Goals. Learning activity diagram in the context of Reactive Block tool and comparing between traditional interactive tutorial and game-like tutorial. Techniques. Arranging UML activity diagram to control the behaviour of a game. Game Elements. Game, levelling, character, challenge, items, rule. Pedagogy. No explicit pedagogical aspect discussed. Outcomes. There is no significant different between the traditional interactive tutorial and the game-like tutorial on their performances. However, the game-like tutorial is more engaging. Challenges. Controlling game from activity diagram in Reactive Block Environment (Eclipse based) is difficult since Eclipse is difficult for a first-time user. Validation. Conducting user testing with three users. Think Aloud method was used for observation. After that, questionnaires were given and an interview was conducted.

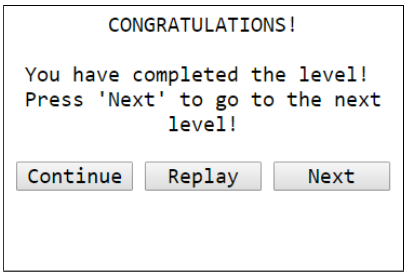
3 Research Methods

Since the output of this work is design artefacts, we decided to utilise the Design Science Research Methodology (DSRM) [26] 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 [27] to produce a gameful design in the design and development activity. As information, our work is still in the design and development activity and there are three activities remaining to work on. We use the first three activities of DSRM as our guidelines to report our in-progress accomplishment.


Select Level
Back
Object Modelling
Hello World!!!
1 2 3 4 5 6 7 8 9 10 11 12
Create Login Screen (Coming Soon)
Collaboration Modelling (Comming Soon)
Collaborating "Hello, World!!!" (Coming Soon)

(a) Level Selection


CONGRATULATIONS!

You have completed the level! Press 'Next' to go to the next level!

Continue Replay Next

(b) Positive Reinforcement

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

4 Problem Identification and Motivation Activity

5 Define Objectives and Evaluation Activity

6 Design and Development Activity

6.1 Game Design

6.2 Architecture

6.3 Model Validation

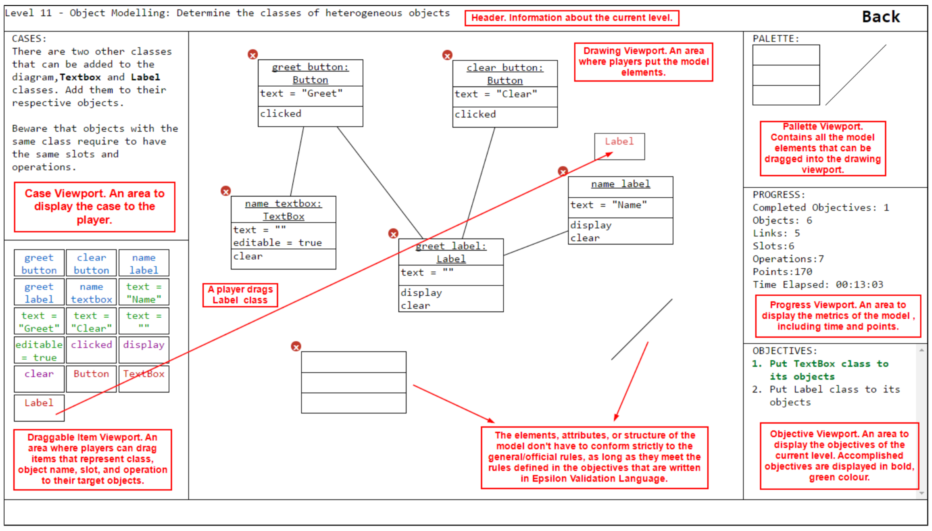
6.4 Modelling Editor

7 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.


Level 11 - Object Modelling: Determine the classes of heterogeneous objects

Header. Information about the current level.

CASES:
There are two other classes that can be added to the diagram,**Textbox** and **Label** classes. Add them to their respective objects.

Beware that objects with the same class require to have the same slots and operations.

Case Viewport. An area to display the case to the player.

Draggable Item Viewport. An area where players can drag items that represent class, object name, slot, and operation to their target objects.

A player drags Label class

Drawing Viewport. An area where players put the model elements.

The elements, attributes, or structure of the model don't have to conform strictly to the general/official rules, as long as they meet the rules defined in the objectives that are written in Epsilon Validation Language.

PALETTE:

Pallette Viewport. Contains all the model elements that can be dragged into the drawing viewport.

PROGRESS:
Completed Objectives: 1
Objects: 6
Links: 5
Slots:6
Operations:7
Points:170
Time Elapsed: 00:13:03

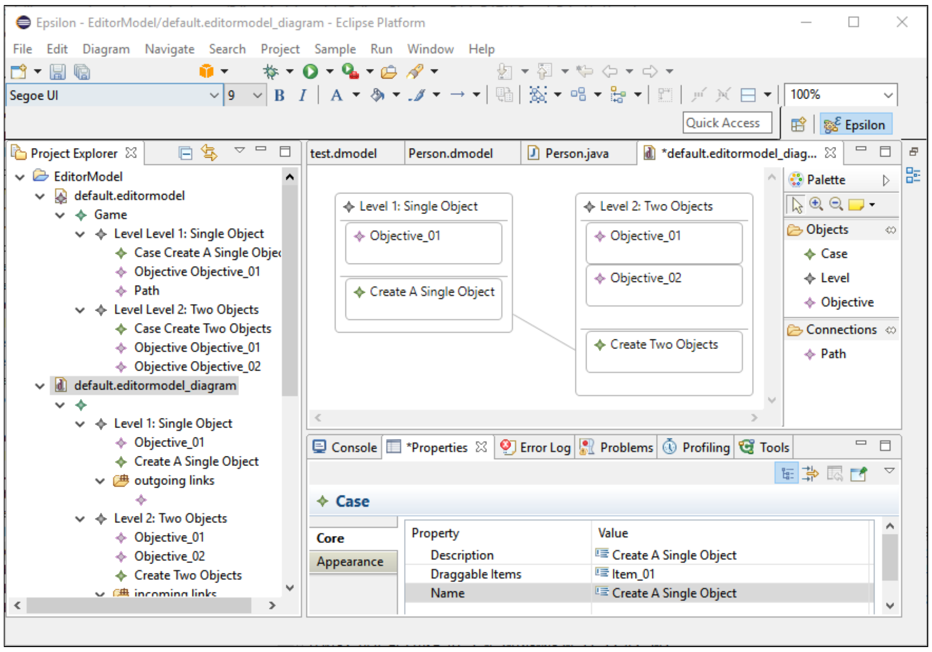
Progress Viewport. An area to display the metrics of the model , including time and points.

OBJECTIVES:
1. **Put TextBox class to its objects**
2. Put Label class to its objects

Objective Viewport. An area to display the objectives of the current level. Accomplished objectives are displayed in bold, green colour.

Fig. 2: The game's display.

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.



Property	Value
Description	Create A Single Object
Draggable Items	Item_01
Name	Create A Single Object

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

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. K. Werbach, "(Re)defining gamification: A process approach," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8462 LNCS, pp. 266–272, Springer Verlag, 2014.
18. K. M. Kapp, "What Is Gamification?," *The Gamification of Learning and Instruction: Game-Based Methods and Strategies for Training and Education*, pp. 1–23, 2012.
19. A. R. Yohannis, Y. D. Prabowo, and A. Waworuntu, "Defining gamification: From lexical meaning and process viewpoint towards a gameful reality," *Information Technology Systems and Innovation (ICITSI), 2014 International Conference on*, no. June 2015, pp. 284–289, 2014.
20. 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.
21. 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.
22. 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.
23. 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.
24. J. Groenewegen, S. Hoppenbrouwers, and E. Proper, "Playing ArchiMate models," 2010.
25. O. Richardsen, "Learning Modeling Languages Using Strategies from Gaming," 2014.
26. 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.
27. 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.