

PolyGloT: A Personalized and Gamified eTutoring System for Learning Modeling and Programming Skills

Antonio Bucchiarone^a, Tommaso Martorella^b, Davide Frageri^c, Diego Colombo^d

^a*Fondazione Bruno Kessler, Trento, Italy*

^b*École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*

^c*University of Trento, Italy*

^d*Microsoft Corporation Redmond WA, United States*

Abstract

The digital age is changing the role of educators and pushing for a paradigm shift in the education system. Individualization through adaptivity is crucial to nurture individual potential and addressing accessibility needs and neurodiversity. By formalizing a learning framework that takes into account all these different aspects, we aim to define and implement an open, content-agnostic, and extensible eTutoring platform to design and consume adaptive and gamified learning experiences. We present PolyGloT, a system able to help teachers to design and implement adaptive learning paths. We demonstrate through a concrete case study the use of a mixed narrative, that is a gamified learning path mixing lessons, quizzes and modelling exercises to learn concepts of the SysML-v2 modelling language.

Keywords: Adaptive Learning, 1-1 eTutoring, Modelling, Programming, Gamification.

Metadata

This ancillary data table is required for the sub-version of the codebase. Please replace the text in the right column with the correct information about your current code and leave the left column untouched.

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v0.1.0
C2	Permanent link to code/repository used for this code version	https://github.com/polyglot-edu/node-editor/tree/a492588e115aa879b58761fec1b01804c768dbfc https://github.com/polyglot-edu/backend/tree/e4dde78be2359b46c4f29150971fa0fd60e7135f https://github.com/polyglot-edu/runtime/tree/42d4f21e9eeca92ae8688ad42feb0ba5620579fc
C3	Permanent link to Reproducible Capsule	
C4	Legal Code License	MIT License
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	C#, TypeScript, .NET Interactive
C7	Compilation requirements, operating environments and dependencies	The PolyGloT editor for teachers is a web-based tool that does not require any compilation. For the students only Visual Studio Code (with the Polyglot Notebooks extension v1.0.3611020) is required to execute the learning paths locally.
C8	If available, link to developer documentation/manual	https://github.com/polyglot-edu/runtime/blob/07ce25d0ae7c913fa6dd0fab8b79ae74189a5d05/README.md - Two videos are available to show how PolyGloT works from the Teacher (https://youtu.be/dEaKxsCLTgw) and Student (https://youtu.be/gtp6-Bgn1hE) sides.
C9	Support email for questions	bucchiarone@fbk.eu

Table 1: Code metadata (mandatory)

1. Motivation and significance

Adaptive learning is the delivery of personalized learning experiences that address an individual's unique needs instead of a one-size-fits-all approach. It may be achieved through just-in-time feedback, personalized learning paths, ad-hoc resources, or another wide array of techniques. But why is it important? Education is universally recognized as one of the factors with the highest impact on society and the individual. The United Nations included education in their 2030 Agenda for Sustainable Development¹. UNESCO launched the Global Education Coalition² in response to the COVID-19 pandemic. The European Union created the Digital Education Action Plan (2021-2027)³ to foster and support the adaptation of educational systems in the digital age. This collective global effort is motivated by a continuously increasing technology availability and a rising global enrolment rate.

Learning is also crucial in the industry. The continuous technological disruptures are creating job positions in brand new fields⁴. Entirely new *hard skills* are required to fit the openings. From a student's perspective, effective teaching means **1-1 tutoring**⁵. It allows teachers to target specific misunderstandings and necessities with real-time feedback and explanations relevant to the student's experiences.

A study by K.R. Koedinger et al. shows that the "Doer Effect" is a causal association between practice and learning outcomes and that practising is six times more effective than reading [1]. Another significant drawback lies in the motivational aspect. **Active learning** is more effective in learning outcomes and motivation than passive learning [2]. Despite that, a recent article on PNAS by L. Deslauriers et al. highlights a negative correlation between actual learning and the feeling of learning in the students [3]. However, **gamification** and serious games gained consensus as tools to motivate people to engage in beneficial activities, even if seen as unrewarding or tedious [4, 5, 6, 7].

The design of a learning experience should take into account all of these different factors. On the other hand, **a design framework should not make assumptions about content type, form, delivery, and vali-**

¹<https://www.un.org/sustainabledevelopment/education/>

²<https://en.unesco.org/covid19/educationresponse/globalcoalition>

³<https://education.ec.europa.eu/focus-topics/digital-education/digital-education-action-plan>

⁴<https://www.linkedin.com/business/talent/blog/talent-strategy/linkedin-most-in-demand-hard-and-soft-skills>

⁵<https://www.eschoolnews.com/2022/07/26/ai-intelligent-bots/>

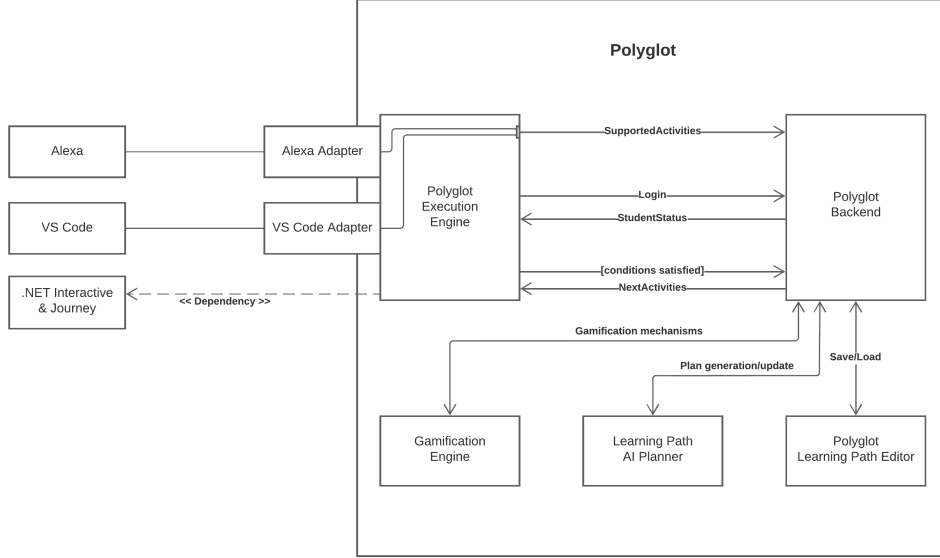


Figure 1: PolyGloT architectural overview.

dition while still removing any obstacle between the teacher, the student, the environment, and the learning experience.

Individual coaching is rarely feasible due to poor scalability, whilst one-to-many general training is scalable but lacks individualization altogether. That is the gap the Personalized and Gamified eTutoring System - PolyGloT - aims to fill. Adaptive eTutoring systems support both teachers and students by combining the benefits of individualized delivery and manageability by leveraging software personalization, while gamification can be used to enhance motivation through personalized rewards.

2. Software Architecture and Functionalities

In this Section, we present the overall architecture of PolyGloT and its functionalities. Then, we describe how it has been implemented.

PolyGloT aims to provide an open, content-agnostic and extensible framework (see Figure 1 for its architecture) for designing and consuming adaptive and gamified learning experiences. We imagine a student experience entirely tailored to their needs and choices. For example, we think students should be able to do some lessons and quizzes with Alexa, switch to VS Code to do some coding activities, and then move to another frontend (i.e., Moo-

dle⁶) to do something else, all without friction. That is why we exploit the flexibility of **.NET Interactive**⁷ in PolyGloT’s execution engine. Students’ interactions with external tools occur through **Adapters** that bind actions on the student frontend to **.NET Interactive** commands and bind **.NET Interactive** events to a supported rich output (e.g. audio for Alexa) with custom formatters.

The word **”adaptive”** in adaptive learning paths means being able to adjust the students’ needs, like proposing exercises based on their previous responses or the capabilities of the platform they are on. This adaptation is allowed by the collaboration between the **Execution Engine** and the **Backend**. The former handles the students’ submissions and validates their responses, while the latter uses the results of the validation phase to propose the next possible activities in the learning path.

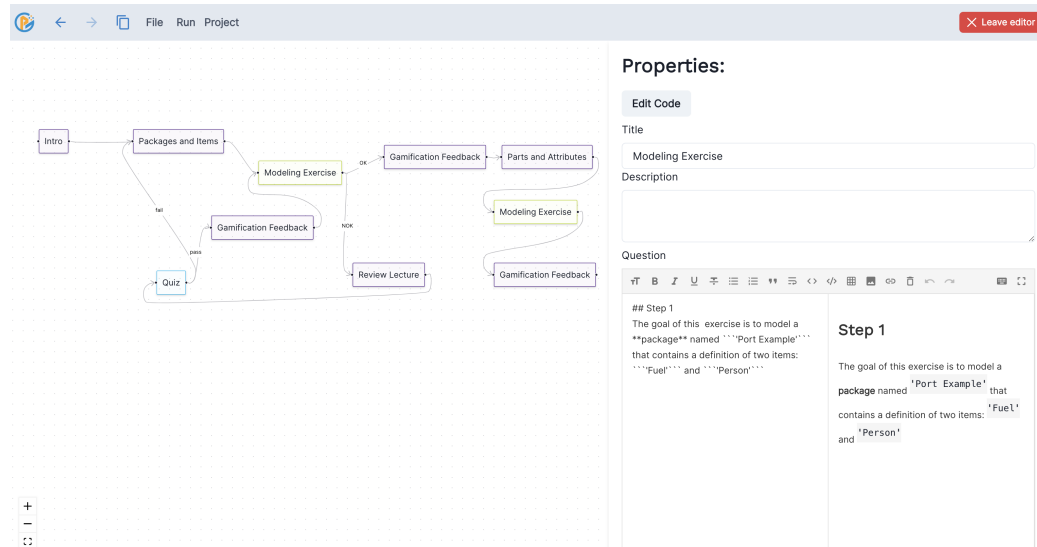


Figure 2: Teacher design tool for learning paths.

Designing and realizing learning paths for such a variegated learning experience might seem a daunting task at first. That is why PolyGloT includes a **Learning Path Editor**⁸ heavily focused on the teachers’ experience (see Figure 2). Its graph-like, visual-editing capabilities and integration with the rest of the platform allow the creation of ready-to-use learning paths with ease, thanks to the abstractions provided.

⁶<https://moodle.org/>

⁷<https://github.com/dotnet/interactive>

⁸<https://polyglot-edu.com/flows>

The teacher design tool is composed of two main elements: the *drawing area* and the *properties panel*. The drawing area is the core of the visual editing experience. Nodes can be added, connected, and rearranged with simple clicks or drag-and-drop interactions. Nodes and edges themselves can be interactive components that augment the visual editing experience. The properties panel, instead, provides fine-tuning tools to define the activities and the links between them.

To define the learning path, a teacher can create new activities with a right-click on the canvas, change their type and parameters from the properties panel, and connect them by dragging from the handle on the source node to the handle on the destination node. By clicking on an edge, the teacher can edit the link condition by choosing from a list of existing abstractions or by writing their own validation code. The tool is implemented using web technologies with Next.js⁹ and TypeScript¹⁰ with a particular focus on flexibility and extensibility. Its design allows it to be part of a more comprehensive learning management tool. The drawing area and the properties panel are different abstract views that operate on the same data differently. Both components manipulate the same PolyglotElements (i.e. nodes and edges) by working on the shared state via Zustand¹¹ actions.

Gamification means creating a game narrative that guides players through complex challenges, keeping them engaged with social activities such as group work or competitions. It means providing immediate feedback (as expected from a game-like environment) and students making autonomous choices to progress down the individually decided path. Gamification mechanics are fundamental to the learning path personalization process in two ways. Not only do they keep the students engaged, but they can also be used as tools to gain insight into the student's behaviour from a different perspective and thus help generate a more personalized and engaging learning path. For this purpose, the editor provides a way to give specific gamification feedbacks to students throughout the advancement of the game narrative, via specific nodes in the learning path.

2.1. Implementation

The teacher design tool tries to achieve two opposite goals: providing useful abstraction to allow teachers to define the learning paths naturally

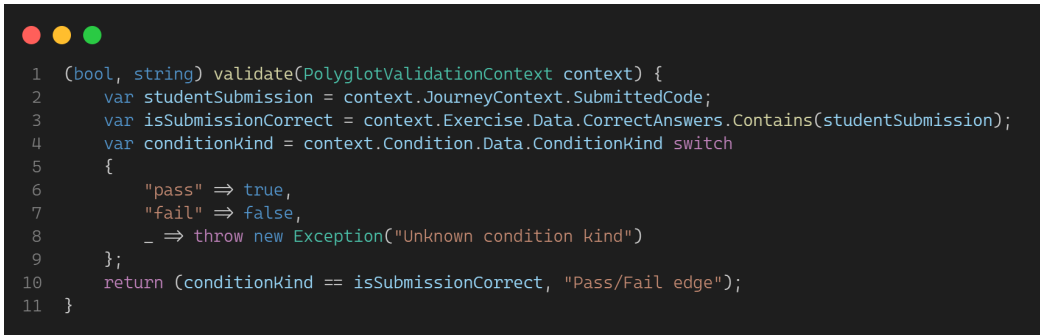
⁹<https://nextjs.org/>

¹⁰<https://www.typescriptlang.org/>

¹¹Zustand is a lightweight, unopinionated state-management library for React <https://github.com/pmndrs/zustand>

while remaining concrete enough to allow students to consume them without further intervention. This dual goal is especially challenging to achieve with edges that allow teachers to define links abstractly with clauses such as "pass" or "fail", but not all activities are suitable for that abstract condition (e.g. a coding activity). Activities and connections must then undergo a programmable transformation phase to be converted into their executable counterparts. Implementation details can be generated behind the scenes to bridge the gap between the abstract world of activities and connections and its concrete executable form. Furthermore, some needed values may depend on the combination of others, so generating them guarantees consistency. For example, edges correspond to **Journey**¹² rules in the execution phase. This correspondence is achieved by generating the validation code in this transformation phase.

Abstractly, a connection between two activities means that if a property on the source activity is satisfied, the student is allowed to pass to the destination activity. This process needs three components to work correctly: (1) information on the source activity; (2) information on the student submission, and (3) information on what property to enforce and how to do it. Suppose a quiz activity has an outgoing edge that enforces the property "pass". In this context, "pass" means that the student's answer must be among the correct solutions to the exercise. If we were to translate it into code, we would write a function that takes three inputs (the components mentioned above) and returns a boolean representing the validation result.



```

1 (bool, string) validate(PolyglotValidationContext context) {
2     var studentSubmission = context.JourneyContext.SubmittedCode;
3     var isSubmissionCorrect = context.Exercise.Data.CorrectAnswers.Contains(studentSubmission);
4     var conditionKind = context.Condition.Data.ConditionKind switch
5     {
6         "pass" => true,
7         "fail" => false,
8         _ => throw new Exception("Unknown condition kind")
9     };
10    return (conditionKind == isSubmissionCorrect, "Pass/Fail edge");
11 }

```

Figure 3: Pass/Fail abstract condition implementation

The actual code (Figure 3) is not too far from the previous description. It takes a `PolyglotValidationContext` as input and returns a pair (result, feedback). The `PolyglotValidationContext` type contains the three essential

¹²Journey is a .NET Interactive library for defining open-ended graphs of challenges

components: (a) a **Journey RuleContext** that holds information on the student submission; (b) an **Exercise** that contains information on the source activity; and (c) a **Condition** that contains information on the edge (e.g. if the condition is pass or fail).

2.1.1. Execution Engine

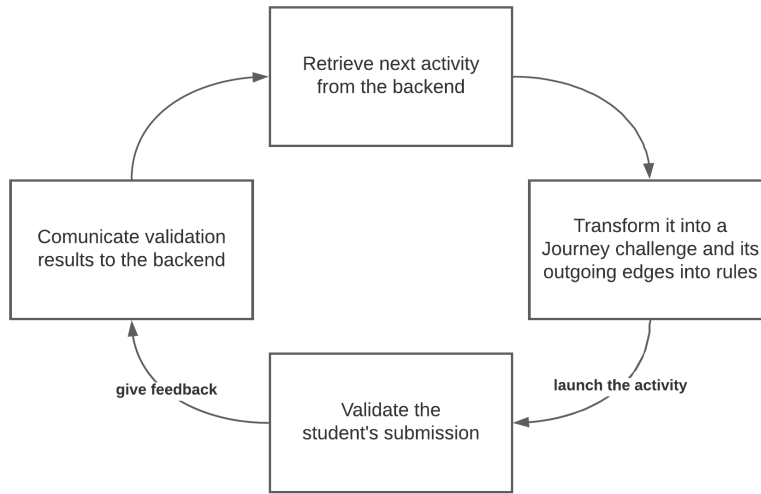


Figure 4: Execution Engine Lifecycle.

The **Execution Engine** component constitutes the core of the student experience. It is responsible for handling user interactions and validating students' submissions. Its straightforward lifecycle (see figure 4) is designed to exploit **Journey**'s strengths by using it in a semi-controlled fashion. When the assigned activity and its outgoing edges are retrieved from the backend, they undergo a transformation phase where the activity is converted into a **Journey** challenge and edges into **Journey** rules. This process is essential because it allows running the validation code mentioned previously. Once converted, the challenge is ready to be launched. The activity is "shown" to the student in their frontend, thanks to **.NET Interactive** formatters. When the active frontend adapter produces an event, **Journey** intercepts it and runs the registered rules against the student submission. PolyGloT then sends the result of this validation phase back to the backend which will use this additional knowledge to suggest the next suitable activity. At the moment of writing, the students can execute the learning path with a VS Code Notebook (see Figure 5).

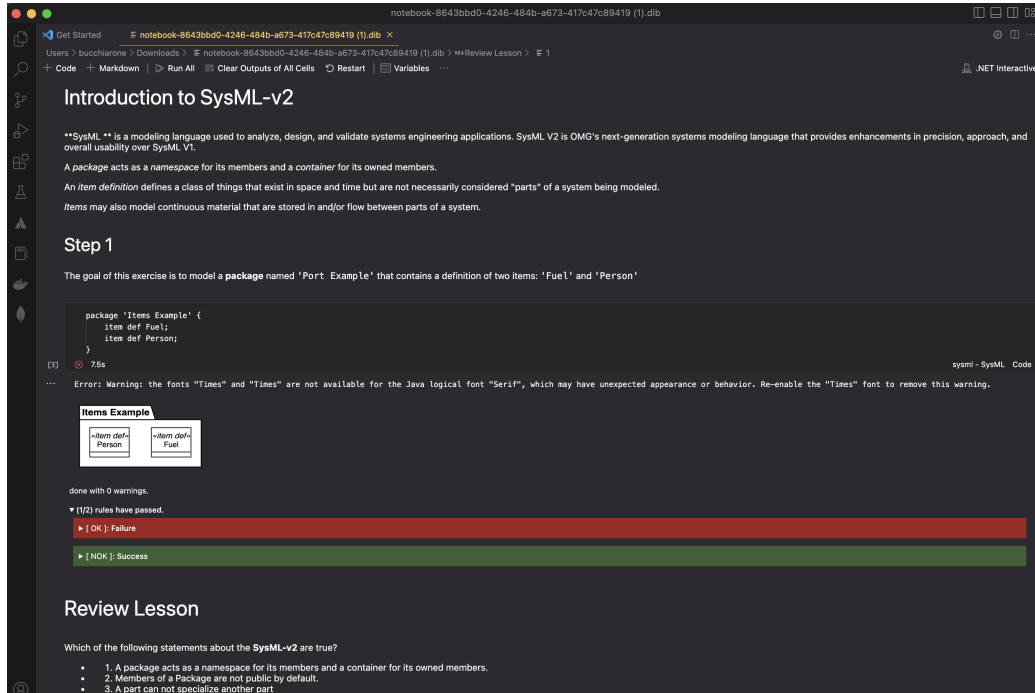


Figure 5: VS Code FrontEnd interaction.

3. Illustrative examples: A Learning Path for basics in SysML-v2

The chosen scenario to illustrate PolyGloT in execution is part of an introductory SysML-v2¹³ course that fits a broader Model-Driven Engineering program. Its goal is for the student to understand the concepts of **Packages**, **Items**, **Parts** and **Attributes**. The teacher has identified three kinds of concrete learning activities needed to define the scenario: lessons, quizzes and modelling exercises (as depicted in Figure 2). Each time that a student fails a modelling exercise, the path provides him a **Review Lecture** and an additional **Quiz** to reinforce the learning before proceeding. Once students have completed the first section, they move to the second part, where they can learn the remaining concepts (i.e., **Parts** and **Attributes**) following the same pattern as the first section. Gamification has the role of motivating the students while executing the learning activities. This is done thanks to a set of game elements (i.e., points, badges, levels) used to link the successful execution of learning activities to the advancement in the game narrative (i.e., Gamification Feedback nodes in Figure 2). Once a teacher is logged in

¹³<https://www.omgsysml.org/SysML-2.htm>

PolyGloT¹⁴, they can define a new learning path or open and re-use existing learning paths. The showcase page (as described in Figure 6) lists the available paths with the possibility to search them by keyword (i.e., learning concepts). Once a specific learning path is selected, it is opened in the modelling editor (as the one depicted in Figure 2).

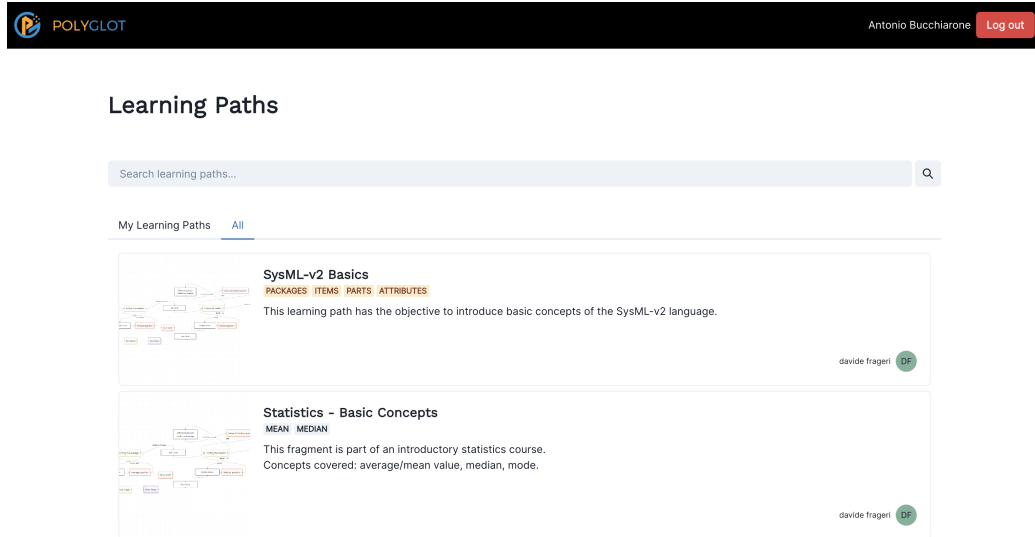


Figure 6: PolyGloT - Learning Paths Showcase.

The student can consume learning paths thanks to PolyGloT’s **Execution Engine** through standard VS Code notebooks (see Figure 5) just by loading the PolyGloT runtime NuGet package `Polyglot.Interactive` and choosing the activity to learn. To see how a learning path can be defined by a teacher and how it can be executed by students, two illustrative videos are available ¹⁵. To demonstrate the versatility of PolyGloT, we have implemented a second learning path (available on the showcase page) that targets an introductory statistics course that fits a broader data science program. We want students to first learn about the average, then the median, and then how to code them using the C# programming language.

4. Impact

PolyGloT, in its current state of development, already provides an interactive, engaging, and motivating experience for students learning modelling

¹⁴<https://polyglot-edu.com/flows/>

¹⁵<https://youtu.be/dEaKxsCLTgw>, <https://youtu.be/gtp6-Bgn1hE>

and programming. The flexibility of .NET Interactive allows PolyGloT to seamlessly mix variegated activities such as quizzes, modelling and coding exercises (and even mixing different programming/modelling languages) and create comprehensive learning paths that can be tailored to the student and accompany them to the learning goal while helping to revise the more complex concepts. Due to its expandability, it is an excellent foundation for quickly creating more content for learners (i.e., learning paths). This way, PolyGloT allows students to practice modelling and programming in a non-conventional way while leveraging the advantages of an active and constructive learning style. Although briefly touched on in this paper, the integration with other student frontends, such as Alexa, opens the possibility of novel student experiences where people can consume learning material on the go and learn in informal contexts.

5. Conclusions and Future Plans

In this paper, we presented an innovative eTutoring platform characterized by its adaptivity and gamification mechanics. An overview of its architecture has been presented together with a practical demonstration to deepen how processes are linked and how the framework would handle the activities' lifecycle.

The **adaptation process** necessary for effective eTutoring relies on *abstract activities*: activities defined only in terms of their goals (i.e., concepts to learn or skills to obtain) and not on concrete exercises the students have to solve (like those support today in PolyGloT). Those abstract activities are then replaced with the most suitable composition of existing learning fragments that satisfy the goal decided by an **AI planner**. The latter exploits the automatic generation of learning paths by adapting an existing approach based on AI planning presented in [8]. To make this extension possible, we need the availability of a substantial amount of learning fragments to choose from. We think that creating an **open learning fragments database** would benefit students (with better refinement and high-quality educational content) and teachers who can focus on the big picture and let the platform adjust the details.

We plan to add **support for existing learning management systems**¹⁶ (e.g. Moodle LMS) in the same modular fashion as we are doing with students' frontends. This extension would allow incremental integration of parts of PolyGloT within existing learning infrastructures and increase the adoption of adaptive learning technologies.

¹⁶https://en.wikipedia.org/wiki/Learning_management_system

The combined evolution of the engagement and learning systems is still an open research problem. Gamification mechanics are effective only when they fit the learning path perfectly. Usually, it is the gamification designer's duty to design the game narrative and the other game elements to achieve that fit. However, the power of adaptive learning paths comes from the run-time refinement; therefore, paths are not available upfront. The engagement system must then evolve through an **automatic calibration** phase onto the mechanics of the underlying activities.

Acknowledgements

This work is supported by the Erasmus+ 101055893 project, named "ENCORE -ENriching Circular use of OeR for Education", funded by the European Commission.

References

- [1] K. R. Koedinger, E. A. McLaughlin, J. Z. Jia, N. L. Bier, Is the doer effect a causal relationship? how can we tell and why it's important, in: Proceedings of the Sixth International Conference on Learning Analytics & Knowledge, LAK '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 388–397, <https://doi.org/10.1145/2883851.2883957>. doi:10.1145/2883851.2883957.
- [2] C. A. Benware, E. L. Deci, Quality of learning with an active versus passive motivational set, American Educational Research Journal 21 (4) (1984) 755–765, <https://doi.org/10.3102/00028312021004755>. arXiv:<https://doi.org/10.3102/00028312021004755>, doi:10.3102/00028312021004755.
- [3] L. Deslauriers, L. S. McCarty, K. Miller, K. Callaghan, G. Kestin, Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom, Proceedings of the National Academy of Sciences 116 (39) (2019) 19251–19257, <https://doi.org/10.1073/pnas.1821936116>. arXiv:<https://www.pnas.org/doi/pdf/10.1073/pnas.1821936116>, doi:10.1073/pnas.1821936116.
- [4] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, D. Dixon, Gamification. using game-design elements in non-gaming contexts, in: CHI'11 extended abstracts on human factors in computing systems, 2011, pp. 2425–2428.

- [5] A. Bucchiarone, A. Cicchetti, N. Bencomo, E. Loria, A. Marconi, Gamified and self-adaptive applications for the common good: Research challenges ahead, in: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2021, pp. 149–155, <https://doi.org/10.1109/SEAMS51251.2021.00028>. doi:10.1109/SEAMS51251.2021.00028.
- [6] S. Deterding, D. Dixon, R. Khaled, L. Nacke, From game design elements to gamefulness: defining” gamification”, in: Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments, 2011, pp. 9–15.
- [7] S. Bassanelli, N. Vasta, A. Bucchiarone, A. Marconi, Gamification for behavior change: A scientometric review, *Acta Psychologica* 228 (2022) 103657. doi:<https://doi.org/10.1016/j.actpsy.2022.103657>.
- [8] P. Bertoli, R. Kazhamiakin, M. Paolucci, M. Pistore, H. Raik, M. Wagner, Control Flow Requirements for Automated Service Composition, in: Proc. ICWS’09, 2009, pp. 17–24.