POLITECNICO DI TORINO
Repository ISTITUZIONALE

Enhancing a gamified tool for UML modeling education

Original
Enhancing a gamified tool for UML modeling education / Garaccione, G., Coppola, R., Ardito, L.. - ELETTRONICO. - (In corso di stampa). (30th International Conference on Evaluation and Assessment in Software Engineering Glasgow, UK 09/06/2026-12/06/2026).

Availability:
This version is available at: 11583/3010410 since: 2026-04-29T15:44:54Z

Publisher:
Association for Computing Machinery

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
ACM postprint/Author's Accepted Manuscript

(Article begins on next page)

17 October 2026

Enhancing a gamified tool for UML modeling education

Giacomo Garaccione
Politecnico di Torino
Turin, Italy
giacomo.garaccione@polito.it

Riccardo Coppola
Politecnico di Torino
Turin, Italy
riccardo.coppola@polito.it

Luca Ardito
Politecnico di Torino
Turin, Italy
luca.ardito@polito.it

arXiv:2604.22400v1 [cs.SE] 24 Apr 2026

ABSTRACT

Unified Modeling Language (UML) Use Case and Class Diagrams are fundamental modeling notations in Software Engineering (SE) education due to their importance for requirements and model-based engineering, yet their relevance is underestimated by students, who tend to dismiss the topic as secondary. Gamification has been adopted to make modeling education more appealing, but existing tools focus almost exclusively on class diagrams, leaving support for use cases and other notations unexplored. In 2025, we designed UMLegend, a gamified tool for class diagrams that offered dynamic feedback to help students learn correct modeling practices and multiple long-term mechanics to increase engagement, and performed a study with the tool. With this paper, we describe how we enhanced UMLegend following the results of the experiment so that it can support more modeling languages, with use case diagrams being added to the type of available exercises in the tool. The revised version has been refactored to have a modular architecture, to make it easier to add other software engineering topics and additional modeling notations. We also describe the potential impact we expect the new version to have, and outline a longitudinal study we intend to perform in 2026 where we will assess whether long-term UML gamification leads to improved student performance.

CCS CONCEPTS

• **Software and its engineering** → **Requirements analysis**; *Object oriented development*; **Unified Modeling Language (UML)**.

KEYWORDS

Gamification, Software Engineering Education, Model-Based Engineering, Requirements Engineering

ACM Reference Format:

Giacomo Garaccione, Riccardo Coppola, and Luca Ardito. 2026. Enhancing a gamified tool for UML modeling education. In *Proceedings of The 30th International Conference on Evaluation and Assessment in Software Engineering (EASE 2026).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Unified Modeling Language (UML) is a graphical notation that is one of the most commonly used in software engineering practices, playing a pivotal role for the visual representation of multiple aspects in the software engineering life-cycle such as requirements definition and design.

Among the many types of diagrams that it defines, use case and class diagram stand out as key topics due to their versatility in requirements engineering, where they are adopted to represent functional requirements, intended system usage, and key implementation, as well as in model-based engineering, where they are used as an artifact on which implementations depend.

UML is commonly taught in SE curricula due to its importance, although students perceive it as a somewhat secondary topic, dedicating less attention to it compared to coding or testing activities, resulting in turn in the production of diagrams with errors and limited ability to represent the intended domain, reducing the positive impact of UML models. Educators thus need to identify techniques and approaches that can increase student engagement and commitment, and apply such techniques to UML education.

Gamification refers to the use of game design elements in non-game contexts with the aim of increasing user engagement and motivation [6], and is a technique that has emerged over time as an effective means to increase student engagement and motivation in activities that are usually perceived as unappealing or uninteresting such as software design and testing, as well as requirements engineering activities [2].

Gamification has been applied to many software engineering disciplines, including UML education with mechanics such as points, badges, achievements, levels, and leaderboards, with positive results in terms of student engagement and performance [12, 13].

We previously introduced UMLegend [8], a gamified tool for UML class diagram exercises that allowed dynamic exercise solving with feedback dependent on the students' produced diagrams, and support for long-term mechanics to allow for continuous classroom use aimed at increasing participation in practical sessions.

We conducted an empirical experiment at Politecnico di Torino with 280 students to assess its impact on student performance, demonstrating that by gamification brought an increase in diagram correctness, together with positive student perception of the experience. However, the study was performed in a single session, preventing any generalization regarding longitudinal impact; furthermore, the tool presented certain limitations such as exclusively supporting class diagrams as the modeling notation, and having a limited evaluation engine for the exercises.

As a consequence, we continued developing the tool so that it could support more classes of diagrams and reworked both the game engine and the evaluation architecture to make using the tool more pleasant, with the goal of creating a multi-purpose gamified environment for software engineering education. This paper describes the revised version of the tool, for which we have made the following changes:

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
EASE 2026, 9–12 June, 2026, Glasgow, Scotland, United Kingdom
© 2026 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

- Most game mechanics have been reworked following feedback from the original experiment, resulting in a less penalizing experience with clearer feedback.
- The evaluation engine has been extended to support more variability in the space of available solutions, and the ability for teachers to create references has been enhanced.
- Use case diagrams have been added as a supported modeling notation for both the gamified mechanics, the evaluation engine, and the reference solution creator.
- The tool architecture has been refactored in a modular fashion to allow for easy insertion of other topics, including SE ones not related to modeling, with the end goal of making the tool architecture available as an open-source platform for the educational and research community.

We intend to perform a longitudinal study throughout spring 2026 by adopting the revised UMLegend into a software engineering course, to assess whether gamification can improve student performance for both class and use case modeling exercises.

The remainder of this paper is structured as follows: Section 2 presents an overview of gamification applied to software engineering disciplines, with a focus on model-based education and requirements engineering, Section 3 outlines the changes made in the revised version of UMLegend, while Section 4 describes the expected impact of the new version of the tool and the plans for the experiment we intend to perform.

2 BACKGROUND

In recent years, gamification has been adopted in multiple educational contexts to stimulate student participation and improve learning performance through the implementation of engaging game mechanics and elements. Multiple empirical studies have investigated the effectiveness of educational gamification, showing that it can have a positive effect on motivation and engagement [9], despite different contexts and implementation strategies performing better than others; gamification has also shown to be beneficial for both cognitive and behavioral engagement when the mechanics' design connects them to specific learning objectives [16].

Applications focused on software engineering education showed that gamification can lead to improvements in student engagement and learning performance for coding assignments [10, 14], team-based development[1], requirements definition [5], and testing activities [7, 17].

Despite the widespread use of gamification in software engineering education, examples of its application to software modeling or model-based engineering activities are still quite rare, with most of them focusing on UML class diagrams [4, 11, 15] through various mechanics such as competition, feedback, points, rewards, and achievements.

However, gamification focused more on use case diagrams remains a relatively unexplored field. Jurgelaitis et al. [12] discussed the design of an Information System course where multiple-choice exercises and questions on multiple UML notations (including use case diagrams) were served through the online platform Moodle, and enhanced these exercises with experience points, badges, and leaderboards. An application of their design was positively received by students [13], with around half of them managing to complete the use case portion of the course.

Other tools exist that cover multiple UML notations without an explicit focus on use cases such as Papygame [3], an extension for the Papyrus modeling platform where completing exercises awards progression points and feedback for errors, and GaMoVR [18], a virtual-environment game where the UML components can be directly rearranged to compose a valid diagram, with levels, experience points, and unlockable rewards as the main mechanics. However, the evaluation of these tools has focused mainly on class diagrams, meaning that the application of gamification to use case diagrams remains unexplored.

Furthermore, the existing solutions for gamifying multiple UML notations present static exercises where students have to select correct options among a set of available elements, and no tool exists where students can directly attempt to draw their personal solution on a modeling canvas. The absence of such a tool was our original motivation behind the development of UMLegend, and our intent with this paper is to extend it with a dedicated module for gamifying use case diagram modeling, while maintaining the original philosophy of the tool.

3 TOOL FEATURES

3.1 Game Mechanics

Following the original implementation of UMLegend, the new version keeps the game mechanics separated into two categories: long-term mechanics that support the users' growing skill in modeling languages over time and mechanics that impact how users interact in a specific exercise, reacting to the solutions produced and their content in relation to the exercise's context.

3.1.1 Long-term Mechanics. All the long-term mechanics implemented in the original version of UMLegend have been kept in this revised version, although all of them have been changed to improve the overall student experience.

For what concerns **Levels and Experience Points** still exist as a way to measure the students' natural progression and growth in modeling skill, with experience being obtained by successfully completing exercises like in the original version. The main change made is that all parameters tied to experience points can now be directly customized by teachers on a course by course basis, instead of being hard-coded values; as a consequence, teachers can define how many levels can be obtained, the experience threshold for each level, as well as the thresholds needed to obtain experience multipliers, as well as the values associated with those multipliers.

A similar change has been made for the **Avatar Customization** mechanic: the level conditions for unlocking avatar props were originally hard-coded and can now be changed directly by teachers. Furthermore, the way avatar props are displayed so that students can customize their look has been changed to show how each prop changes the avatar's style compared to the currently selected options.

Avatars still maintain their original purpose of being shown inside **Leaderboards**, which remain the way UMLegend implements competition between students. In addition to the original rankings (by level and experience points, and by correctness score per each

exercise), a third leaderboard has been added, ranking users by the number of completed exercises.

Lastly, a new long-term mechanic has been added in the form of **Completion Rewards**, that apply changes to the student's homepage after completing an exercise. The first change is that the completed exercise's corresponding block is highlighted in golden and displays the corresponding enemy's icon, giving a visual change to the exercise page and providing a satisfactory improved display; the second change consists of allowing the students to access a dedicated section of the tool where it is possible to consult the reference solutions for the exercises that have been completed.

3.1.2 Exercise-specific Mechanics. Similarly to the long-term mechanics, all the original exercise-dependent mechanics have been kept in the improved version, with most of them also being changed following feedback from the original experiment; new mechanics have been added as well, but the original concept remains the same: mechanics are activated in reaction to a student's diagram being evaluated and allow for a personalized experience that guides toward learning correct modeling practices.

Indicators remain as the guiding mechanic that helps students track how well they are performing in an exercise by knowing their current completeness score and obtainable experience points. Completeness is still computed in the same way as in the original version, that is, as the count of elements in the reference solution that are considered a valid match over the total number of reference elements. The experience reward has been changed in some ways, following feedback gathered after the original experiment: one of the main complaints was that the reduction in experience felt too strict and frustrating, and we decided to address this by making it so that only new errors cause a reduction of the obtainable experience; repeating an error does not reduce the experience points anymore, and we assume that this should mitigate the frustration found in the original version. Additionally, the experience calculation still ensures that errors never reduce the obtainable value below 35% of the total value, and correcting errors increases the experience score by the same amount it was reduced originally. Additionally, indicators have received a visual change and became progress circles, to improve their visibility (see Figure 1(3)).

The **Avatar Feedback** was used as a negative motivator in the original implementation, with the student's avatar becoming progressively sadder with the reductions in the obtainable experience points. The revised version has implemented two changes: first of all, the avatar's mood is updated after every check by comparing its results with those obtained in the preceding one, deciding the new mood depending on the difference between the experience values, the number of new errors, the number of fixed errors, and the difference in completeness values. With this change on the criteria that affect the avatar's mood, it is possible to have a change that makes it sadder, does not change its mood, or makes it happier, leading to three positive states, three negative ones, and a neutral one. The avatar's mood is stated in the upper left side of the exercise page, with Figure 1(1) showing an example of how it looks like.

To support the students' understanding of this mechanic, a new one has been implemented to allow students to easily gauge the changes between two subsequent evaluations in the form of the **Evaluation Recap**: after every check, a modal window appears showing the results and listing how many existing errors have been fixed, how many new errors have been found and, consequently, the changes in experience and correctness.

Another new mechanic that relates to the students' avatars during an exercise has been added in the form of a robotic **Boss** that visually represents the challenge associated with the exercise and taunts the students with a customizable phrase, as shown in Figure 1(2). The boss disappears with an animation when the exercise is completed, signifying that the student has managed to win the challenge, and it appears on the exercise's icon in the homepage as a badge of success. The boss' look and its corresponding dialogue can be customized by a teacher for each exercise in a dedicated section of the tool.

The **Completion Screen** is another mechanic that has been added in this improved version: when a student completes an exercise, a modal window appears showing the obtained experience points (with the eventual bonus multipliers) and a progress bar that shows the updated student experience. If the student has reached a new level, the list of unlocked avatar props is also shown in this window.

Two mechanics that were present in the original version (**Error Lists** and **Diagram Feedback**) have been changed to provide improved guidance and help toward successful exercise completion and correct modeling understanding due to a change in how the evaluation results are managed: the original version only considered syntactic and semantic errors as information to convey to students, while the revised version considers matching elements something relevant that students should know.

As a consequence, the tool now displays **Feedback Lists** for syntactic and semantic errors, as well as a third list for the diagram elements that are considered valid matches for elements in the reference solution. Furthermore, messages have been rewritten to ensure that the affected diagram element is mentioned explicitly, and the reasoning behind the error in relation to the reference solution is clear. Feedback lists appear in the upper right side of the exercise page, as shown in Figure 1(4).

Diagram Feedback has been changed as well, since now correct elements can also be colored in green. Coloring is done in a way that ensures correct elements can always be identified by prioritizing the green background, while errors have a red/orange text depending on whether they are semantic or syntactic ones. Figure 1(5) shows a diagram where this mechanic is applied and all elements are considered correct.

The revised version also keeps the same in-exercise leaderboard as in the original version, with no additional change made to said mechanic.

3.2 Use Case Diagram Evaluation

The main improvement of the revised version of UMLegend compared to the one used in the original experiment is the support for use case diagrams in its automated evaluator, extending the range of software modeling topics that it can cover and teach. The evaluation module for UCDs identifies the following three main elements for its intended reference solution structure: actors, use cases, and systems, distinguishing the latter between "owning" systems that are allowed to contain use cases, and "external" systems that cannot

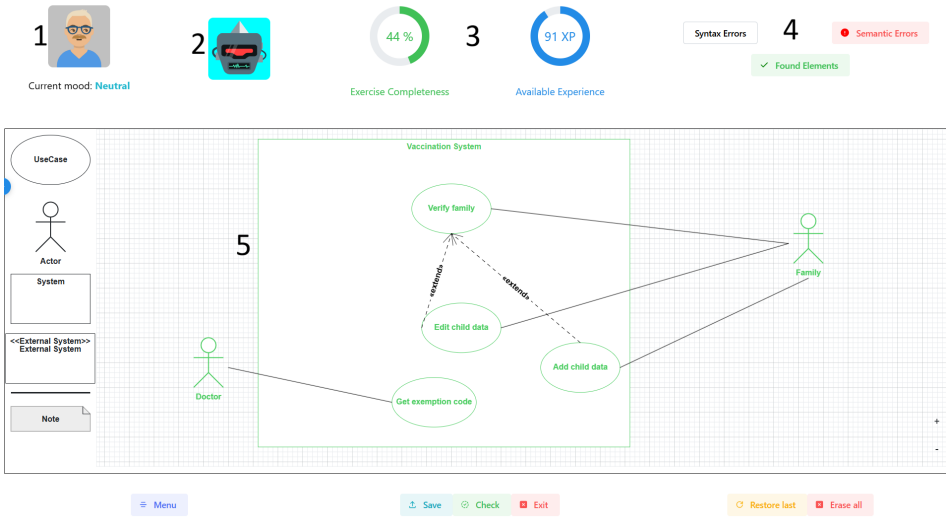

Figure 1: Exercise page showing the exercise-dependent mechanics

contain use cases and are intended to be used as supporting actors for the use cases to represent.

In addition, relationships between elements are also defined in the reference solution structure, with three different relationship types: I) associations between use cases and actors/systems, used to state which actors perform which use cases or, if the corresponding flag is set, if the actor is a supporting one for the use case; II) associations between actors, representing inheritance between two actors; III) associations between two use cases, with a flag being used to state whether the association is an *Inclusion* or an *Extension*.

The automated evaluator works by receiving the Apollon source code of a student's diagram and parsing it so that it is converted to a data structure of the same type as the reference solution, ensuring compatibility for the comparison between the two.

The evaluation procedure's first step consists of comparing each of the reference elements (actors, use cases, system) with all diagram elements of the same type until one element with a similar enough name (computed with string similarity using Levenshtein distance and requiring at least 75% similarity) is found, signifying a match between the two elements; following this, the evaluator iterates through the reference associations to check if, for each of them, both involved elements have a matching element in the diagram and, if that is true and there is an association between the pair, considers the relationship matched.

Once all matches have been found, the evaluator computes completeness metrics for each element type, as well as the overall completeness, and then performs two additional evaluation steps to identify syntactic and semantic errors. Syntax errors do not depend on the student's solution, since they focus on structural issues such as elements with missing or duplicate names, wrong or not allowed associations, misplaced elements such as actors inside systems or use cases outside systems.

Semantic violations, instead, depend on the result of the comparison and highlight errors such as missing matches for reference elements, matching use case associations with wrong type (e.g., an *Include* in place of an *Extend*) or with the wrong order of use cases, use cases belonging to the wrong system, elements with names that are not allowed, and relationships between elements that should not be included.

The reference solution structure for an exercise can be defined by teachers in two ways: drawing it by using Apollon's modeling interface, or through a form where it is possible to specify all expected elements and their details. The form allows teachers to define all elements separately and establish their relationships after creation: Figure 2 shows how the solution form looks like when editing a use case by specifying its owning system and possible alternative names for it.

An important change compared to the original version is the support for multiple solutions to the same exercise, letting teachers cover more variability in the possible representations of a problem. Consequently, the evaluation procedure described above is performed after every check for all solutions created for an exercise, and the feedback is displayed for the solution that has achieved the highest completeness score.

4 PLANNED USE AND IMPACT

The original experiment we conducted using UMLegend showed that gamifying UML modeling assignments results in increased diagram correctness and fewer errors, and that it is a well-received experience by students. However, the experiment was conducted as a single session, meaning that we have no way to generalize the findings to longitudinal use of the tool. Furthermore, the original version covered exclusively class diagrams, leaving out other modeling notations such as use case or deployment diagrams that are commonly used in requirements engineering and are a relevant topic for software engineering education.

The revised version has been developed to provide an educational environment that can support multiple modeling notations

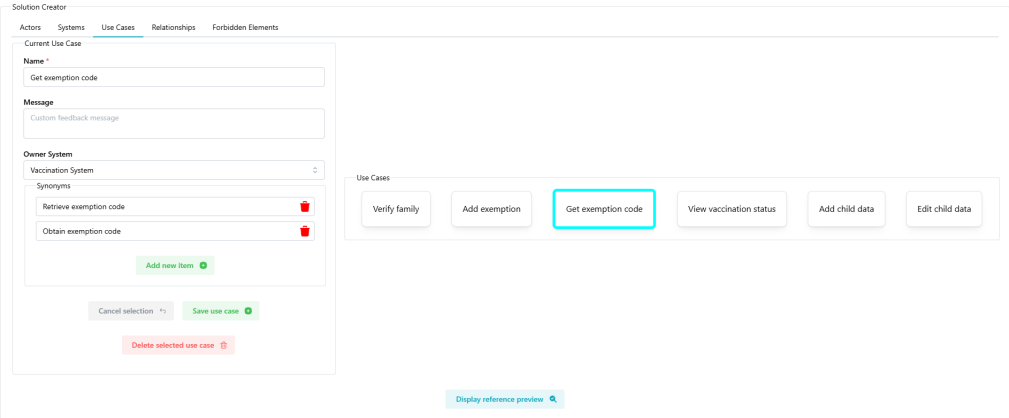

Figure 2: Form for creating a reference solution for a use case diagram, showing an editable use case

in a modular structure, meaning that additional notations can be included in the tool by defining the related evaluation logic and the corresponding solution creation form. As a consequence, we envision the possibility to define a basic gamified educational environment that can be extended to support different software engineering topics, not strictly related to modeling languages, that teachers can extend and deploy.

We intend to deploy the revised UMLegend and adopt it during the 2026 edition of the Software Engineering course at our university, where it will be made available for students to use for the duration of the course to solve multiple class diagram and use case diagram assignments. Our goal is to assess whether continuous use of a gamified modeling platform can lead to improved student performance by observing how student performance changes over time, as well as how student perform in modeling assignments during the course's final exam.

The results of this examination will serve as an example of how the long-term application of gamification impacts software engineering education, providing insights for the research community; additionally, we plan to make the tool available as an open-source project so that other researchers can apply it in their courses, and extend it so that it supports other topics.

REFERENCES

[1] Bilal Akpolat and Wolfgang Slany. 2014. Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification. 149–153. https://doi.org/10.1109/CSEET.2014.6816792

[2] Manal M. Alhammad and Ana M. Moreno. 2018. Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software* 141 (2018), 131–150. https://doi.org/10.1016/j.jss.2018.03.065

[3] Antonio Bucchiarone, Maxime Savary-Leblanc, Xavier Le Pallec, Antonio Cicchetti, Sébastien Gérard, Simone Bassanelli, Federica Gini, and Annapaola Marconi. 2023. Gamifying model-based engineering: the PapyGame experience. *Software and Systems Modeling* 22, 4 (01 Aug 2023), 1369–1389. https://doi.org/10.1007/s10270-023-01091-8

[4] Felix Cammaerts and Monique Snoeck. 2023. ModelDefenders: A novel gamified mutation testing game for model-driven engineering. In *PoEM Companion*. https://api.semanticscholar.org/CorpusID:268946614

[5] Sergio Galvan Cruz, Mirna Muñoz, Jezreel Mejía, Adriana Peña, and David Bonilla Carranza. 2025. Experimental Evaluation of Using Gamification in Software Engineering Education in Software Requirements. *CLEI Electronic Journal* 28, 2 (2025), 3–1.

[6] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From Game Design Elements to Gamefulness: Defining "Gamification". In *Proceedings of the 15th International Academic MindTrek Conference*. https://doi.org/10.1145/2181037.2181040

[7] Gordon Fraser, Alessio Gambi, Marvin Kreis, and José Miguel Rojas. 2019. Gamifying a Software Testing Course with Code Defenders. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 571–577. https://doi.org/10.1145/3287324.3287471

[8] Giacomo Garaccione, Riccardo Coppola, Luca Ardito, and Marco Torchiano. 2026. Gamification of conceptual modeling education with UML class diagrams: an experimental analysis. *Software and Systems Modeling* 25, 1 (01 Feb 2026), 239–270. https://doi.org/10.1007/s10270-025-01282-5

[9] Juho Hamari, Jonna Koivisto, and Harri Sarsa. 2014. Does Gamification Work? – A Literature Review of Empirical Studies on Gamification. In *Proceedings of the 47th Hawaii International Conference on System Sciences*. https://doi.org/10.1109/HICSS.2014.377

[10] María-Blanca Ibáñez, Ángela Di-Serio, and Carlos Delgado-Kloos. 2014. Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on Learning Technologies* 7, 3 (2014), 291–301. https://doi.org/10.1109/TLT.2014.2329293

[11] Ed Júnior and Kleinner Farias. 2021. ModelGame: A Quality Model for Gamified Software Modeling Learning. In *Proceedings of the 15th Brazilian Symposium on Software Components, Architectures, and Reuse* (Joinville, Brazil) *(SBCARS '21)*. Association for Computing Machinery, New York, NY, USA, 100–109. https://doi.org/10.1145/3483899.3483910

[12] Mantas Jurgelaitis. 2018. Using Gamification for Teaching UML in Information System Design Course. https://api.semanticscholar.org/CorpusID:54041007

[13] Mantas Jurgelaitis, Lina Čeponienė, Jonas Čeponis, and Vaidotas Drungilas. 2019. Implementing gamification in a university-level UML modeling course: A case study. *Computer Applications in Engineering Education* 27, 2 (2019), 332–343. https://doi.org/10.1002/cae.22077 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cae.22077

[14] Beatriz Marín, J. Frez, José Cruz-Lemus, and Marcela Genero. 2018. An Empirical Investigation on the Benefits of Gamification in Programming Courses. *ACM Transactions on Computing Education* 19 (11 2018), 1–22. https://doi.org/10.1145/3231709

[15] Beatriz Marín, Felipe Larenas, and Giovanni Giachetti. 2018. Learning Conceptual Modeling Design Through the Classutopia Serious Game. *International Journal of Software Engineering and Knowledge Engineering* 28 (11 2018), 1679–1699. https://doi.org/10.1142/S0218194018400235

[16] Katie Seaborn and Deborah I. Fels. 2015. Gamification in theory and action: A survey. *International Journal of Human-Computer Studies* 74 (2015), 14–31. https://doi.org/10.1016/j.ijhcs.2014.09.006

[17] Philipp Straubinger and Gordon Fraser. 2024. Improving Testing Behavior by Gamifying IntelliJ. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 49, 13 pages. https://doi.org/10.1145/3597503.3623339

[18] Enes Yigitbas, Maximilian Schmidt, Antonio Bucchiarone, Sebastian Gottschalk, and Gregor Engels. 2024. GaMoVR: Gamification-based UML learning environment in virtual reality. *Science of Computer Programming* 231 (2024), 103029. https://doi.org/10.1016/j.scico.2023.103029