PolyGloT-UML: A Gamified Framework for Enhancing UML Learning Paths

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

Tommaso Guidolin
tommaso.guidolin@studenti.unitn.it
Universita' degli Studi di Trento, Italy
Trento, Italy

Lorenzo Fasol
lorenzo.fasol@studenti.unitn.it
Universita' degli Studi di Trento, Italy
Trento, Italy

Gianluca Schiavo
gschiavo@fbk.eu
Fondazione Bruno Kessler
Trento, Italy

Jörg Kienzle
Joerg.Kienzle@{uma.es,mcgill.ca}
ITIS Software, University of Málaga / McGill University
Málaga / Montreal, Spain / Canada

Sébastien Gérard
Sebastien.Gerard@cea.fr
Université Paris-Saclay, CEA, List
91120, Palaiseau, France

David Négrier
d.negrier@workadventu.re
WorkAdventure
Paris, France

Tommaso Martorella
tommaso.martorella@epfl.ch
EPFL
Lausanne, Switzerland

ABSTRACT

Modeling is a key activity in conceptual design and system design, making learning and understanding modeling languages like the Unified Modeling Language (UML) crucial. Despite the availability of various interactive and gamified UML learning applications, many lack the flexibility to integrate diverse tools and create personalized learning paths. PolyGloT-UML addresses these limitations by providing a highly adaptable framework that seamlessly integrates with a wide array of tools used within the Model Driven Engineering community. This framework facilitates the creation of dynamic and personalized learning experiences, enhancing learner engagement and improving learning outcomes. This demo paper presents PolyGloT-UML, showcasing its capability to unify different educational tools into a cohesive and effective learning environment for UML and other modeling concepts.

CCS CONCEPTS

• **Software and its engineering** → *Software design engineering*; • **Applied computing** → **Interactive learning environments**.

KEYWORDS

Model Driven Engineering Education, UML, Personalized Learning, Gamification

ACM Reference Format:
Antonio Bucchiarone, Tommaso Guidolin, Lorenzo Fasol, Gianluca Schiavo, Jörg Kienzle, Sébastien Gérard, David Négrier, and Tommaso Martorella. 2024. PolyGloT-UML: A Gamified Framework for Enhancing UML Learning Paths. In *ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24), September 22–27, 2024, Linz, Austria.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3652620.3687785

1 INTRODUCTION

Model Driven Engineering (MDE) has become an essential approach in both academic and industrial settings for conceptual design and system development. The unified modeling language (UML) stands out as one of the most widely adopted modeling languages in this domain. However, the complexity of UML can present significant learning challenges. To address these challenges, various interactive and gamified UML learning applications have been developed. Despite their benefits, many of these applications lack flexibility in integrating diverse tools and creating personalized learning paths, which are crucial for effective and engaging learning experiences [7].

PolyGloT [3] addresses these limitations by offering a highly flexible framework that seamlessly integrates with a wide array of tools used within the MDE community. This flexibility is particularly valuable for creating personalized learning paths tailored to individual learners' needs and preferences with the support of GenAI functionalities. By leveraging the adaptability of PolyGloT, educators can incorporate various modeling tools, interactive environments, and assessment mechanisms into a cohesive learning experience.

In this demo paper, we present PolyGloT-UML, a specialized version of Polyglot designed to enhance UML learning paths through gamification and interactive content. PolyGloT-UML enables educators to craft dynamic learning experiences that can potentially include learning tasks, facilitated through a variety of tools such as PapyrusWeb (Section 3.5) for modeling activities. This integration ensures that learners can engage with the content in a manner that best suits their learning needs and progresses seamlessly through their personalized learning paths.

The PolyGloT-UML framework not only has the potential to enhance motivation and engagement but also ensures the comprehensive realization of gamification benefits. By combining a diverse array of educational tools within a cohesive learning environment, PolyGloT-UML provides a support for instructing UML and other modeling concepts. This integrated approach not only streamlines the learning process but also enhances its efficacy and enjoyment.

2 RELATED WORK

In the following, we briefly present prior work on the topics of gamified learning environments for UML modeling and of learning paths for personalized education.

2.1 Gamified Learning Environments for UML

Gamification is acknowledged as a promising approach in education, including engineering education, as evidenced by its applications documented in the literature within the fields of MDE and UML modeling [4, 5, 7]. For instance, PapyGame [4] is an application that gamifies learning UML modeling extending the modeling tool Papyrus[1]. PapyGame integrates a game-like interface with the Papyrus 2D modeling environment, enabling users to earn XP as they advance through UML challenges and monitor their progress though a leaderboard.

LearnER [5] is another gamified modeling tool, primarily designed for training students in data modeling incorporating game elements like a point system and leaderboards and personalized feedback on the students' learning progress.

Furthermore, [7] presents a gamified course design for modeling lectures, modifying the e-learning platform MOODLE[2] to incorporate game elements. However, this application does not involve active modeling tasks; instead, it covers various UML diagram types with lecture content and gamified knowledge tests.

Another gamified learning application for UML modeling is discussed in [15] which focuses on a model-driven development framework for creating modeling games. This work includes example games featuring a 2D modeling environment and a scoring system for progressing through levels with diverse modeling tasks.

In summary, these approaches provide gamification-based UML learning environments to motivate learners in their tasks. However, they are unable to manage learning paths that mix various types of learning tasks (single or collaborative) within the same gamified environment, without requiring students to install different tools, which can make the learning process more complex and frustrating.

2.2 Learning Paths for Personalized Education

Learning is the cognitive process of acquiring new knowledge or skills [16] and it can be supported by providing educational pathways for guiding the learners. Learning paths are defined as *sequences of activities with designated goals to help students build up their knowledge or skills in a subject area* [14] that can be represented as *sequences of nodes accessed by a learner, including resource elements, learner data, and access records* [16]. These paths systematically arrange learning tasks, incorporating `knowledge elements` (KEs) [14], content units designed to achieve specifics learning objectives. Any content sequence meeting users' prerequisites while guiding them to their goals can be a learning path [16]. This structuring can enhance engagement and comprehension, facilitating the acquisition of knowledge, skills, and competencies. Recent initiatives focus on adaptive learning pathways to guide students efficiently [14], using KEs as foundational units. However, there is no consensus on learning path composition, with variations such as units of learning (UOL) [8], or learning nodes [16].

The challenge in constructing learning paths lies in the diversity of KEs and individual student needs, requiring tailored and varied educational methods. Gamified learning environments can aid in content customization, providing context-appropriate feedback, and adjusting learning paths based on students' needs and preferences [11].

3 CONCEPTUAL SOLUTION

The initial step in developing a system for defining and executing gamified learning paths is designing its architecture (detailed in Section 3.1). This architecture includes various components essential for creating and implementing gamified learning paths for UML Modeling. The components include an editor (3.2) that educators will use to define learning paths for their students. An execution engine is then described for managing learners' progress (3.3). The learning path can then be accessed through a 2D environment (3.4). A web tool is included for allowing UML modeling activities (3.5). To evaluate UML modeling tasks, the architecture includes a Grader (3.6) and a WebApp that supports additional learning activities beyond modeling tasks (3.7). The components provide educators a tool to define learning paths and a backend to store them, then the system manages the execution of activities, relying on multiples platforms for the validation. The learners are able to access through a 2D environment the learning paths and the required platform. The whole system is based on the synthesis of learning paths (i.e. flows), they can include multiples activities represented as nodes. Each node type recreates an activity we want to provide, the customized data guarantees the customization through the abstraction process of the activity. Gamification is core to enhancing student engagement, providing an immersive framework where learners can interact with one another and be rewarded for their achievements. This approach is fundamental to PolyGloT.

3.1 Architecture

The goal of PolyGloT-UML is to provide an open, flexible framework that accommodates various types of content and can be expanded as needed for creating experiences in UML Modeling (refer to Figure 1 for its architectural layout).

This tool is a specialization of PolyGloT [3] specifically designed for the learning of UML Modeling. By focusing on UML, it addresses the unique challenges of teaching and mastering UML concepts. This specialized version retains the original framework's flexibility and extensibility, enabling educators to create diverse and engaging learning experiences.

[1]https://www.eclipse.org/papyrus/
[2]https://moodle.com/

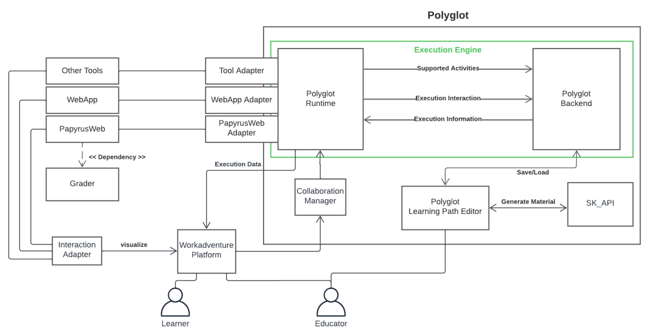

Figure 1: Overview of the high-level components comprising the implementation.

Starting from the flexibility of PolyGloT, we derived a specialized version for UML learning. This UML-focused version allows students to transition seamlessly between different activities, such as using specific tools for lessons and quizzes (see Section 3.7), switching to IDEs like PapyrusWeb (described in Section 3.5) for modeling tasks without disruptions.

We leverage the flexibility of the PolyGloT's `Execution Engine`. Students' interactions with external tools occur through `Adapters` that connect actions on the student frontend to commands and events, enabling outputs with custom formatters.

The *adaptive* learning paths indicates responsiveness to students' needs, adjusting the execution to improve the experience, by suggesting activities based on their past responses or preferences. This is facilitated by the collaboration between the `PolyGloT Runtime` and the `PolyGloT Backend` components of the `Execution Engine`. The Runtime handles submissions and validates responses, while the Backend uses validation results to propose subsequent activities in the learning path.

The architecture (Figure 1) is divided into two areas: PolyGloT (with its `Execution Engine`) and a set of external educational tools. This division highlights the architecture's adaptability, allowing efficient extension or substitution of components.

3.2 PolyGloT Learning Path Editor

PolyGloT-UML includes a `Learning Path Editor` focused on the educators' experience. Its graph-like, visual-editing capabilities and integration with the rest of the framework allow the creation of ready-to-use learning paths with ease, thanks to the abstractions provided. The `PolyGloT Learning Path Editor` serves as the primary interface with PolyGloT. This front-end application offers educators two main pages: the *discovery* page and the *editor* page. On the *discovery* page, educators can explore existing learning paths, select one, or create a new one. The *editor* page allows educators to customize the selected learning path. These two pages interact with the `PolyGloT Backend`, which loads and updates the flow's data chosen by the user through its APIs. Additionally, during the learning path creation process, educators are supported by AI features that can be used to generate material or entire exercises using AI-based tools provided directly in the editor.

3.3 PolyGloT Runtime

The `PolyGloT Runtime` is the engine of the execution of a Learning Path, it produces the information required by each tool. It is responsible for handling user interactions and updating learners' progress. The execution's lifecycle (see figure 2) is designed to be accessible from any external tool at any point and to simplify the user's experience; with this goal, this component is created as an adapter for many different platforms included in the architecture. The figure highlights the interaction of between the user and the execution, each execution starts from the choice of the learning path and the following creation of the context object (i.e., a unique session that defines the student's execution of a specific learning path) by the engine with a unique ctxId. This `id` is used through the execution to retrieve the state and update the advancement of the learning path so that each platform is synchronized.

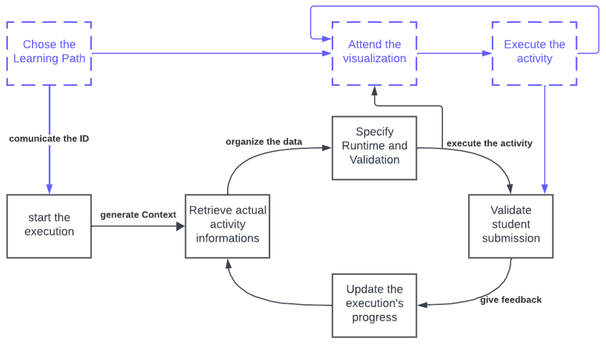

Figure 2: PolyGloT-UML Runtime Lifecycle.

When an educational tool initiates a learning activity, a concrete activity is selected for the students, and the connections (i.e., edges) to the next activities are retrieved from the `PolyGloT-UML Backend`. Both the activity and the edges undergo a transformation phase where the activity is converted into a format suitable for the frontend. Then the user is able to visualize and execute the activity s and, once completed, the active frontend adapter run the registered rules against the student submission. The adapter then sends the result of this validation phase back to the backend which will use this additional knowledge to proceed with the execution, returning the next activity. This lifecycle is extremely adaptable as, at any moment, any platform can request the activity's information independently, moreover we can see that the users' flow is smooth and continuous.

3.4 WorkAdventure

WorkAdventure[3] is a platform to build virtual 2D environments where multiple users can meet and interact. One of PolyGloT-UML core value is the deep integration with gamification mechanisms and, in this perspective, WorkAdventure has a central role in providing interactive and learner-empowering learning experiences. PolyGloT-UML can provide a workspace that acts as a central hub

[3]https://workadventu.re/

for learning paths and activities to give learners freedom in a gameful and autonomous way. Taking advantage of WorkAdventure's collaboration features, the workspace can be extended to act as a social network for learners: the access to this environment enables students to interact with peers and educators. The virtual environment is divided into different areas representing different educational platforms that can be directly accessed from within WorkAdventure.

3.5 PapyrusWeb

PapyrusWeb is the web implementation of Papyrus[4], the Eclipse UML, and SysML modeler of the Eclipse foundation developer. The web version of Papyrus is based on the open-source technology of the OBEO company, SiriusWeb[5]. The web edition of Papyrus facilitates easier authoring and collaboration on software projects and sharing UML models.

3.6 Grader

The framework includes also an automated grading approach [1, 2] for evaluating class diagram models. A class diagram model M contains a set of classes $c \in C$ with attributes $a_c \in A$ and operations $o_c \in O$, a set of enumeration type declarations $e \in E$ with enumeration literals $l_e \in L$, as well as relationships between the classes $r_{ci,cj} \in R$. The instructor assigns to each important model element m in the solution class diagram T a numerical point value.

The algorithm is a function that takes as input a instructor solution model T with associated grades, and a student model S. It outputs a list of matched classes, attributes, operations, and relationships in the student model with associated grades and feedback, as well as a list of classes, attributes, operations, and relationships not found in the student model.

A matching algorithm determines the mapping from the model elements in T to the model elements in S that maximizes the numerical grade for the student. The main steps of the algorithm are the following:

In step 1, the algorithm first determines the optimal mapping for the classes, i.e., $c_i \in T \rightarrow c_j \in S$. To find possible matches, the algorithm compares each pair of classes:

- *syntactically*, by comparing the class names, taking into account spelling mistakes by calculating the Levenshtein distance [9],
- *semantically*, by using similarity metrics provided by WordNet Similarity for Java (WS4J), including HSO [6], WUP [13] and LIN [10].
- *structurally*, by counting in how many properties (i.e., attributes, operations) they differ,
- $relationship - wise$, by counting in how many relationships (i.e., associations and inheritance) they differ.

Any class that is matched in this step scores the student the associated number of points. Classes $c_i T$ that are not matched in this step are added to the list of missed classes.

In step 2, attributes and operations are matched. It starts by matching the attributes and operations that are located in classes that have been matched in step 1, again considering *syntactic*, *semantic* and *structural* (i.e., type or signature) information. For each matched element the student scores the associated number of points. Subsequently, attributes and operations from T that have not been matched are compared with attributes and operations that have not been matched in S. If a match can be found, this means that the attribute or operation is not in the class where it should be. To determine the severity of the mistake, the algorithm considers inheritance relationship information and then assigns partial points to the student accordingly. Any properties that are not matched in this step are added to the list of missed attributes and missed operations.

In step 3 relationships are matched. Similar to step 2, the algorithm first starts by looking for matches of relationships between classes that have been matched. The association from S that contains association class preferentially matches with the association that also contains association class from T. For each found match the student is awarded the corresponding points. The algorithm then determines whether any of the still unmatched relationships in the student model could be derived from the instructor model and if yes, partial points are given. Any relationships that are not matched in this step are added to the list of missed relationships.

Finally, step 4 goes through the enumeration types defined in T and looks for *syntactic*, *semantic* or *structural* matches in the set of enumeration types found in S. Matched enumerations score points to the student, unmatched ones are added to the list of unmatched enumeration types. For every matched type, matches for the contained enumeration literals are determined as well.

Initially, this grader was integrated into the TouchCORE tool [12]. For this work, we extracted the grader component into a standalone command-line tool.

3.7 WebApp and WebApp Adapter

The WebApp provides an all-in-one platform for the students to perform many different learning activities such as reading material, multiple-choice questions, and others, by accessing external web pages.

4 IMPLEMENTATION

This section includes details on the core components of PolyGloT-UML's implementation: the *Learning Path Editor*[6] and the *Runtime*. The Learning Path Editor is implemented using Next.js and React with TypeScript to provide a robust foundation for developing a performing and secure web application. Chakra-UI is used to create the user interface, and Reactflow is employed for building interactive node-based graphs and diagrams.

The goal of the Editor is to offer a tool for visualizing and crafting Learning Paths, which is defined as a diagram flow consisting of nodes representing activities and edges, the logic connections. This "flow" is the core model of the Editor, abstracted with multiple context information and the two arrays of nodes and edges that encompass all the specifics of the learner's journey.

From PolyGloT-UML's perspective, defining an edge involves specifying how and under what conditions an advancement occurs.

[4]https://www.eclipse.org/papyrus/
[5]https://eclipse.dev/sirius/sirius-web.html
[6]https://staging.polyglot-edu.com/flows

This type is known as `PolyglotEdge`. We implemented multiple validations methods, the range of choices of edge type depends on the activity's logic, so only compatible edge types are suggested. Currently we supports various edges types like `unconditionalEdge` (used when no validation is needed), `passFailEdge` (used for true/-false validation, with "pass" and "fail" paths) and `customEdge`(primarily for coding activities, allowing custom validation functions in C#).

Similar to the `PolyglotEdge`, the `PolyglotNode` model is simplified, besides the standard base fields, it has `data` and `platform` field essential for each activity's execution.

When defining each activity, we first consider how it can be executed to determine the platform to include in that type. Once the tool is selected, we proceed to the specifics of the `data` field. This dynamic field is customized for each type and serves as an abstraction of the necessary components to recreate the tailored exercise. For example a multiple-choice activity includes a question (string), choices (string array), and isChoiceCorrect (boolean array). Additionally, each activity is associated with a Bloom category to assist educators in finding the needed node.

4.1 Demo Video and PolyGloT-UML code

A video tutorial is available here[7]. The video is divided into two parts: the first part demonstrates how educators can use the editor to create a learning path and utilize the AI creation tool for multiple-choice exercises. The second part focuses on the learner's experience, showing how students can access and navigate the 2D virtual environment in WorkAdventure to complete a learning path.

You can access the editor's discovery page here[8] to create a new learning path. The WorkAdventure map, where you can execute the learning path, is available here[9]. The source code for the framework can be found on GitHub here[10].

5 SUMMARY AND FUTURE WORK

PolyGloT-UML is a learning tool designed for the Model-Driven Engineering (MDE) community, enabling teachers to create customized learning paths for students. Its open platform allows integration with other MDE tools and the development of student-centered materials. The framework supports traditional assessments and integrates tools like PapyrusWeb and a UML Class Diagram grader, contributed by the community, to enhance MDE education.

The platform is evolving to include more activity types, new educational tools, and AI-driven features for personalized learning. Future plans involve supporting additional UML diagrams, modeling languages like SysML and OCL, and enhancing gamification and student interaction through AI-driven customization and collaborative challenges.

For the execution phase, we will extend activity execution, enhancing the student experience with AI-driven customization. Learners will receive tailored activities based on their preferences and gaps. We will also upgrade the WorkAdventure platform, adding new areas and a custom zone where learners can select subjects or skills to train using AI tools. Finally, we plan to enhance gamification by enabling learners to challenge each other in specific activities, promoting collaboration and interaction.

[7]https://youtu.be/x1ZfyBzwABo
[8]https://staging.polyglot-edu.com/flows
[9]https://play.workadventu.re/@/fondazione-brunokessler/encore/encore-learning-world
[10]https://github.com/polyglot-edu

6 ACKNOWLEDGMENTS

Jörg Kienzle's contributions were partially supported by Junta de Andalucía under project QUAL21 010UMA. The contributions by Antonio Bucchiarone and Gianluca Schiavo are supported by the Erasmus+ Programme of the European Union: "Partnerships for Innovation: Alliances" (ERASMUS-EDU-2021-PI-ALL-INNO). Project ENriching Circular use of OeR for Education (ENCORE) (Reference number 101055893).

REFERENCES

[1] Weiyi Bian, Omar Alam, and Jörg Kienzle. 2019. Automated Grading of Class Diagrams. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 700–709.
[2] Weiyi Bian, Omar Alam, and Jörg Kienzle. 2020. Is Automated Grading of Models Effective? Assessing Automated Grading of Class Diagrams. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems – MODELS 2020* (Virtual Event, Canada) *(MODELS '20)*. Association for Computing Machinery, New York, NY, USA, 365–376.
[3] Antonio Bucchiarone, Tommaso Martorella, Davide Frageri, and Diego Colombo. 2024. PolyGloT: A personalized and gamified eTutoring system for learning modelling and programming skills. *Sci. Comput. Program.* 231 (2024), 103003.
[4] 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*, Esther Guerra and Ludovico Iovino (Eds.). ACM, 5:1–5:5.
[5] Olav Dæhli, Bjørn Kristoffersen, Per Lauvås jr, and Tomas Sandnes. 2021. Exploring Feedback and Gamification in a Data Modeling Learning Tool. *Electronic Journal of e-Learning* (2021).
[6] Graeme Hirst, David St-Onge, et al. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database* 305 (1998), 305–332.
[7] Mantas Jurgelaitis, L Ceponiene, and Vaidotas Drungilas. 2018. Using Gamification for Teaching UML in Information System Design Course. *CEUR-WS* 2145 (2018), 88–94.
[8] Rob Koper and Bill Olivier. 2004. Representing the learning design of units of learning. *Journal of Educational Technology & Society* 7, 3 (2004), 97–111.
[9] Vladimir I Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10 (February 1966), 707.
[10] Dekang Lin. 1998. An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 296–304. http://dl.acm.org/citation.cfm?id=645527.657297
[11] Amir Hossein Nabizadeh, José Paulo Leal, Hamed N Rafsanjani, and Rajiv Ratn Shah. 2020. Learning path personalization and recommendation methods: A survey of the state-of-the-art. *Expert Systems with Applications* 159 (2020), 113596.
[12] Maximilian Schiedermeier, Bowen Li, Ryan Languay, Greta Freitag, Qiutan Wu, Jörg Kienzle, Hyacinth Ali, Ian Gauthier, and Gunter Mussbacher. 2021. Multi-Language Support in TouchCORE. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion*. IEEE Computer Society, Los Alamitos, CA, USA, 625–629.
[13] Zhibiao Wu and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 133–138.
[14] Fan Yang, Frederick WB Li, and Rynson WH Lau. 2010. An open model for learning path construction. In *Advances in Web-Based Learning–ICWL 2010: 9th International Conference, Shanghai, China, December 8-10, 2010. Proceedings 9*. Springer, 318–328.
[15] Alfa Yohannis. 2016. Gamification of Software Modelling Learning. In *Proceedings of the Doctoral Symposium at the 19th ACM/IEEE International Conference of Model-Driven Engineering Languages and Systems 2016 (MoDELS 2016), Saint Malo, France, October 2, 2016 (CEUR Workshop Proceedings, Vol. 1735)*, Shiva Nejati and Rick Salay (Eds.). CEUR-WS.org. http://ceur-ws.org/Vol-1735/paper1.pdf
[16] Yuwen Zhou, Changqin Huang, Qintai Hu, Jia Zhu, and Yong Tang. 2018. Personalized learning full-path recommendation model based on LSTM neural networks. *Information sciences* 444 (2018), 135–152.