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, it is perceived as relatively challenging for learners to develop the necessary abstraction skills to master the subject. On the other side, 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 the 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. The target outputs of this research are a gamification design framework and its games, which so far have been translated into initial prototypes. The effectiveness of the framework and its games will be evaluated using controlled experiments.

Keywords: software modelling, gamification, learning, abstraction

1 Introduction

Software modelling is commonly perceived as a demanding 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 which is essential for analysing and designing real-world software. Weak software modelling skills will likely cause software engineering students to face further with their degrees, as most of the software engineering related subjects involve of intrinsic abstraction problems [2].

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 students to grasp information hiding, generalisation, approximation or reformulation, and separating relevant from irrelevant aspects [4]. To overcome these challenges, we need to put more effort into software modelling learning design, 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 purposes other than leisure has drawn significant attention. Gamification [5] and Serious Games [6]

have been proposed as solutions to motivational problems that emerge when users are required to engage in activites that they perceived as boring, irrelevant, or difficult, e.g. Learning sorting algorithms [7] or C-programming [8].

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:

1. Which processes, aspects, principles, or components of software modelling and their teaching and learning practices would benefit from gamification?
2. What types of game elements and in what roles 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 achieve software modelling gamification?
4. To what extent does gamification of software modelling improve learners motivation, engagement, and performance?
5. To what extent are software modelling tutors benefited from the gamification design framework?

Due to space limitation, this paper does not cover some aspects of this research, such as the architecture of the gamification design framework 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], learning modelling language through alternative communication channel [11], immersive visual modelling through virtual environments [12], project-based learning [13], to learning modelling from code generation investigation [14]. However, most of the approaches have weaknesses in motivating learners to engage continuously, frequently, and actively to learn software modelling, which are the important aspects impacting greatly on learning [15].

To address the lack of engagement, we investigate gamification of learning, an approach that provides students with a new way of learning software modelling that is more fun and engaging. The use of game elements for a purpose other than leisure is called gamification [5]. Gamification design is still an ongoing challenge [16], and, to date, there is no gamification design framework that particularly structures the design of software modelling gamification.

There is very little/superficial work on software modelling gamification. 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 [17]. After extensive literature exploration, only four works have been identified on applying gamification for

software modelling. None of these works addresses software modelling learning in general. Instead, they address specific topics such as activity diagramming [18], coupling and cohesion [19], and enterprise architectures [20] [21]. Most of the works also cover pedagogical aspect superficially or not at all and validation is restricted to a very limited number of users [18][19][20].

3 Research Methods

Since the output of this work is a design artefact—a gamification design framework that facilitates software modelling tutors design and develop games of software modelling learning, we decided to utilise the Design Science Research Methodology (DSRM) [22] 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, (2) definition of objectives for a solution, (3) setting of targets for a solution, (4) design and development, (5) demonstration and evaluation, 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 the Design Lenses and Intrinsic Skill Atoms [23] to produce a gameful design in the design and development activity.

4 Gamification Design Framework

There will be two outputs of this research in terms of software, the gamification design framework—the framework to produce software modelling games—and the software modelling gamification—the generated games. Instead of developing these games manually, we plan to follow a model-based approach. We will use metamodel annotations, in the spirit of Eugenia [24], to define the graphical syntaxes of modelling languages and separate models to specify the game elements (levels, objectives, constraints, etc.) of each game. These models will be then consumed by a model-to-text transformation to produce fully-functional language-specific games. Therefore, the framework supports software modelling tutors in the design and customisation of the games at the high level of abstraction and so as to automatically build the game.

So far we have implemented an editor of the gamification design framework (Figure 1) and its prototype game (Figure 2) for object diagrams, a first version of the language that can specify the flows, levels, challenges, and objectives of the game. There will be more modelling languages (e.g. BPMN, state-charts, GSN, UML) that we envision to support in the future to reduce bias in our experiments. Moreover, we also plan to implement the games using web technologies so that they are easily accessible to a wide audience.

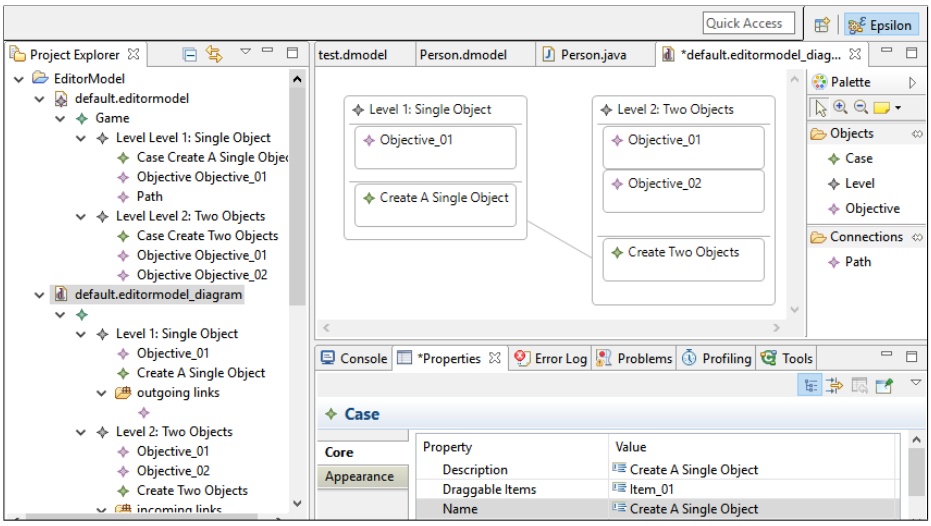

Fig. 1: The visual modelling editor to automatically generate the game.

5 Game Design

For each modelling language we envision the development of a dedicated game containing game elements that are derived from the Design Lenses and Intrinsic Skill Atoms [23]. The generated will mimic a graphical modelling tool and at each level it will require the learner to graphically construct/adapt a model to satisfy a set of requirements and constraints.

The game will have levels with gradually increasing difficulty as well as variety in its challenges, to expose learners to different kinds of domains, models, and diagrams. Tutorial is also planned to be implemented into the game to help learners familiarise themselves with the control system and the flow of the game. The game will also utilise pre-constructed models to ease learners perform modelling without having to start from scratch.

The game will includes interim goals and intrinsic rewards to motivate learners. For software modelling, each type of modelling (e.g. object modelling, collaboration, process) will have several stories. A story will represent a specific case study to introduce learners to problems in specific domains. Every story will consist 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 a previous level, giving the learner a sense of step-by-step progress to complete the domain problems. Each story or level will introduce new concepts as well as combining old and new concepts. Thus, as the learners progress, their competence in modelling is developed, giving an intrinsic reward.

A real-world problem can be very complex and time-consuming to model. 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

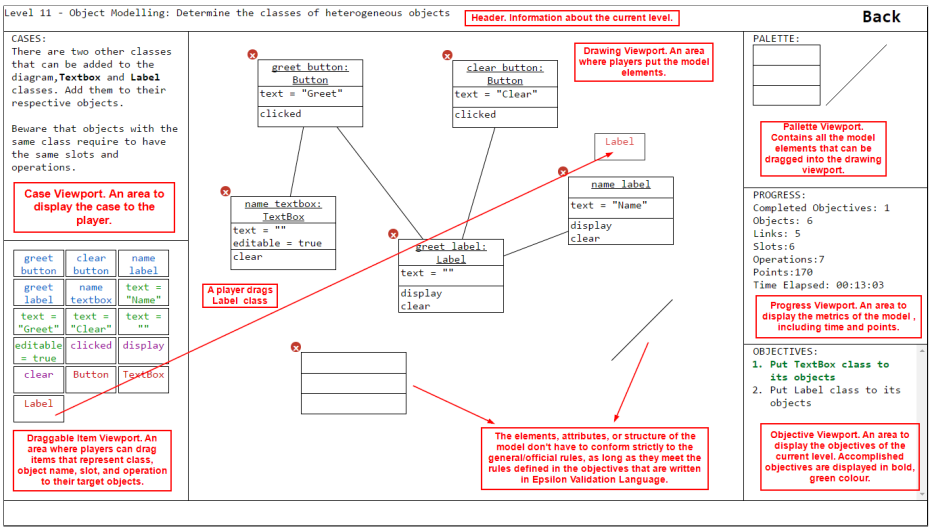

Fig. 2: The display of the generated game.

on the main concepts. Thus, 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) wil be implemented to facilitate learners in performing the core activities. Likewise, fuzziness will also be used to provoke learners' creativity since most of the time there is no single correct model for the problem at hand. Attractive design will also be significant to motivate learners to interact with the game. The game should be able to give immediate, glanceable, actionable feedback to keep learners on track and monitor the state of the game. Interesting and varied feedback should be designed appealing to the learners' motives.

6 Evaluation

We wish to evaluate (1) the effectiveness of the games and (2) the productivity and maintainability benefits of the gamification design framework. For the effectiveness evaluation, controlled experiments will be used. 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 the support from the games. After some time learning, their performance will be measured by their ability to solve the problems. For the evaluation of the gamification design framework, the respondents will come from software modelling tutors. They will be asked to prepare certain materials in software modelling and use the materials while in their teaching. The comparison will be on the productivity and maintainability benefits *with* or *without* the support of the framework. To

evaluate the generality of the results of both evaluation, conducting experiments in different years and countries is also considered.

Additionally, surveying with questionnaires or interviews might be conducted to investigate the underlying variables or processes. Structural equation modelling [25] is also an option if measuring the effects of the identified underlying variables is required. An alternative method for understanding of the underlying variables and processes is through investigating the games' 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 game and framework, 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.

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*, 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. 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.
14. 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.
15. T. L. Naps, "Jhavé: Supporting algorithm visualization," *IEEE Computer Graphics and Applications*, vol. 25, no. 5, pp. 49–55, 2005.
16. 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.
17. 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.
18. O. Richardsen, "Learning modeling languages using strategies from gaming," Master's thesis, Norwegian University of Science and Technology, Norway, 2014.
19. 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.
20. J. Groenewegen, S. Hoppenbrouwers, and E. Proper, "Playing archimate models," in *Enterprise, Business-Process and Information Systems Modeling*, pp. 182–194, Springer, 2010.
21. D. Ionita, R. Wieringa, J.-W. Bullee, and A. Vasenev, "Tangible modelling to elicit domain knowledge: an experiment and focus group," in *International Conference on Conceptual Modeling*, pp. 558–565, Springer, 2015.
22. 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.
23. 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.
24. 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.
25. 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.