Progress Report

Model-Driven Gamified Software Modelling Learning

Alfa Ryano Yohannis
ary506@york.ac.uk

Supervisor:
Dimitris Kolovos
Fiona Polack

Department of Computer Science
University of York
United Kingdom

March 10, 2017

gamification does generate benefits and positive effects. Specifically in the field of software engineering, Pedreira et al. [13] also performed a systematic review on the application of gamification in software engineering. Most existing studies focus on software development, project management, requirements, and other support areas, but none of them focuses on software modelling. They also found fewer studies reporting empirical evidence to support gamification research. They argued that existing studies in the field are quite new, thus more research effort is needed to investigate the impact of gamification in software engineering. Reports of the positive impact of gamification in various fields, particularly addressing engagement issues, encourage us to leverage it to deliver gamified software modelling learning, an area which has received little attention for the application of games and game elements so far.

In terms of learning, pedagogical aspect cannot be neglected. Several concepts from pedagogy will be applied to drive the design of the platform, particularly the best practices from instructional design, a mature field in designing learning activities so that learning processes can be more efficient and effective. Therefore, incorporating gamified approaches as well as the best practices of instructional design will bring great benefits to tackle the problems in learning software modelling. The platform is being designed to be suitable for higher-level undergraduate and postgraduate students with some of experience of software engineering.

Rather than just being a learning content for learners, software modelling also broadens opportunity to improve the meta-processes of the gamified software modelling learning, through the application of Model-Driven Engineering (MDE) approaches. Instead of developing the software modelling games manually, a model-based approach is applied. A platform is being developed and it will act as a design framework to design learning activities for topics in software modelling. The learning activity design then will be transformed to generate gamified learning activities of software modelling.

1.2 Research Questions

Thus, this research proposes a main research question "What kind of platform that produce gamified learning activities to improve software modelling learning?". The word 'platform' means a software environment for tutors and learners to design and perform gamified software modelling learning. The word 'produce' indicates the platform support tutors in designing and generating gamified learning activities. Thus, the platform should be expressive enough to support various visual modelling notations and the creation of different patterns of learning activities for learning software modelling. The word 'improve' implies that learning with gamified approaches enhances learners' engagement and learning performance. They are more durable, frequent, and active compared to learners that only use didactic approach. Also, the former perform better in knowledge and skill acquisition and application compared to the latter. To answer the main research question, following sub research questions need to be investigated:

1. How can a platform that implements Model-Driven Engineering approaches help tutors and learners to produce and learn gamified software modelling learning?

2. Does the platform improve learners' engagement and performance in learning software modelling?

3. Does the platform help tutors in developing gamified software modelling learning?

1.3 Research Objectives

The solution proposed by this research is to produce a platform that can support tutors to design gamified learning activities as well as learners to learn software modelling in a gamified way. More precisely, this research aims to meet the following research objectives that are derived from the solution:

1. Design and develop a platform that accommodates gamified approaches and instructional design to be implemented through harnessing the best practices of Model-Driven Engineering.

2. Perform controlled experiments to measure the significance of the platform in improving learning engagement and performance compared to traditional method, didactic learning without the support of gamified approaches.

3. Perform controlled experiments to measure the productivity and maintainability of the platform regarding supporting tutors in designing and producing gamified learning activities of software modelling learning.

1.4 Research Outputs

By the end of this research, three potential research outputs have identified will have been produced:

1. A platform for tutors to design and produce gamified learning activities of software modelling and for learners to learn software modelling in gamified ways.

2. Controlled experiments: comparisons in learning engagement and performance of learners between gamified version and the traditional one of software modelling learning.

3. Controlled experiments: comparisons in productivity and maintainability of of the platform in supporting tutors designing and producing gamified software modelling learning activities.

Chapter 2

Progress Review

In this section, the progress review of this research is discussed. First, the Design Science Research Methodology (DSRM) [14], the main research method employed on this research, is discussed briefly, and then followed by a discussion of progress on activities composing DSRM.

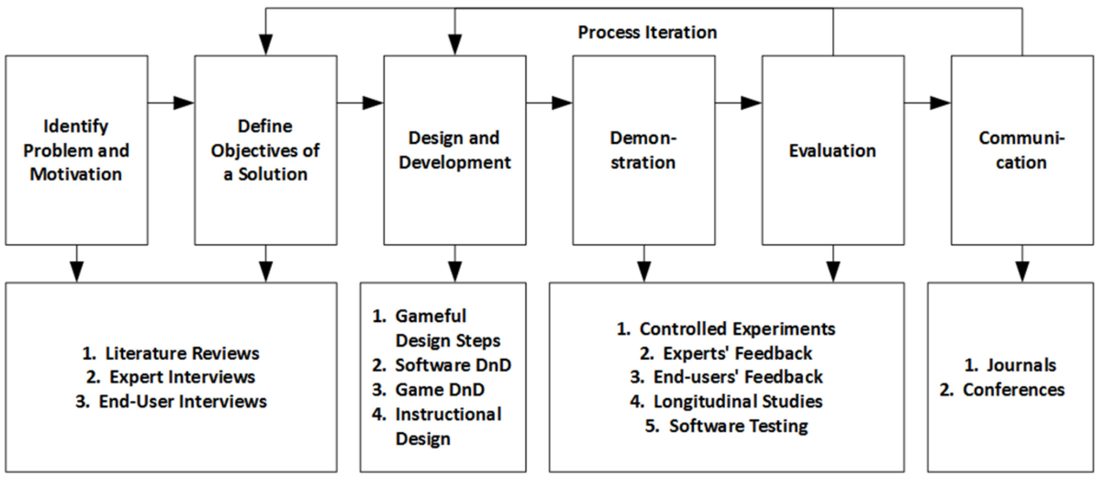

Figure 2.1: Design Science Research Methodology. Adapted from Peffer et al. [14].

2.1 Research Methods

Design Science Research Methodology (DSRM) is selected as the main research method since the main output of this research is a design artefact, a platform to design and generate gamified software modelling learning. DSRM provides a comprehensive conceptual framework and activity guidelines for understanding, developing, executing, and evaluating a design artefact (Figure 2.1). Another reason is that it positions itself at the top level of abstraction without going into much detail of how to perform each activity, researches can freely choose other more concrete research methods to carry out the activities. For examples, we can conduct literature reviews, surveys, or expert interviews to determine research problems, motivations, of this research.

2.3.1 Requirements from a Preliminary Survey

A preliminary survey to identify learners' needs, motivations, and challenges has been conducted according to the Research phase of the Deterding's Gameful Design Framework [15]. The preliminary survey aims to identify the requirements in designing the gameful aspect of the platform. It is not intended to measure significance, but it is qualitative investigation to reveal needs, motivations, and challenges in learning software modelling from learners' perspective. The preliminary survey is also in line with the Design Science Research Methodology, in order to identify the problem and motivation so that this research can define objective of a solution accurately in the second activity.

Online questionnaires were distributed to students of Model Driven Engineering (MODE) 2015/2016 module. The students were in their Software Engineering master programme at the University of York. From 21 students, only 4 completed the questionnaires. Their responses can be found in Appendix A. Since the number of respondents are small for generalisation, the same survey is planned to applied to next term MODE students.

Results. To identify the learners' needs, respondents were asked two questions. Question 1 aims to identify students' expectations before starting the module, while question 2 aimed at identifying what the students found important after taking the module. Based on the responses, the reasons why the students took MODE module because they wanted to increase their knowledge on Model-Driven Engineering (MDE), possess new advanced skills or abilities and improve their literation of MDE tools. After completing the module, the students valued that the most important lessons were getting new knowledge—domain modelling, metamodel, abstract syntax, abstract thinking, model validation, and the application of models—and skills—generating code, creating DSL, and improving their tool skills.

Other three questions (Q3-Q5) were also asked the students to identify their motivation in taking MODE module and Learning Model Driven Engineering. Question 3 asked about the reasons behind their decision taking MODE module. Two students stated that they took the module because it is compulsory, but the rest of the students said that they took the module because MDE is an advanced topic and they wanted to see its applicability in the industry and whether it will improve their ability–knowledge and skills.

Question 4 asked the students about what would motivate them more to learn MDE. The students responded that they would be more motivated if they could perceive the advantages of MDE: efficiency and effectiveness it could offer, the benefits of its application in the organisation or real world examples, and its genericity—MDE application in languages other than Java or models other than UML/EMF.

Question 5 asked the students the most basic, underlying motives that make them commit to learning MDE. Substantially, they answered that their main motivation is to gain new ability–knowledge and skills, such as the ability to make an abstraction, advanced skills that are applicable in industry, and knowledge of real-

Table 2.2: Requirements derived from the literature review.

Category	Code	Requirements from Literature Review
Contents	RL01	Teach MDE Definition
	RL02	Teach semantics, syntaxes, notations
	RL03	Teach Modelling, metamodelling, model validation, model transformation
	RL04	Teach the applications of MDE
	RL05	Teach modelling in various domains/contexts
Priciples and Practices	RL06	Modelling is abstract thinking
	RL07	Object-orientation prerequisite
	RL08	Measure student's model's quality
	RL09	Problem solving first, detail specifications and tools get in the way
	RL10	Provide support to solutions, not answers
	RL11	Teach broadly, throughout, not deeply
	RL12	Teach with different modelling languages
	RL13	Make it fun
	RL14	Teach from ground, real-world objects, up to abstraction
Tool Design	RL15	Support and documentation
	RL16	Build knowledge incrementally
	RL17	Flexibility to explore learning
	RL18	Positive reinforcement
	RL19	Convince of the value of the topic being learned
	RL20	High usability

2.4.1 Design

The discussion of the design of the platform is divided into three sections, the pedagogical aspect design, the game aspect design, and the model-driven aspect design. They are discussed in the following subsections.

Pedagogical Aspect Design

Several existing learning concepts will be applied in the design process of the platform. In this section, we will explain the relationships between the learning models, their contributions, and how they will be applied to the design of the platform are depicted in in Figure 2.2 and Figure 2.3.

We decided to implement challenge as the fundamental game element that exists in the design of our game since it is one of the key features that exist in every game. The challenge is a crucial game element since it stimulates and provokes a player to engage with platform. We translate challenge into series of levels in our design and of course higher levels come with higher difficulty. To realise this, we borrow the learning activities—remember, understand, apply, analyse, evaluate, concrete experience (CE), reflective observation (RO), abstract conceptualisation (AC), and active experimentation (AE).

We could apply this cycle to frame learners' activities in gaining new knowledge through solving a problem given in a level. For example, the first time players play a new level, at that moment they encounters a concrete experience (CE). Immediately, they attempt to identify and characterise the problem given in that level and recall any knowledge that is relevant to solve the problem (RO). Next, they construct a solution for the problem that they face (AC). After constructing the solution, they apply the solution to the problem (AE), experience the result (CE), and then evaluate whether the solution solves the problem of the level or not (RO). Any gap that appears will update their knowledge. They use their newly updated knowledge to produce a new solution (AC) that could be applied to the same problem at the same level or different problem in other levels (AE). In case that the player already 'game Over' and they cannot apply their new solution, AE occurs when they replay the same level or play a similar level.

Table 2.3: Requirements derived from learning models (section **??**).

Category	Code	Requirements from Learning Models
Learning Models	RM01	Design satisfies Bloom's taxonomy.
	RM02	Design suffices Kolb's experiential learning model.
	RM03	Design meets Keller's ARCS motivational model.
	RM04	Design fulfils scaffolded learning.
	RM05	Design complies with the theory of Flow.

Learning activities in Bloom's taxonomy also correspond to the steps in Kolb's learning cycle [21] and both have been applied together to design instructions in different fields [22, 23, 24]. Therefore, we argued that both could be implemented simultaneously; Bloom's taxonomy provides learning activities while Kolb's model addresses learning cycles in the design of our game. To simplify our work, we summarise the elaboration of design and learning models into a list of requirements (Table 2.3) that will be used in the design and evaluation activities.

Game Aspect Design

In this research, we plan to assess whether gamification is beneficial for learners of graphical software modelling languages. We choose graphical modelling languages since they are the common languages used in modelling, whether in academia or industry, and extensively used in Model-Driven Engineering. Standard graphical modelling languages like UML[1] and BPMN[2], often used in Model-Driven Engineering, are some of the use-cases.

Modelling can be expressed in different modelling languages. To minimise bias and ensure the generality of our platform, we plan to experiment and support

[1]http://www.uml.org/
[2]http://www.bpmn.org/

several graphical modelling languages (e.g. UML, BPMN, state-charts). For each modelling language, we envision the development of dedicated platform that will be derived from the Gameful Design Framework [15]. The platform will mimic a graphical modelling tool, and at each level, it will require the learner to graphically construct or adapt a model to meet a set of constraints and requirements.

The platform will have levels of gradually increasing difficulty as well as variety in its challenges, to expose learners to different kinds of domains, models, and diagrams. Tutorials are planned to be embedded into the platform to help learners familiarise themselves with the control system and the flow of the platform.

The platform will incorporate interim goals and intrinsic rewards to motivate learners. Each type of modelling language (e.g. object modelling, collaboration, process) will have several stories. A story will represent a specific case study to introduce learners to particular problems in specific domains. Every story will be composed of several levels, and every level will have one or more objectives that a learner needs to accomplish to complete it. A level may also be a continuation of a previous level, giving the learner step-by-step progression to complete the domain problems. Each story and level will introduce new concepts and link them with previously introduced concepts.

A real-world problem can be time-consuming and very complex to model. Thus, the inessential activities that are not significant to the core concepts that are being taught should be excluded. As a result, learners will be more focused on the main concepts. Thus, game elements like limited choices (i.e. only limited items can be dragged), microflows (i.e. put the right element to its right place), and bite-sized actions (e.g. drag and drop) will be implemented to facilitate learners in performing the core activities. Likewise, fuzziness will also be used to stimulate 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 platform. The platform should be able to give instant, noticeable, and actionable feedback to maintain learners' engagement and monitor their progress. Interesting and varied feedback should be designed to appeal to the learners' motives. We also plan to implement the platform using web technologies so that they are accessible to a wide audience.

The details of the application of Deterding's Gameful Design to our design process are presented in Appendix B. The process also produced storyboards that are the preliminary design of levels and graphical user interface of our game (Appendix C).

Model-Driven Aspect Design

We plan to build a platform that will facilitate the design and generation of platform. Rather than developing platform for each graphical modelling language manually, we will follow a model-based approach. In the spirit of Eugenia [25], we will use metamodel annotations to define the graphical syntaxes of modelling languages and separate models to specify the game elements (constraints, objectives, levels, etc.) of the platform. These models will be then consumed by a model-to-text transfor-

Chapter 3

Research Plan

We plan to complete literature study and develop prototype by the end of the first year (2016), and address gamification of modelling and metamodelling in the second year (2017) and third year (2018) respectively.

Table 3.1: Research Timetable

Month	**2016**	**2017**	**2018**	**2019**
01	Literature Review	- Develop Prototype - 1st experiment - 2nd survey	- Update Prototype - 3rd experiment - 4th survey	Thesis writing
02				
03				
04	- **25-minute Seminar** - Questionnaire Design - 1st survey	**Progress Report**	**Thesis Audit**	
05		(In Indonesia) - Update Prototype - 2nd experiment - 3rd survey	(In Indonesia) - Update Prototype - 4th experiment - 5th survey	
06	Develop Prototype (for modelling)			
07				
08				
09		**Thesis Outline**		
10		Develop Prototype (metamodeling)	**40-minute Seminar**	
11	**Qualifying Dissertation**		Thesis writing	**Thesis Submission**
12	Develop Prototype			

~~Chapter 4~~

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.* ACM, 2012, pp. 39–50.

[2] J. Whittle, J. Hutchinson, M. Rouncefield, H. Burden, and R. Heldal, “Industrial adoption of model-driven engineering: Are the tools really the problem?” in *International Conference on Model Driven Engineering Languages and Systems.* Springer, 2013, pp. 1–17.

[3] J. Bezivin, R. France, M. Gogolla, O. Haugen, G. Taentzer, and D. Varro, “Teaching modeling: why, when, what?” in *International Conference on Model Driven Engineering Languages and Systems.* Springer, 2009, pp. 55–62.

[4] G. Engels, J. H. Hausmann, M. Lohmann, and S. Sauer, “Teaching uml is teaching software engineering is teaching abstraction,” in *International Conference on Model Driven Engineering Languages and Systems.* Springer, 2005, pp. 306–319.

[5] J. Kramer, “Is abstraction the key to computing?” *Communications of the ACM*, vol. 50, no. 4, pp. 36–42, 2007.

[6] L. Saitta and J.-D. Zucker, *Abstraction in artificial intelligence and complex systems.* Springer, 2013, vol. 456.

[7] O. Hazzan, “Reflections on teaching abstraction and other soft ideas,” *ACM SIGCSE Bulletin*, vol. 40, no. 2, pp. 40–43, 2008.

[8] 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.

[9] 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.* ACM, 2011, pp. 9–15.

Chapter 5

Publications

We have published papers in the following conferences or journals:

1. A. Yohannis, “Gamification of software modelling learning,” in *the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS 2016) Doctoral Symposium*. CEUR, 2016. [28].

Appendix C

Storyboard: Object Diagram

- Objects
  - Create single object (Level 01)
  - Create two objects (Level 02)
  - Create multiple objects (Level 03)
- Links (Relationships)
  - Create single link (Level 04)
  - Create multiple links (Level 05)
- Slots (Attributes)
  - Determine a slot and its value (Level 06)
  - Determine slots and their values (Level 07)
- Operations/Methods
  - Determine operation (Level 08)
  - Determine multiple (Level 09) operations
- Class of Objects
  - Determine the class of homogeneous objects (Level 10)
  - Determine different classes of heterogeneous objects (Level 11)
- Case Studies
  - Reconstruct the model from the beginning (Level 12)
  - Apply the skills on different/similar problem (Level 13)