UMLegend: A Gamified Learning Tool for Conceptual Modeling with UML Class Diagrams

Christian Cagnazzo
Politecnico di Torino
Turin, Italy
s304045@studenti.polito.it

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

Marco Torchiano
Politecnico di Torino
Turin, Italy
marco.torchiano@polito.it

ABSTRACT

Conceptual modeling is a fundamental skill for analysts and software engineers, as it is necessary for abstracting concepts and expressing them in a meaningful way that can then be translated into effective software design. Conceptual modeling is taught in different Software Engineering university curricula, with Unified Modeling Language (UML) class diagram being one of the most commonly used notations for this purpose. This paper presents a proposal for the Gamification of conceptual modeling education in a university environment. We describe a tool prototype with common gamified mechanics such as experience points, levels, and customizable avatars, together with an underlying evaluation system for assessing the correctness of the diagrams modeled by the students. A preliminary assessment on existing lab assignments was performed to gauge the ability of detecting errors. We discuss the tool capability and the potential benefits that such a tool could bring, as well as envision future plans for an empirical evaluation of those benefits.

CCS CONCEPTS

• **Applied computing → Computer-assisted instruction**; **Interactive learning environments**.

KEYWORDS

Education, Conceptual Modeling, Gamification, UML Class Diagrams, Software Engineering

ACM Reference Format:
Christian Cagnazzo, Giacomo Garaccione, Riccardo Coppola, Luca Ardito, and Marco Torchiano. 2023. UMLegend: A Gamified Learning Tool for Conceptual Modeling with UML Class Diagrams. In *Proceedings of the 2nd International Workshop on Gamification in Software Development, Verification, and Validation (Gamify '23), December 4, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3617553.3617883

Gamify '23, December 4, 2023, San Francisco, CA, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0373-7/23/12.
https://doi.org/10.1145/3617553.3617883

1 INTRODUCTION

Conceptual modeling is an essential skill for analysts as well as for software engineers, as it is necessary for abstracting concepts from the problem domain and expressing them in an effective way that can be later translated into software design.

Conceptual modeling is a necessary step in the design phase of a new software system since it offers a way to clearly define the different views and requirements of the system, detailing all the most important concepts involved.

One of the most commonly used specifications for conceptual modeling is the Unified Modeling Language (UML) Class Diagram [7]: a graphical notation that uses classes to express the various concepts that make up a system; those concepts may be physical entities, roles, events, geographical entities, time records, and generally all the necessary concepts needed to describe the various elements involved in a system.

UML class diagrams are commonly taught in software engineering courses as an introductory topic typically used for the requirements definition phase. Examples of common errors [1] performed by students when learning conceptual modeling include multiplicity errors, missing classes, attributes, or associations, using wrong names for classes, and including concepts that should not be present such as events in a diagram.

Teacher assistance during classroom exercises can help students understand the reasoning behind their errors and guide them toward correct modeling practices, but providing precise feedback becomes unfeasible when the number of students becomes too large for teachers to manage.

Gamification, the practice of using elements, strategies, and mechanics that are typically found in games in non-recreational contexts [4], has been widely adopted in software engineering over recent years. The benefits of using gamification include increased interest in unappealing topics, higher participation in activities, and increases in motivation thanks to competition and collaboration mechanisms.

We posit that using gamification could facilitate the learning process of conceptual modeling by increasing students' interest in the topic through game mechanics that would make the activity more appealing; moreover, using a dedicated learning tool with detailed feedback mechanisms inserted in a game environment could be more effective in reaching students in comparison to human feedback given by teachers.

The remainder of the paper is structured in the following way: Section 2 introduces relevant background information and related work, and Section 3 describes the tool. In Section 4 we present the theorized impact of our tool, while Section 5 details the conclusions we draw as well as the plans for future usage of the tool.

2 BACKGROUND AND RELATED WORK

Gamification has become more commonly used in recent years in educational contexts, as an effective strategy to increase students' motivation and participation [5, 6].

Elements commonly seen in gamified systems depend on concepts such as rewarding players for their actions, increasing motivation with unknown and unexpected events, competition mechanisms with other participants, and in general, features that enrich the experience by making it more fun.

One of the most commonly used frameworks for assessing how well gamification is implemented in a gamified system is called Octalysis [3]: the framework defines eight Core Drives that represent different facets of human behavior stimulated by gamification such as accomplishment, ownership, scarcity, social influence, avoidance, empowerment, unpredictability, and giving meaning to one's actions.

The eight Core Drives are divided into drives that motivate users by making them feel powerful, in control of their actions, and satisfied with themselves (White Hat Drives), and drives that exploit negative feelings by making users feel anxious, addicted, obsessed, and generally worried in order to motivate them in taking part in a gamified system (Black Hat Drives).

A second division of the core drives is in right-brained ones, related to creativity, social interactions, and self-expression, and left-brained ones, associated with logic, ownership, and calculations.

The horizontal division between left-brained and right-brained drives comes together with the vertical division between White Hat and Black Hat drives in the form of an octagon, which can be used to assess the distribution of the game elements in a system by increasing the size of each side of the figure depending on how many elements of the corresponding drive are present in the system. A balanced octalysis graph is interpreted as an evidence of a balanced gamified system.

In the field of software engineering, examples of gamified tools have recently been applied to the various activities that are part of the discipline, such as mutation testing [6], refactoring [5], and requirements definition [10]. The following works, instead, focus on the gamification of conceptual modeling languages.

We identify three main examples in literature that applied gamification to software modeling. Papygame [2] is a plugin developed for Papyrus [1], a modeling tool that supports UML class diagrams and other modeling languages derived from UML. Students can solve assignments composed of different exercises, each associated with a level; these exercises can either be played with a *Hangman* game, where a new piece of the hanged man is drawn for every mistake made, or without any kind of game element. Completing an exercise without drawing the full hangman picture unlocks new exercises and rewards, with failure (with no new unlocked exercises as a consequence) being represented by the full picture being obtained. A preliminary evaluation conducted with students focusing on the plugin's usability and user experience yielded promising results, and the authors mention room for improvement and plans for future tool usage.

Jurgelaitis et al. [8] describe the gamification of a UML university course with a level-based structure where new topics can only be obtained by increasing one's level. In the course structure, class diagrams are considered a necessary prerequisite, as they consist of the topic of the first level of the entire course. The course used elements commonly seen in gamification theory such as Rewards (in the form of experience points, badges, and currencies that can be spent for purchasing course resources), Tasks connected to the various exercises, and Leaderboards where students are ranked based on their experience points. A comparison between the average student grade for the gamified edition of the course and a non-gamified edition showed that gamification led to an increase in the average assessment score.

Lastly, LearnER [9] implements a gamified editor for UML class diagrams as well as Entity-Relationship diagrams using common elements such as points awarded for correctly solved exercises, leaderboards, both regarding general standing and per-exercise, progress indicators, and hints toward the correct solution expected for the model. The authors mention that the tool has been used continuously since 2017, with gamification and feedback being identified as effective and beneficial for the learning process.

3 TOOL IMPLEMENTATION

We describe in this section the main concepts that compose our tool and the gamified mechanics that we have adopted. The tool has been developed as a React-based web application that implements the mechanics inside a UML modeling canvas.

3.1 Diagram Evaluation

Our tool was built in order to be used in a classroom environment to assist students in learning correct modeling practices, so it is necessary for the tool to offer a way for students to assess the correctness of the diagrams they can produce using the tool. As a side effect, automated assessments can also speed up the evaluation of the diagrams performed by teachers, allowing an increase in the number of evaluated exercises provided during a software modeling course.

In order to do so, we implemented a correctness check inside the tool that analyzes the diagram produced by students and returns a list of all violations present in the diagram. Following commonly used specifications regarding conceptual modeling [1], we divided the possible violations into three different categories:

- *Syntax Errors.* Violations of the syntax rules defined for UML class diagrams. These errors may include: an attribute not having a type or a type that is not allowed, an association missing its multiplicities or not having a name, a class having no name, no attributes, or no associations to other classes.
- *Semantic Errors.* Violations that are specific to the exercise for which the student is modeling a class diagram. These errors may include: a specific concept not being modeled with a class, an association between classes not being present

[1]https://www.eclipse.org/papyrus/, last accessed on: 10/07/2023

or being present but with incorrect multiplicities, a class missing important attributes, or the existence of a concept that is not allowed to be modeled with a class.
- *Pragmatic Quality Warnings.* Violations that are not considered errors, but still represent something that can be modeled in a more correct way. These warnings can be given for modeling a concept that can be a single attribute as a class, having unnecessary associations between classes, associations with the wrong name, or having unnecessary classes.

Students are able to have their diagram evaluated whenever they wish, and the entire diagram is checked for every possible violation, with syntax errors being checked first.

In case the syntax evaluation finds missing relevant information (e.g. a class without attributes, an association without a name, an attribute without a type) the semantic check is performed with placeholder values, with the affected elements not being considered in the subsequent evaluation.

3.2 Gamified Mechanics

We selected the following mechanics for implementation in the tool:

(1) *Levels.* A mechanic that is commonly seen in gamified environments, levels represent a student's modeling skill in a numeric and direct way. Students start at level 1 and are able to earn experience points by completing exercises. Exercises have an increasing difficulty level and award higher experience when completed.
Moreover, there is no fixed order for completing exercises, meaning that students are free to attempt solving higher-level exercises first for higher rewards (experience is multiplied if the exercise is at a higher level than the student's current level), at the cost of facing a harder challenge.
The experience reward for each exercise is also tied to the checks performed during the exercise: modeling a diagram that contains syntax or semantic errors will lead to an experience reduction; similarly, successfully completing an exercise with a higher number of checks will slightly reduce the experience obtained.
Following the Core Drives defined in the Octalysis framework described above, this mechanic fits in the *Development and Accomplishment* drive: experience points and levels are a classical example of indicators of the progress a user has made in a gamified system, and new levels (with the corresponding rewards) give a sense of fulfillment. Additionally, the presence of an experience reduction in case of repeated failures ties to the *Loss and Avoidance* drive.

(2) *Avatars.* Avatars offer a way for students to customize their experience and express their own individuality inside the tool. We implemented avatars in our tool with a web-based implementation of the Avataaars Sketch library [2]: the library allows for the customization of a human avatar by changing different components such as its clothes, its hairstyle and hair color, presence of facial hair, and accessories.
In our tool, students begin with a few unlocked props for their avatars and are able to unlock new ones by completing exercises and leveling up: a dedicated tool section offers the chance to customize the avatar by changing every single prop in detail; the section also shows the props that are yet to be unlocked, together with the needed level. An example of the avatar customization page is shown in Figure 1.
Using customizable avatars refers to Octalysis's concept of *Ownership*: students are more motivated in using the tool if they feel a close connection with something personal that lets them express their individuality

[2]http://avataaars.com, last accessed on: 07/07/2023

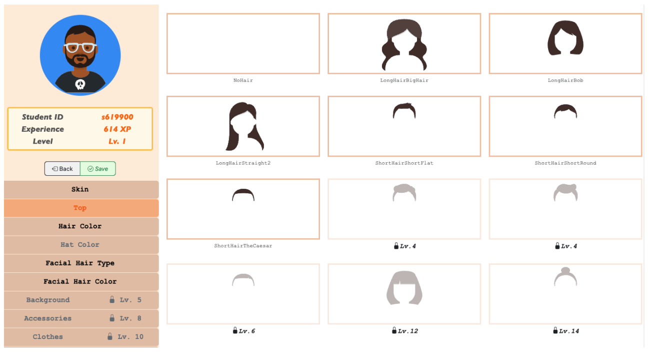
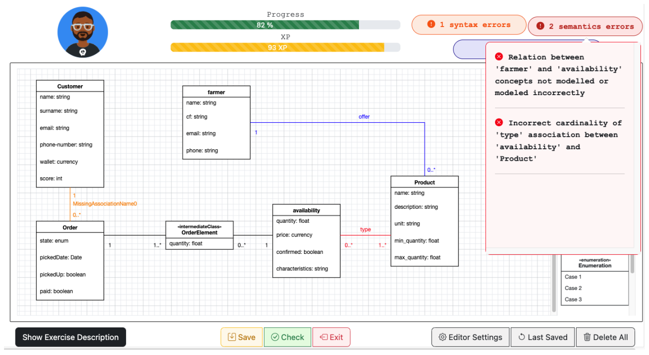


Figure 1: Tool section dedicated to the avatar's customization



Relation between 'farmer' and 'availability' concepts not modelled or modeled incorrectly

Incorrect cardinality of 'type' association between 'availability' and 'Product'

Figure 2: Exercise page with colored feedback on the diagram

(3) *Feedback.* As a way to enhance the learning activity performed by the students, feedback is given after every correctness check made by the students. All the violations found in a diagram lead to the diagram part that contains an error to be colored differently, directly highlighting where students need to turn their focus to improve the model. The coloring used also reflects the type of error: syntax errors are orange-colored, elements that cause a semantic error are red, while pragmatic quality warnings are identified by a blue color. Additionally, a list of all the violations found is available for consultation after a check, so that students are able to directly assess and correct the errors. Figure 2 presents an example of a diagram that contains violations for every type of error as well as a list detailing the semantic errors found.
An additional feedback mechanism, directly tied to the student's *Avatar*, consists of a visual change of the avatar itself after too many errors have been performed: the student's

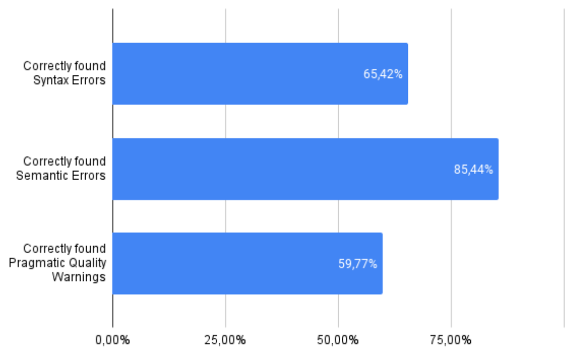


Figure 3: Percentages of syntax errors, semantic errors, and pragmatic quality warnings correctly identified by the tool

avatar will change its facial expression, changing from a happy emotion to a progressively sadder one the more mistakes the student makes. This kind of negative feedback is used to motivate the student in thinking carefully and avoiding making mistakes.

The feedback in the form of error messages and coloring parts of the diagram can be connected to Octalysis's *Empowerment of Creativity and Feedback* Core Drive: students are able to directly see which parts of their diagrams are incorrect and the reason behind the errors and are in turn motivated to perform new actions (in this case changing the model and checking again) to overcome the error.

Additionally, the visual effects on the student's avatar can be thought of as an example of *Loss and Aversion* drive: students who see the change in the avatar's status will be more motivated in future modeling tasks, in order to spare the avatar further "suffering".

4 POTENTIAL IMPACT

We performed a preliminary evaluation by inserting solutions from a laboratory exercise performed by students of the previous edition of the course and had the tool evaluate those solutions based on an example of a correct diagram.

A total of 30 solutions have been analyzed by the tool, which resulted in a list of errors and warnings for each solution; each list has then been reviewed by a human evaluator in order to assess how many errors and warnings the tool was able to identify correctly. The distribution of errors and warnings identified correctly is presented in Figure 3.

As can be seen in the Figure, the tool is able to identify errors in a way that can be comparable to a human evaluator, although there is a margin for improvement that has emerged during the assessment process.

We expect our tool to assist students in learning correct modeling practices by providing detailed feedback directly on the diagrams produced by the students themselves: the tool is planned to be used in a laboratory environment where students are encouraged to solve modeling exercises on their own, with the presence of a teaching assistant to answer their doubts and provide support.

Additionally, the presence of in-game rewards for solving exercises in a correct way is something that we assume will act as an effective motivator for students: previous editions of the course offered no incentive for students to attempt solving the exercise other than learning conceptual modeling, and we expect that the tool will change this trend.

5 CONCLUSION AND FUTURE WORK

In this paper, we described a prototypical tool for the gamification of learning conceptual modeling using UML class diagrams. The web tool, which also includes an evaluation engine that assesses the correctness of the produced diagrams, is intended to be used by students to facilitate their learning of correct modeling practices.

Our future plans for the tool include its usage in an academic environment with a longitudinal experiment, where we plan to track students' progress during the course.

We also plan to expand the tool's features with gamified mechanics that are suited for long-term usage: competition mechanisms such as leaderboards are planned, together with the creation of a quest-line mechanic based around the exercises, in order to assess whether continuous usage of a gamified tool can improve students' modeling practices.

REFERENCES

[1] Narasimha Bolloju and Felix SK Leung. 2006. Assisting novice analysts in developing quality conceptual models with UML. *Commun. ACM* 49, 7 (2006), 108–112.

[2] A. Bucchiarone, M. Savary-Leblanc, X. Le Pallec, J. M. Bruel, A. Cicchetti, J. Cabot, and S. Gérard. 2023. Gamifying model-based engineering: The PapyGame tool. *Science of Computer Programming* 230 (2023), 102974. https://doi.org/10.1016/j.scico.2023.102974

[3] Y.k. Chou. 2015. *Actionable Gamification: Beyond Points, Badges, and Leaderboards.* Createspace Independent Publishing Platform. https://books.google.it/books?id=jFWQrgEACAAJ

[4] S. Deterding, D. Dixon, R. Khaled, and L. Nacke. 2011. From game design elements to gamefulness: defining" gamification". In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments.* 9–15.

[5] H. M. dos Santos, V. H. S. Durelli, M. Souza, E. Figueiredo, L. T. da Silva, and R. S. Durelli. 2019. CleanGame: Gamifying the Identification of Code Smells. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering* (Salvador, Brazil) *(SBES 2019).* Association for Computing Machinery, New York, NY, USA, 437–446. https://doi.org/10.1145/3350768.3352490

[6] G. Fraser, A. Gambi, M. Kreis, and J. 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

[7] Object Management Group. 2017. OMG, Unified Modeling Language (UML) 2.5.1 Superstructure Specification.

[8] M. Jurgelaitis, L. Čeponienė, J. Čeponis, and V. 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

[9] Olav O. Dæhli, B. Kristoffersen, P. Lauvås Jr, and T. Sandnes. 2021. Exploring Feedback and Gamification in a Data Modeling Learning Tool. *Electronic Journal of e-Learning* 19, 6 (2021), 559–574.

[10] W. Prasetya, C. Leek, O. Melkonian, J. ten Tusscher, J. van Bergen, J. Everinkr, T. van der Klis, R. Meijerink, R. Oosenbrug, J. Oostveen, et al. 2019. Having fun in learning formal specifications. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET).* IEEE, 192–196.

Received 2023-08-03; accepted 2023-08-17