Papyrus for gamers, let's play modeling

Antonio Bucchiarone
Fondazione Bruno Kessler, Trento, Italy
bucchiarone@fbk.eu

Maxime Savary-Leblanc
University of Lille, CEA LIST, France
maxime.savary-leblanc@univ-lille.fr

Xavier Le Pallec
CRIStAL, IRCICA, University of Lille, France
xavier.le-pallec@univ-lille.fr

Jean-Michel Bruel
IRIT, University of Toulouse, France
bruel@irit.fr

Antonio Cicchetti
IDT Department, Mälardalen University, Västerås, Sweden
antonio.cicchetti@mdh.se

Jordi Cabot
ICREA, UOC, Spain
jordi.cabot@icrea.cat

Sebastien Gerard
CEA-LIST, France
sebastien.gerard@cea.fr

Hamna Aslam
IRIT, Innopolis University, Russia
h.aslam@innopolis.ru

Annapaola Marconi
Fondazione Bruno Kessler, Trento, Italy
marconi@fbk.eu

Mirko Perillo
Fondazione Bruno Kessler, Trento, Italy
mirko.perillo@fbk.eu

ABSTRACT

Gamification refers to the exploitation of gaming mechanisms for serious purposes, like learning hard-to-train skills such as modeling.

We present a gamified version of Papyrus, the well-known open source modeling tool. Instructors can use it to easily create new modeling games (including the tasks, solutions, levels, rewards...) to help students learning any specific modeling aspect.

The evaluation of the game components are delegated to the GDF gamification framework that bidirectionally communicates with the Papyrus core via API calls. Our gamified Papyrus includes as well a game dashboard component implemented with HTML/CSS/Javascript and displayed thanks to the the integration of a web browser embedded in an Eclipse view.

KEYWORDS

Software Modeling, Modeling Education, Gamification, Papyrus

ACM Reference Format:
Antonio Bucchiarone, Maxime Savary-Leblanc, Xavier Le Pallec, Jean-Michel Bruel, Antonio Cicchetti, Jordi Cabot, Sebastien Gerard, Hamna Aslam, Annapaola Marconi, and Mirko Perillo. 2020. Papyrus for gamers, let's play modeling. In *MODELS '20: ACM / IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS), Companion, October 18–23, 2020, Montreal, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

MODELS '20, October 18–23, 2020, Montreal, Canada

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION AND MOTIVATIONS

While modeling is now validated as effective for solving complex problems or developing complex systems, its widespread and systematic application has not yet lived up to expectations. Several studies have been led to analyse the reasons of this situation and highlighted several issues, including the complexity of the modeling languages and tools [1–3] and the lack of required skills in abstraction [4].

Indeed, in many studies around Model-Based Systems Engineering (MBSE) adoption, usability of tools is very often stressed as one of the key issues for the adoption of MBSE paradigms. We do agree of course that user experience of tools can be improved, and lot of people are working hard to improve including as for example new HCI or even AI-empowered assistants. However, we are definitively also convinced that it is fundamental to assist users to face the complexity - accidental or essential - of MBSE related artefacts by augmenting the MBSE tools with self-training facilities. In this way, it could be possible to reduce tools learning curve and encourage their users to persist with them despite the required initial efforts. Indeed, it is to be expected that tools, languages, etc. used to solve complex problems will remain complex, even if their user experience is optimised. As a consequence, it is important to provide additional features supporting the learning.

In this demo, we illustrate the role gamification could play to lower the entry barrier of modeling and modeling tools. Gamification is the exploitation of gaming mechanisms for serious purposes, like promoting behavioral changes, soliciting participation, and engagement in activities. Gamification is gaining popularity in all those domains that would benefit from the increased engagement of their target users [5]. Therefore, disparate contexts use gamification applications, such as education [6–8], health and environment [9, 10], e-banking [11], and even software engineering [12].

We present a gamified software modeling environment realized to design games for specific training/learning goals. In particular we present how using it, the Papyrus[1] modeling tool has been gamified with the objective to help students learning specific modeling aspects using both UML and Papyrus for UML. Through a concrete case study, we demonstrate that our approach is applicable with limited effort, does not require any major modifications in the existing modeling environment, neither introduces complex dependencies. In Section 2 we detail the main components of the gamified software modeling environment and in Section 3 we illustrate its application in a concrete software design courses where a "Hangman Game" is used to improve the learning curve.

2 THE GAMIFIED SOFTWARE MODELING ENVIRONMENT

This section describes the main components of the gamified software modeling environment. In this respect, a typical use case of the environment might include the design of a game based on specific training/learning goals, which is linked to corresponding modelling tasks to be accomplished by users/players using Papyrus. Subsequently, on one side relevant modelling actions need to be captured in Papyrus and communicated externally, notably when the player (i.e., student) commits a completed exercise. On the other side, the game has to be encoded into corresponding gamification elements, such that it is possible to keep track of players' progresses. With respect to this, it is necessary to create gaming actions, points, rules, etc. interconnected to the modelling events. In this way, players will operate in Papyrus and their modelling activities will be translated into game-meaningful events (e.g., a completed exercise, a mistake, etc.). In turn, the events will track the progress of the players, give points/bonuses, updated level, and so forth.

By going into more details, the gamified version of Papyrus is composed of seven components, illustrated in Figure 1. In the following we explain each component, their relations, and how the different stakeholders (i.e., Gamification Expert, Modelling Teacher, and the Students) are involved in performing specific activities.

Game UI. Students interact with the gamification plugin through different views embedded in the Papyrus environment. The entry point to gamified Papyrus is the login view where students are asked to provide their username and, eventually, their password. Once logged in, they access the game dashboard view. This interface provides typical user account management features, and access to available games and player status (i.e., history, achievements, and progresses). Moreover, the dashboard maintains player profiles including avatars, level of expertise, and so forth, which can be exploited to enhance user experience. When students start a game, the dashboard is temporarily replaced with game-specific content such as the purpose of the game and the progression.

Papyrus Modeling Tool. Students interact with Papyrus in the form of modeling operations to complete the exercises assigned for a certain game. Since modeling actions and game progress are two separate concerns, modeling actions generated by students need to be caught, evaluated, and translated into corresponding game events. To do so, our plugin implements model listeners to catch modelling actions and retrieve model elements from the Papyrus editor. This data is sent to the *Model Comparator* for evaluation. In a process described later in the paper, this component assesses the correctness of player's action or model. Evaluation results are finally sent to the *Gamification Engine* to make the current game status to advance. Catching and storing players' actions discloses also other interesting uses, notably game replay and post-game analysis.

Gamification Design Framework. Each modelling game should be designed to target specific learning objectives, while at the same time keeping learners (i.e. students) motivated in pursuing their learning goals. For this purpose, the gameful aspects of the modeling tasks in Papyrus should be conceived to promote users' engagement. These aspects could include awarding points and rewards.

When the game grows in its complexity, keeping track of all the mechanisms and maintaining the implementation can become error-prone and tedious activities. To tackle this issue, we exploit a model-driven gamification design framework called GDF [13]. GDF provides a modular approach that allows for the specification of gameful systems with different degrees of customization; in particular, gamification solutions can be built-up from pre-existing mechanisms (e.g. by naming appropriately actions, points, bonuses, levels, etc.), and refining available game rules (e.g. actions triggering points, thresholds to complete the current level, and so on). Otherwise, game designers can partially/completely replace the mechanisms by defining ad-hoc ones tailored to a specific family of new gamification solutions (e.g. a bonus mechanism that awards faster players, so that decreases while time passes). In both scenarios, GDF provides support for automatically deploying the games in a target gamification engine. Left part of Figure 2 shows, how using GDF, the rules regulating the game behavior can be defined using a specific MPS editor[2]. The rules in this case specify the number of `points` (i.e., 20) and `gold coins` (i.e., 2) are accumulated when a game action `taskCompleted` is executed by a student with 5 `moves` and 0 `errors`. GDF includes code generation features that generate DROOLS [3] files corresponding to the game rules and their deployment in the gamification engine. The right part of Figure 2 shows the DROOLS code deployed in the gamification engine.

Gamification Engine. Once a game is appropriately defined, it has to executed. Indeed, the gamification solutions specified by means of GDF include players and game logic, where the latter is usually expressed in terms of rules over players' status and incoming events. Therefore, these specifications need to be taken in charge by a mechanism able to handle the game status, that is manage players', their actions, and their progression through the game. Technically, this is accomplished by deploying the game specification on a selected gamification engine (GE). Consistently, the GE component takes care of both running games by animating the logic as defined in GDF, and keeping updated the game status, including its storage, for all players.

More specifically, the GE component embeds the Open Source DROOLS rule engine, a state-of-the-art rule engine technology based on reactive computing models [14]. The GE is also equipped

[1] https://www.eclipse.org/papyrus/

[2] GDF is realised by means of JetBrains MPS - http://www.jetbrains.com/mps - and available at the following link: https://github.com/antbucc/GDF

[3] https://www.drools.org/

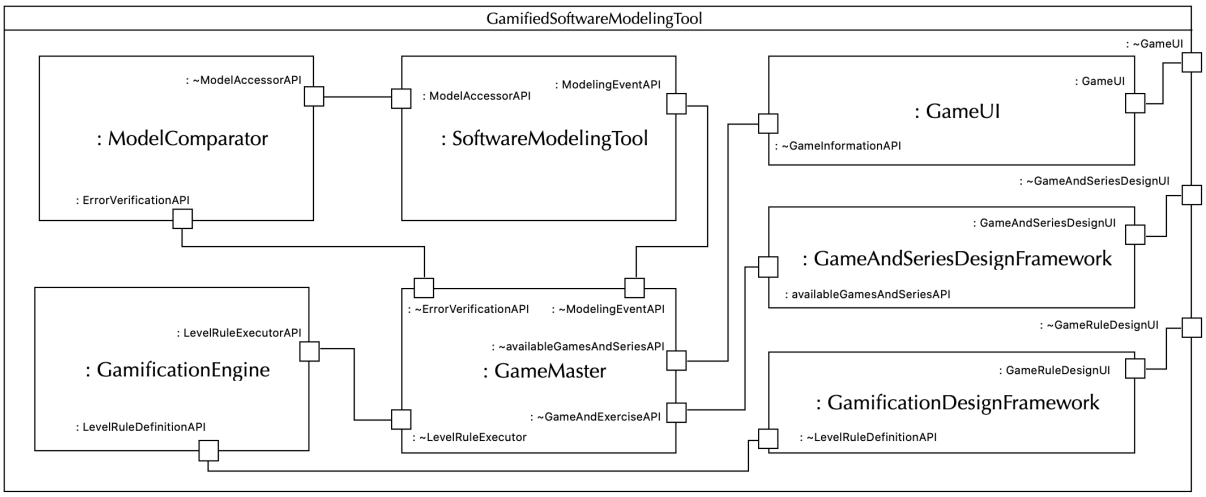

Figure 1: UML Composite Structure Diagram of the gamified software modeling environment.

```
rules file PapyGame-Rules

/* Global variables */
<< ... >>

/* Rules */
rule UpdateToLevel1-5Moves
  when
    inputAction : Action       ( getName() == "taskCompleted"                          )
    pointState1 : PointConcept ( pointState1.name == "errors" && pointState1.getScore() == 0.0 )
    levelState  : Level        ( levelState.getScore() == 0                            )
    pointState2 : PointConcept ( pointState2.name == "moves" && pointState2.getScore() == 5.0  )
  then
    levelState.update(levelState.getScore() + 1);
end

rule RewardsToLevel1-5Moves
  when
    levelState  : Level        ( levelState.getScore() == 1                            )
    pointState1 : PointConcept ( pointState1.name == "moves" && pointState1.getScore() == 5.0  )
    pointState2 : PointConcept ( pointState2.name == "errors" && pointState2.getScore() == 0.0 )
    pointState3 : PointConcept ( pointState3.name == "points"                          )
    pointState4 : PointConcept ( pointState4.name == "gold coins"                      )
  then
    pointState3.updateScore(pointState3.getScore() + 20);
    pointState4.updateScore(pointState4.getScore() + 2) ;
end
```

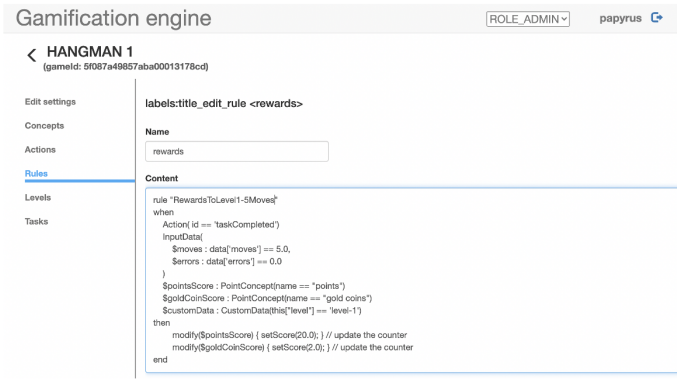

Figure 2: Game Rules Definition and Deployment with GDF.

with a uniform service interface (i.e., REST API) that gives access to the internally persisted game status. Notably, in this way the game dashboard is able to retrieve and show the status of each player.

Model Comparator. In order to apply gamification principles to modelling, it is necessary to assign points and rewards to players according to the model they create. To this end, these models must be analyzed to determine whether they meet the instructions of a given game. Based on the results of this analysis, the rules of the game define the number of points and bonuses that the player will be rewarded. The *Model Comparator* is responsible for performing such analysis by assessing the correctness of modelling actions, or complete diagrams, over a reference: the source diagram. According to the game, this evaluation is either performed in real time or at the end of the exercise. In the first case, any action on the model triggers a request to the Model Comparator. This component then verifies that the result of the action matches an element in the source diagram, which represents the correction of the modelling exercise. In the second case, the Model Comparator compares both source diagram and player diagram to check for differences. No difference is interpreted as a correct answer to the exercise. In addition to the comparison task, this component is able to check if a model verifies defined constraints (minimal number of elements, specific necessary relationships, elements with specific names). Moreover, the Model Comparator could be used as a tool to measure the progression of a player in a specific game, e.g. to introduce some additional tasks if the player is proceeding faster than expected, or vice versa to provide hints if the player is stuck with a particular problem.

Regardless of the complexity of the evaluation task, having a separate component for the Model Comparator brings important benefits: the logic of the games does not get intertwined with proposed modelling tasks, enabling the creation of games tailored to specific learning objectives; the evaluation of the models becomes tool independent, making it possible to create a library of modelling exercises and corresponding solutions. Furthermore, this latter avoids re-implementing the assessment for each modeling tool.

Game Master. The Gamified Papyrus features a Game Master whose role is similar to that of many board games: set up the game, apply its mechanics and rules, and inform the players about the state

of the game. When the player starts a game level, the Game Master sets up the Papyrus environment and the game-specific views in order to be ready to play. During the game, it is responsible for applying the game mechanics and rules, by displaying information or preventing certain actions. The Game Master monitors the game execution through notifications incoming from Papyrus modeling tool and, eventually, calls the Model Comparator to evaluate modelling actions. In this way, it can detect undesired game patterns, e.g. too fast or too slow players, and adopt corresponding countermeasures, e.g. add new difficulties or provide hints, respectively. At the end of the game, the Game Master sends the results of the game to the Gamification Engine, which applies the rewarding rules and calculates the player's rewards based on the results of the Model Comparator evaluation(s). Finally, the Game Master notifies the player's progress by displaying the game dashboard.

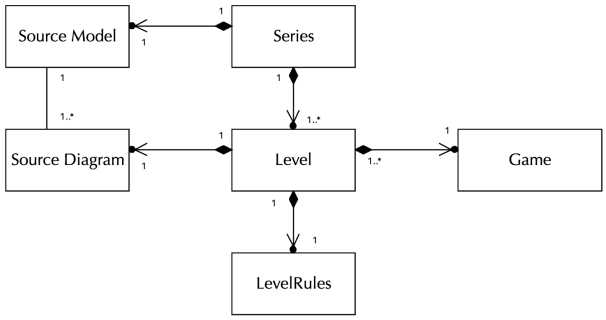

Figure 3: Level Model Artifacts.

Game and Series Design Framework. In order to customize the gamified Papyrus, our system provides a high-level framework allowing to define (i) custom games, and (ii) custom series of exercise.

A `Game` consists of a set of java classes representing the operating rules of the game within the Papyrus environment, and HTML, CSS and JS files representing the views of the game. Operating rules define the Game Master behaviour such as the set up operations required prior to start a game, the in-game behavior and the game ending conditions. The modular architecture of our system provides high-level functions and callbacks to exploit modelling information without the expertise of the Papyrus environment. The views, defined in HTML, CSS and JS, are automatically bound to specific view java code, what allows a simplified definition of the game visual behavior. Developers can therefore rely upon a simplified framework to create games within our system that might be exploited by teachers in new series of exercises.

In order to get access to the exercises, students must register to one or more series. As described in Figure 3, a `Series` is defined by a set of `Levels` and a `SourceModel`. A `Level` refers to one `Game` associated to a `SourceDiagram` from the `SourceModel` and linked to `LevelRules` defined in GDF. Teacher can define custom series of exercises by defining one or several levels following this approach: (i) choose a game among the available games in the system, (ii) create a source diagram in Papyrus according to the game requirements about source diagrams, (iii) create the reward rules about points and eventual bonuses in GDF, and (iv) write the high-level

```
{
  "name": "Bachelor of Software Engineering",
  "seriesGameId": "5f087a49857aba00013178cd", 1
  "levels": [
    {
      "label": "Level 1",
      "modelPath": "/responses/Gamification/Library", 4
      "diagramName": "Level 1", 3
      "gameName": "org.eclipse.papyrus.gamification.games.oyo.Oyo", 2
      "statement": "Nothing to say" 5
    },
    {
      "label": "Level 2",
      "modelPath": "/responses/Gamification/Library",
      "diagramName": "Level 1",
      "gameName": "org.eclipse.papyrus.gamification.games.oyo.Oyo",
      "statement": "Nothing to say"
    },
    {
      "label": "Level 3",
      "modelPath": "/responses/Gamification/Library",
      "diagramName": "Level 2",
      "gameName": "org.eclipse.papyrus.gamification.games.hangman.Hangman",
      "statement": "Nothing to say"
    },
    {
      "label": "Level 4",
      "modelPath": "/responses/Gamification/Library",
      "diagramName": "Level 2",
      "gameName": "org.eclipse.papyrus.gamification.games.hangman.Hangman",
      "statement": "Nothing to say"
    },
```

Figure 4: JSON Description of a Series.

JSON[4] definition of the `Level` (as presented in Figure 4). Each level connects to the Gamification Engine rules thanks to a specific id (1) held by the series, references a game (2), a source diagram (3) from a source model (4), and features textual instructions (5).

3 THE PAPYGAME DEMONSTRATION

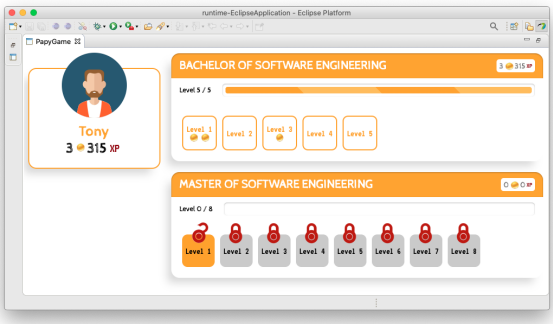

Figure 5: Dashboard of PapyGame.

This scenario takes place in a software design course for master students. In this course, the main elements of the learning process are lectures and an assignment to submit. Students are taught concepts and principles during lectures, and are required to apply this new knowledge through one assignment. Additional exercises are available for them for practice more according to this rule : the more a student practices and succeeds, the simpler his assignment

[4]https://www.json.org

will be. Such exercises are gamified according to 3 mechanisms. The first one is based on the use of game mechanisms such as the *Hangman* or *On Your Own*. The second mechanism is based on obtaining experience (XP) and gold coins. Each passed level unlocks the next exercise and brings its share of XP and gold coins. 10 gold coins allow the student to remove one « thing » to be done in their assignment. The accumulated XP can be used to buy additional gold coins. The third mechanism refers to the virtual teacher who explains the rules and exercises and congratulates students on their performance.

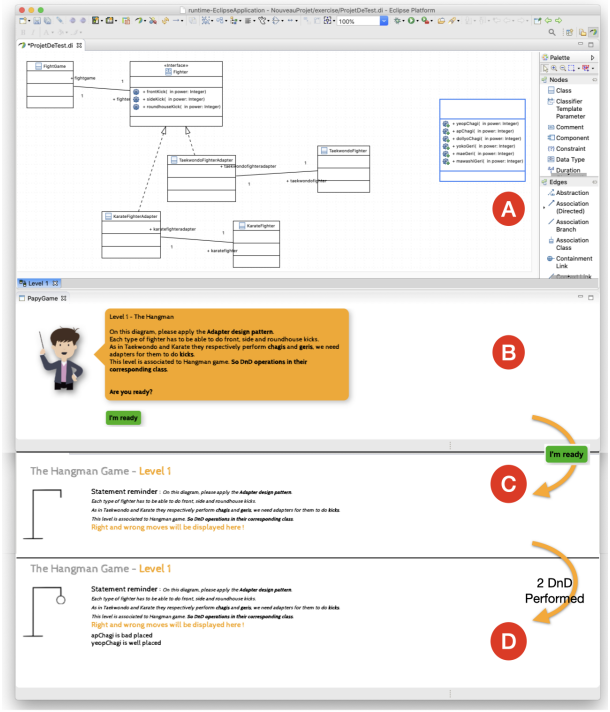

Figure 6: Playing Hangman (Level 1).

To start a PapyGame session, the player must first enter their login and password. Once connected, PapyGame displays a Dashboard (see figure 5) representing the series of the player. A series consists of a succession of *levels* which each feature a new exercise. We can see that the player subscribed to a series during their Bachelor course (B.SE) and that they are currently progressing within a series associated with their Master (M.SE) For each series, all levels are displayed. The completed (successfully) ones are displayed in orange with the corresponding number of gold coins rewarded. Remaining levels are colored in grey with a lock except for the first one which is the next level to be played (unlocked). The Dashboard shows on the left the amount of XP and gold coins accumulated. To play an unlocked level, the player can click it to start the loading of its associated game. Figure 6 shows the loading of level 1 of the Master series. This is the Hangman game with the associated diagram containing one interface and five classes (part A of figure 6). The goal of this level is to help player/student to practice the Adapter design pattern. An interface (*Fighter*) is defined with 3 operations (all related to *kicking*). A blue no-name class contains six operations: three about kicking in Karate and three others in Taekwondo. The diagram contain four other classes: one for Karate fighter and its associated adapter, and the same for Taekwondo. The player has to indicate - with drag-n-drop - in which classes these 6 operations has to be: in the adapters or in the *original fighter* classes? The statement of the level (part B of the figure 6 gives clues to meet this challenge. A bad drag-n-drop (moving an operation into a class that is not the one that should contain it) adds a part of Hangman's the body (part D of Figure 6). Once the player has understood the rules and clicked on « I'm ready », the game starts with a new view displaying the gallows and a box where the correctness of each DnD will be displayed (part C of the figure 6). If the player manages to place all operations correctly without the body of the hanged person being completely displayed, they win. The number of gold coins and XP is calculated according to the number of errors (bad drag-n-drops). If the hangman's body is completely displayed, the player looses, and the next level is not unlocked. They will, therefore, have to play this level again. Whether they had won or lost, PapyGame returns to the DashBoard view. To install the PapyGame and see it in action, with an illustrative video, you can visit our website at https://www.cristal.univ-lille.fr/miny/papygame/.

REFERENCES

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

[2] J. Whittle, J. E. Hutchinson, M. Rouncefield, H. Burden, and R. Heldal, "A taxonomy of tool-related issues affecting the adoption of model-driven engineering," *Software and Systems Modeling*, vol. 16, no. 2, pp. 313–331, 2017.

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

[4] A. Bucchiarone, J. Cabot, R. F. Paige, and A. Pierantonio, "Grand challenges in model-driven engineering: an analysis of the state of the research," *Software and Systems Modeling*, vol. 19, no. 1, pp. 5–13, 2020.

[5] J. Koivisto and J. Hamari, "The rise of motivational information systems: A review of gamification research," *Int. Journal of Information Management*, vol. 45, pp. 191 – 210, 2019.

[6] D. Dicheva, C. Dichev, K. Irwin, E. J. Jones, L. B. Cassel, and P. J. Clarke, "Can game elements make computer science courses more attractive?," in *Proc. of the 50th ACM Tech. Symp. on Computer Science Education, SIGCSE 2019*, p. 1245, 2019.

[7] V. Cosentino, S. Gérard, and J. Cabot, "A model-based approach to gamify the learning of modeling," in *Proc. of the 5th Symp. on Conceptual Modeling Education at ER 2017, Valencia, Spain, November 6-9, 2017.*, pp. 15–24, 2017.

[8] J. J. Lee and J. Hammer, "Gamification in education: What, how, why bother?," *Academic Exchange Quarterly*, vol. 15, no. 2, p. 2, 2011.

[9] D. Johnson, S. Deterding, K.-A. Kuhn, A. Staneva, S. Stoyanov, and L. Hides, "Gamification for health and wellbeing: A systematic review of the literature," *Internet Interventions*, vol. 6, pp. 89–106, 2016.

[10] A. Marconi, G. Schiavo, M. Zancanaro, G. Valetto, and M. Pistore, "Exploring the world through small green steps: improving sustainable school transportation with a game-based learning interface," in *Proc. of the 2018 Int. Conf. on Advanced Visual Interfaces, AVI 2018*, pp. 24:1–24:9, 2018.

[11] L. F. Rodrigues, C. J. Costa, and A. Oliveira, "Gamification: A framework for designing software in e-banking," *Computers in Human Behavior*, vol. 62, pp. 620 – 634, 2016.

[12] 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.

[13] A. Bucchiarone, A. Cicchetti, and A. Marconi, "Exploiting multi-level modelling for designing and deploying gameful systems," in *22nd Int. Conf. on Model Driven Engineering Languages and Systems, MODELS 2019, Munich, Germany, September 15-20, 2019*, pp. 34–44, IEEE, 2019.

[14] P. Herzig, B. Wolf, S. Brunstein, and A. Schill, "Efficient persistency management in complex event processing: A hybrid approach for gamification systems," in *Theory, Practice, and Applications of Rules on the Web - 7th Int. Symp., RuleML 2013, Seattle, WA, USA, July 11-13, 2013. Proceedings*, vol. 8035 of *LNCS*, pp. 129–143, Springer, 2013.