Gamifying Model-Based Engineering: the PapyGame Tool

Antonio Bucchiarone[a], Maxime Savary-Leblanc[b], Xavier Le Pallec[b], Jean-Michel Bruel[c], Antonio Cicchetti[d], Jordi Cabot[e], Sébastien Gérard[f]

[a]*Fondazione Bruno Kessler, Via Sommarive 18, Trento, Italy*
[b]*Univ. Lille, CNRS, IRCICA, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France*
[c]*University of Toulouse, IRIT, Toulouse, France*
[d]*Mälardalen University, IDT, Västerås, Sweden*
[e]*Luxembourg Institute of Science and Technology, Esch-sur-Alzette, Luxembourg*
[f]*Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France*

Abstract

Modeling is an essential and challenging activity in any engineering environment, and it requires some hard-to-train skills such as abstraction and communication. This makes it difficult for educators to teach or train their students, co-workers, or users. The audience of this paper is both educators and learners who struggle with modeling. To address this challenge, we present PapyGame, a gamified version of a robust modeling environment (Papyrus) that aims to improve the learner's motivation, make the learning process an enjoyable experience, and boost learning outcomes. Gamification is the exploitation of gaming mechanisms for serious purposes, such as promoting behavioral changes, soliciting participation and engagement in activities, and more. The paper presents PapyGame's functionalities, architectures, illustrative scenarios, and its potential impact on both educators and learners.

Keywords: Model-Based Engineering, Education, Training, Gamification, Papyrus

Metadata

Nr.	Code metadata description	Please fill in this column
C1	Current code version	V1.0
C2	Permanent link to code/repository used for this code version	`https://git.eclipse.org/c/papyrus/org.eclipse.papyrus-papygame.git/`
C3	Permanent link to Reproducible Capsule	
C4	Legal Code License	MIT
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	HTML, CSS, Javascript, Java
C7	Compilation requirements, operating environments and dependencies	Thanks to the Rich Client Platform (RCP - `https://ci.eclipse.org/papyrus/view/PapyGame/job/papyrus-papygame-2021-12/lastBuild/`), PapyGame covers all OSs and no compiling is required from the user-side, all is done on the server.
C8	If available, link to developer documentation/manual	*https : //www.papygame.com/*
C9	Support email for questions	bucchiarone@fbk.eu

Table 1: PapyGame Information.

1. Motivation and significance

In many studies around Model-Based Systems Engineering (MBSE) adoption, the usability of tools is very often stressed as one of the key issues for the adoption of MBSE paradigms. Several countermeasures have been tried, like mixing learning styles, partitioning courses into micro-modules, etc. Their common goal is to implicitly keep students motivated through small amounts of effort and rapid/visible progression in their study. On the same line, *gamification* uses gaming mechanisms for enhancing students' engagement [1, 2, 3].

Previous research has shown a strong interest in the use of gamification in modeling, with some promising results [4, 5, 2, 6, 7, 8, 9]. However, these efforts have mainly been ad-hoc attempts by modelers to create gamification environments for specific experimental scenarios[10, 11, 12, 13].

There is still a need for a fully-fledged gamification environment to be integrated into modeling tools. Some initial attempts have been made by authors in [10, 11, 12, 13], but these solutions only address specific learning objectives and are not comprehensive. Our proposed approach aims to address this gap by providing a general approach to integrate modeling tools with a fully-fledged gamification environment that considers various learning dimensions, including learning abstraction, modeling language, and modeling tool. However, there are still challenges to be addressed in terms of research and technical aspects of modeling in order to implement these target gamification scenarios. A more recent survey [14] discusses several game-based and game development-based learning approaches devoted to software design. However, it confirms the scarce experimentation of gamification techniques for software design. All the research introduced until now reinforces the idea that there is a strong interest in the use of *gamification in modeling* and some first promising results. Still, they also show that previous attempts are mainly the work of modelers trying to manually create some ad hoc gamification environment for their specific experimental scenarios. Recently, virtual reality (VR) approaches have also been proposed as more immersive environments to edit UML models [15, 16]. Since these environment propose a radically new way of tackling modelling, they are typically able to capture the attention of students, making them suitable to implement gamified modelling frameworks for education [17].

In this respect, our vision abstracts those attempts by proposing a general approach for integrating modeling tools with a fully-fledged gamification environment to facilitate the specification and deployment of all kinds of modeling games. Within this paper, we present our initiative started two years ago and known as PapyGame[1], a gamified modeling tool that provides an innovative and engaging way for users to learn about modeling. By incorporating game-like elements into the learning process, the tool can make the experience more enjoyable and memorable. *Learners* can interact with the tool in a way that feels less like traditional learning and more like play, which can make the learning process feel less daunting and more approachable. Additionally, the tool can provide instant feedback and rewards, which can help learners stay motivated and engaged throughout the learning process. Overall, the objective of this gamified modeling tool is to provide an effective and engaging way for users to learn and practice modeling skills, while also making the experience fun and enjoyable. Moreover, PapyGame can be a valuable asset for teachers to enhance the learning experience of their students and promote better learning outcomes. We have experimented the gamification of software modeling in a real and robust modeling environment (Papyrus[2]) to reduce its use complexity.

This article focuses on presenting the functionalities, architecture, and an illustrative scenario of PapyGame, as well as its potential impacts on modeling education. The paper aims to introduce the tool to the academic community, highlighting its features, and showcasing how it can be used to create engaging and interactive exercises that can help students learn modeling concepts effectively. The article also discusses the potential impacts of PapyGame on modeling education, such as improving student engagement and enhancing learning outcomes.

2. Software description and functionalities

PapyGame is a plugin[3] for the modeling tool Papyrus[2]. In the following, we present its overall architecture and the main software functionalities provided.

2.1. Software architecture

The objective of PapyGame is to gamify the learning of modeling by integrating a game view within the Papyrus UML editor. Figure 1 shows the general architecture of PapyGame.

The *Gamification Engine* allows to implement each key concept of the game associated to a specific learning path using the Gamification Design Framework (GDF) [18] through user interfaces accessible from a web browser.

[1]`https://www.papygame.com`
[2]`https://www.eclipse.org/papyrus/`
[3]`https://ci.eclipse.org/papyrus/view/PapyGame/`

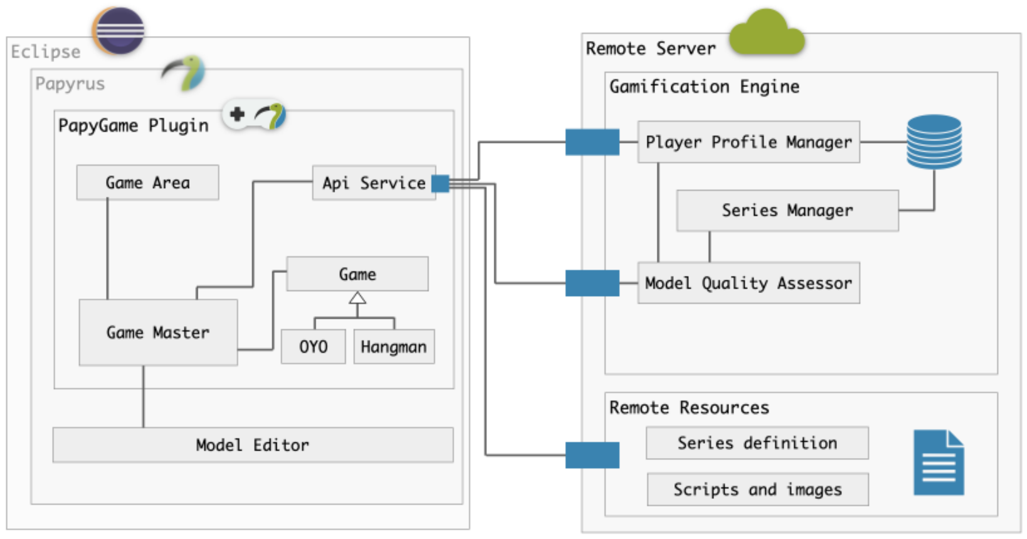


Figure 1: PapyGame Technical Architecture.

For PapyGame, the Gamification Engine plays three main roles: *Learner Profile Manager*, *Series Manager*, and *Model Quality Assessor*. To implement gamification mechanisms related to progression, experience, rewards, and, more generally, to a player's profile, the Gamification Engine provides a set of administration interfaces and endpoint APIs to manage the player database. The Gamification Engine natively manages the notion of a player profile by integrating the nickname, the current level of the player, and the storage of the current level of the rewards obtained by the player. User profiles can be created, read, updated, or deleted from REST API endpoints exposed automatically by the Gamification Engine. These endpoints are called by (i) the PapyGame client when logging in or displaying the dashboard, or (ii) the Model Quality Assessor component of the Gamification Engine when receiving a completed game request from the client.

The definition of a *Series* is done in two distinct steps. First, the Modeling Teaching expert has to design the different levels that will compose the game series by writing their instructions, choosing the associated games, by defining an identifier for them, and eventually associating a UML starting model if a comparison or an automatic generation of the exercise is planned. The series and the levels that compose it are completely defined in JSON and available on the resource server. Each level connects to the Gamification Engine rules thanks to a specific id held by the series, references a game, a source diagram from a source model, and features textual instructions.

In a second step, the Gamification Expert has to define the conditions of success or failure, as well as the mechanism of reward or experience attribution. These rules are defined in the Open Source DROOLS rule engine [19] in the Gamification Engine, which provides the action mechanism. Each action created can be called through an endpoint API in GET or POST, which is called by the *Model Quality Assessor* component.

The *Model Quality Assessor* bridges connect the client requests issued by the *PapyGame plugin* containing the player diagram data and the rules of each level defined in the *Gamification Engine*. When executing the rules, the Model Quality Assessor evaluates whether the attempt is a failure or a success, and if so, updates the player profile (progress and history) thanks to the endpoint exposed by the *Player Profile Manager*. The PapyGame client is distributed as an Eclipse plugin written is Java, that is available for download and installation in its update site[4].

The *Game Area* is a view container that allows other components to display their views. The *Api Service* is in charge of handling network requests to the server. It automatically parses the requests in JSON with the use of Gson[5].

When an API connection fault occurs, the *Api Service* detects the error and attempts to resolve the issue. This may involve retrying the connection, notifying the user of the error, or taking other appropriate actions based on the type and severity of the error. By managing API connection faults, the *Api Service* helps ensure that the application can maintain reliable and consistent communication with the server.

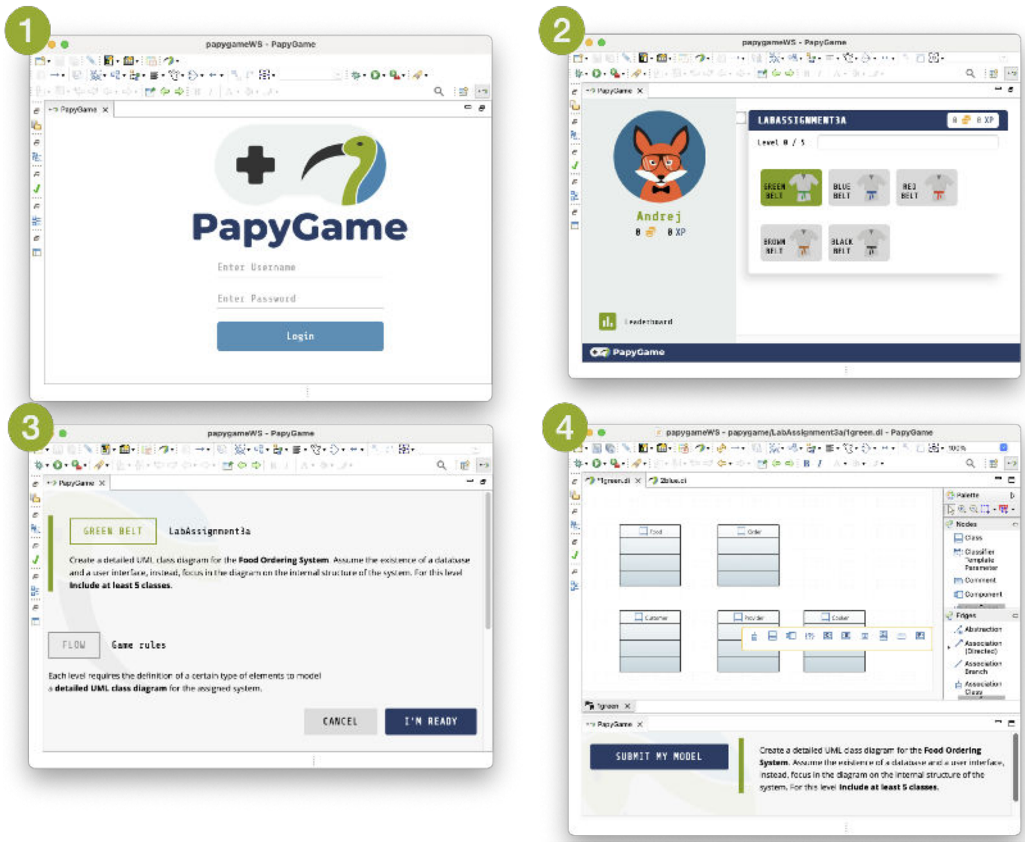
While the Gamification Engine is typically deployed as a Docker image on a server, it is also possible to install and use it locally within the Papyrus plugin. This can be done by downloading the necessary files and configuring the plugin to use the local installation instead of the server. By installing the Gamification Engine locally, developers can test and develop gamification features without the need for a dedicated server or internet connection. However, it's important to note that the local installation may have limitations in terms of scalability and performance compared to the server deployment.

In PapyGame, each game contains four views: the Preview View, the InGame view, the GameSuccess view, and the GameOver view. Technically, each view is associated with one Java class and one HTML file. The Java classes of the views contain the internal behaviour of the Game, e.g., how to react to an input user event and define an entry-point to be executed by the *Game Master* component.

[4]`https://ci.eclipse.org/papyrus/view/PapyGame/`

[5]`https://github.com/google/gson`

The *Game Master* has the role of coordinating the different components. It receives the updates from the events in the *Model Editor*, the click interactions from the *Game Area*, and is connected to the *API Service* to query the server when required. To associate players with their stored profile, PapyGame asks them to create an account. This is achieved directly from the login screen (Figure 2 (1)).



Create a detailed UML class diagram for the **Food Ordering System**. Assume the existence of a database and a user interface, instead, focus in the diagram on the internal structure of the system. For this level **Include at least 5 classes.**

Each level requires the definition of a certain type of elements to model a **detailed UML class diagram** for the assigned system.

Create a detailed UML class diagram for the **Food Ordering System**. Assume the existence of a database and a user interface, instead, focus in the diagram on the internal structure of the system. For this level **Include at least 5 classes.**

Figure 2: User Interfaces of PapyGame.

When the *Game Master* initiates the *Dashboard phase*, it first ask the Api Service to query the server to retrieve (i) all series available and (ii) the fully-loaded player profile. Once received, it displays the series, the progress history, and the player profile information in the Dashboard View that is loaded in the Game Area, as presented in Figure 2 (2). The GameMaster stays idle until the player clicks on one available level. This triggers the launch of the Game phase, which starts with the Game Preview View.

The *Game phase* starts when the player clicks on an available level from the Dashboard View. The Game Master is then responsible for switching the view in the Game Area to the Game Preview View of the game associated to the selected level (Figure 2 ③). Then, the Game autonomously listens to all user input events on its views. When players are ready to start, they click the button on the Preview view, and the Game asks the Game Master to switch view to the InGame view (Figure 2 ④), which in turn, behaves autonomously. The Game Master is idle until the Game requests the end of the in-game phase. The Game Master then collects information provided by the Game to query the remote Model Quality Assessor to update the player profile and to know whether it should display the *GameSuccess* or *GameOver* view. From these two views, the player can ask to resume the Dashboard. When this event is triggered, the Game asks the Game Master to switch view to the Dashboard to initiate a new Dashboard phase.

3. Illustrative examples

The role of PapyGame is to help *master degree students* in Computer Science in learning specific modeling aspects using Papyrus for UML. Each assignment comprises a set of levels that each student should deal with. Each level is assigned a corresponding exercise, and passing a level unlocks the next exercise corresponding to the next level. Once logged-in, PapyGame displays a Dashboard (see Figure 2 ②) representing the levels for the player. All the completed levels are displayed in green with the corresponding number of gold coins (GC) and the experience points (XP) rewarded. Instead, the remaining levels are colored in gray with a lock except for the first one, which is the next level to be played (unlocked). When students start a game, the UI shows the game-specific content, such as the purpose of the game and the progression.

Teachers can define custom series of exercises by defining one or several levels using the following procedure: (i) choose a game among the available games in the system, (ii) create a source diagram in Papyrus according to the game requirements, (iii) create the reward rules about points and eventual bonuses (using GDF [18]), and (iv) write the high-level JSON definition of the Level.

Each *learning exercise* is associated with a specific game type: the `Hangman` — when a new part of the man drawing is added with every wrong answer — the `On Your Own (OYO)` — when the student executes the exercise with no help. At the same time, each exercise has associated with a set of point concepts and rules. All these aspects are defined by the teacher using GDF [18]. Once the student's assignments are designed and deployed, the students can select the respective levels and start to play and accumulate points. Figure 3 shows an example. It presents the UML diagram containing a set of classes connected with the generalization relationship. The goal of this level is to help players/students to associate the right attributes and operations with the right class in the hierarchy. This is done using a drag-and-drop facility. An incorrect user selection (moving an operation into a class that is not the one that should contain it) adds a part of the hangman's body (lower part of Figure 3). If the players manage to place all operations correctly without the body of the hanged person being completely displayed, they win. If the hangman's body is wholly displayed, players lose, and the next level stays locked. As a consequence, they will have to play the level again. Whether they won or lost, after the completion of the game PapyGame players are returned to the Dashboard view.

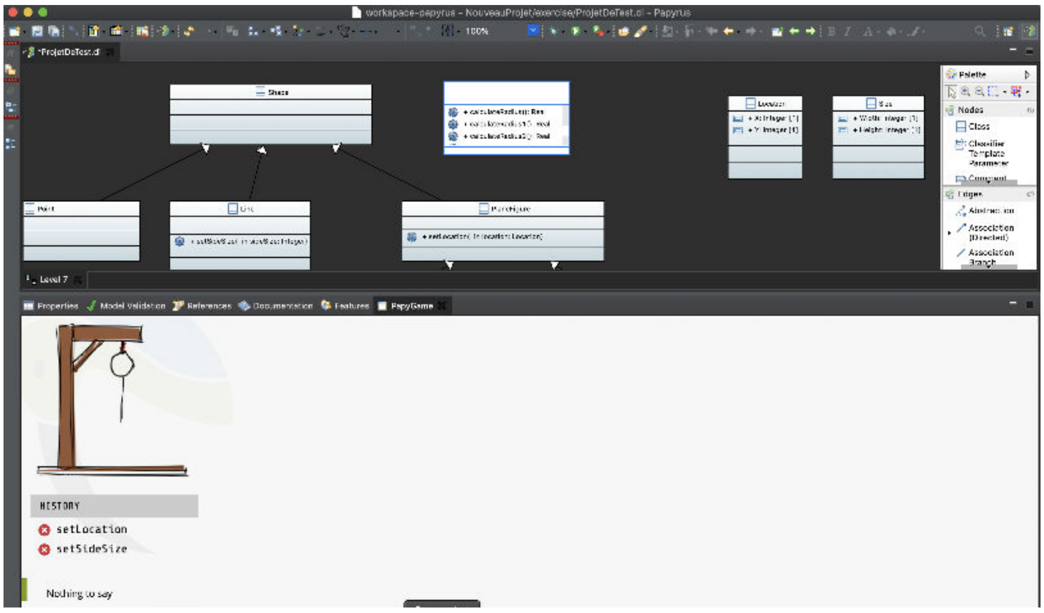


Figure 3: PapyGame Hangman game example.

4. Impact

One of the significant hurdles for more mainstream adoption of modeling practices is the limitations in current modeling tools, as reported in several studies [20, 21, 22, 23, 24]. As such, the introduction of gamification techniques in modeling environments, which is the goal of PapyGame, could have a significant impact on the popularization of modeling and its benefits.

By lowering the barrier to entry of modeling, PapyGame aims to broaden the user base of modeling environments, especially, students and non-expert users that so far were unable to see the return of inversion of this type of tools given the high learning cost they were suffering [23]. This will also enable a better understanding of the actual challenges in learning modeling techniques that could be obscured by these tool limitations.

In this sense, PapyGame has been successfully applied in a set of scenarios that includes software engineering teaching courses at the Universities of Trento (Italy) (~ 150 students) or specific user experiences analysis with 14 students from the University of Lille (France) and with 25 students from Mälardalen University (Sweden).

In a previous article [25], we conducted an evaluation of PapyGame using a user experience questionnaire to exhaustively assess our approach. We made rational design choices based on serious and validated works, and the questionnaire was designed to collect generic data such as age, gender, and country, as well as to differentiate between the two groups. The questionnaire also included an evaluation of several aspects related to the user experience, such as motivation, engagement, attractiveness, and learning perception, as well as software usability and perceived efficacy. Users were asked to provide feedback on what was working well and what needed to be changed in the software. While the tool showed adequate levels during user experience analysis, it also presented some disturbing elements, including elements of UI, the information provided, and the complexity and problems related to the Eclipse environment.

PapyGame impact goes beyond "traditional" modeling environments and has proven useful also as a key component of new learning environments pushing the boundaries of current interfaces [26].

5. Conclusions and Future Plans

The learning curve concerning modeling is, unfortunately, quite high and partly prevents a wide adoption of modeling in IT projects. The reasons for this are threefold: modeling is in itself a complex act requiring capacities of abstraction and synthesis that are not always easy to learn and teach; modeling is also communicating through appropriate languages and notations that often need to be learned and therefore require an effort; finally, the modeling tools, since they are often new, are also perceived as complex.

Our paper focuses on the technical presentation of a tool, PapyGame, that gamifies the teaching and learning process. We aim to boost the educational experience by incorporating game elements into the curriculum. Our work is centered around presenting the technical aspects of this innovative tool and how it can be used to enhance the learning process. In terms of impact, our goal was to experience and experiment with this gamification in a real modeling environment and with real learners to evaluate the interest of such an approach.

We chose Papyrus, the open-source reference environment for UML modeling in Eclipse. The feedback from the students involved in this first experience is very encouraging but also shows room for improvement (new games, a more constrained environment, and better management of technical issues) and identifies areas for improvement in the approach to gamifying software modeling teaching process. At the same time, PapyGame helped us to start planning the adaptation of PapyGame to cover low-code environments. This means going beyond the gamification of modeling and extending it to the gamification of a low-code/model-driven full application generation. At the same time, another direction would be to apply to modeling challenges beyond systems and software modeling [27].

In terms of supported games, we have plans to extend PapyGame beyond the current model of games composed of levels. Our goal is to support *personalized learning paths* that are tailored to the specific needs of individual students and aligned with the objectives of their teachers. This approach would enable teachers to design learning experiences that are engaging, challenging, and relevant to each student's unique learning style and interests. By incorporating a variety of game mechanics and feedback mechanisms, we aim to create a more dynamic and interactive learning environment that encourages students to take an active role in their own learning. Ultimately, our goal is to help students achieve their full potential and develop a lifelong learning.

Teaching modeling is a complex task, and there are limitations to automated support and evaluation approaches. In this regard, we are aware that one of the limitations of PapyGame is its current lack of support for evaluating model quality (correctness and completeness). To address this limitation, we plan to add specific components as a plugin devoted to supporting different types of model quality evaluation. This plugin will extend the existing component, the *Model Quality Assessor*, which evaluates the syntax and the semantics of a model. We believe that this addition will make PapyGame more useful for educators and learners.

References

[1] J. Koivisto, J. Hamari, The rise of motivational information systems: A review of gamification research, International Journal of Information Management 45 (2019) 191 – 210.

[2] D. Dicheva, C. Dichev, K. Irwin, E. J. Jones, L. B. Cassel, P. J. Clarke, Can game elements make computer science courses more attractive?, in: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE 2019, 2019, p. 1245.

[3] V. Cosentino, S. Gérard, J. Cabot, A model-based approach to gamify the learning of modeling, in: Proceedings of the 5th Symposium on Conceptual Modeling Education and the 2nd International iStar Teaching Workshop co-located with the 36th International Conference on Conceptual Modeling (ER 2017), Valencia, Spain, November 6-9, 2017., 2017, pp. 15–24.

[4] M. Jurgelaitis, L. Čeponienė, J. Čeponis, V. Drungilas, Implementing gamification in a university-level uml modeling course: A case study, Computer Applications in Engineering Education 27 (2) (2019) 332–343.

[5] M. Jurgelaitis, V. Drungilas, L. Ceponiene, Gamified moodle course for teaching UML, Balt. J. Mod. Comput. 6 (2) (2018).

[6] H. Stuart, A. Serna, J.-C. Marty, E. Lavoué, Adaptive gamification in education: A literature review of current trends and developments, in: European Conference on Technology Enhanced Learning (EC-TEL), Delft, Netherlands, 2019.

[7] N. Z. Legaki, N. Xi, J. Hamari, V. Assimakopoulos, Gamification of The Future: An Experiment on Gamifying Education of Forecasting, 2019.

[8] B. Morschheuser, J. Hamari, The Gamification of Work: Lessons From Crowdsourcing, Journal of Management Inquiry 28 (2) (2019) 145–148.

[9] T. Jagušt, I. Botički, H.-J. So, Examining competitive, collaborative and adaptive gamification in young learners' math learning, Computers & Education 125 (2018) 444–457.

[10] J. De Smedt, J. De Weerdt, E. Serral, J. Vanthienen, Gamification of declarative process models for learning and model verification, in: M. Reichert, H. A. Reijers (Eds.), Business Process Management Workshops, Springer International Publishing, Cham, 2016, pp. 432–443.

[11] N. Pflanzl, Gameful business process modeling, in: J. Mendling, S. Rinderle-Ma (Eds.), Proceedings of the 7th International Workshop on Enterprise Modeling and Information Systems Architectures, EMISA 2016, Vol. 1701 of CEUR Workshop Proceedings, CEUR-WS.org, 2016, pp. 17–20.

[12] O. C. Tantan, D. Lang, I. Boughzala, Towards gamification of the data modeling learning.

[13] G. Sedrakyan, M. Snoeck, Technology-enhanced support for learning conceptual modeling, in: I. Bider, T. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, S. Wrycza (Eds.), Enterprise, Business-Process and Information Systems Modeling, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 435–449.

[14] M. R. de A. Souza, L. Veado, R. T. Moreira, E. Figueiredo, H. Costa, A systematic mapping study on game-related methods for software engineering education, Information and Software Technology 95 (2018) 201–218.

[15] R. Oberhauser, VR-UML: the unified modeling language in virtual reality - an immersive modeling experience, in: B. Shishkov (Ed.), Business Modeling and Software Design - 11th International Symposium, BMSD 2021, Sofia, Bulgaria, July 5-7, 2021, Proceedings, Vol. 422 of Lecture Notes in Business Information Processing, Springer, 2021, pp. 40–58. `doi:10.1007/978-3-030-79976-2\_3`.
URL `https://doi.org/10.1007/978-3-030-79976-2_3`

[16] E. Yigitbas, S. Gorissen, N. Weidmann, G. Engels, Design and evaluation of a collaborative uml modeling environment in virtual reality, Journal on Software and Systems Modeling (SoSyM) (2022).

[17] E. Yigitbas, M. Schmidt, A. Bucchiarone, S. Gottschalk, G. Engels, Gamification-based uml learning environment in virtual reality, in: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 27–31. `doi:10.1145/3550356.3559088`.
URL `https://doi.org/10.1145/3550356.3559088`

[18] A. Bucchiarone, A. Cicchetti, A. Marconi, GDF: A gamification design framework powered by model-driven engineering, in: 22nd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion, MODELS Companion 2019, Munich, Germany, September 15-20, 2019, IEEE, 2019, pp. 753–758.

[19] P. Herzig, B. Wolf, S. Brunstein, A. Schill, Efficient persistency management in complex event processing: A hybrid approach for gamification systems, in: L. Morgenstern, P. S. Stefaneas, F. Lévy, A. Z. Wyner, A. Paschke (Eds.), Theory, Practice, and Applications of Rules on the Web - 7th International Symposium, RuleML 2013, Seattle, WA, USA,

July 11-13, 2013. Proceedings, Vol. 8035 of Lecture Notes in Computer Science, Springer, 2013, pp. 129–143.

[20] E. Planas, J. Cabot, How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and papyrus, Comput. Stand. Interfaces 67 (2020).

[21] J. Whittle, J. E. Hutchinson, M. Rouncefield, H. Burden, R. Heldal, A taxonomy of tool-related issues affecting the adoption of model-driven engineering, Software and Systems Modeling 16 (2) (2017) 313–331.

[22] T. Weber, A. Zoitl, H. Hußmann, Usability of development tools: A case-study, in: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2019, pp. 228–235.

[23] J. Cabot, D. S. Kolovos, Human factors in the adoption of model-driven engineering: An educator's perspective, in: S. Link, J. Trujillo (Eds.), Advances in Conceptual Modeling - ER 2016 Workshops, Gifu, Japan, November 14-17, 2016, Proceedings, Vol. 9975 of Lecture Notes in Computer Science, 2016, pp. 207–217.

[24] A. Bucchiarone, J. Cabot, R. F. Paige, A. Pierantonio, Grand challenges in model-driven engineering: an analysis of the state of the research, Software and Systems Modeling 19 (1) (2020) 5–13.

[25] A. Bucchiarone, M. Savary-Leblanc, X. Le Pallec, A. Cicchetti, S. Gérard, S. Bassanelli, F. Gini, A. Marconi, Gamifying model-based engineering: the PapyGame experience, Software and Systems Modeling (Mar. 2023).

[26] E. Yigitbas, M. Schmidt, A. Bucchiarone, S. Gottschalk, G. Engels, Gamification-based UML learning environment in virtual reality, in: T. Kühn, V. Sousa (Eds.), Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS 2022, Montreal, Quebec, Canada, October 23-28, 2022, ACM, 2022, pp. 27–31.

[27] J. Cabot, A. Vallecillo, Modeling should be an independent scientific discipline, Software and Systems Modeling (2022) 1–7.