S. Götz, L. Linsbauer, I. Schaefer, A. Wortmann (Hrsg.): Software Engineering 2021 Satellite Events,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2021 1

Platform Architecture for the Diagram Assessment Domain

Meike Ullrich[1] Martin Forell[1] Constantin Houy[2] Peter Pfeiffer[2] Selina Schüler[1] Tobias Stottrop[3] Brian Willems[2] Peter Fettke[2] Andreas Oberweis[1]

Abstract: Using e-learning and e-assessment environments in higher education bears considerable potential for both students and teachers. In this contribution we present an architecture for a comprehensive e-assessment platform for the modeling domain. The platform – currently developed in the KEA-Mod project – features a micro-service architecture and is based on different inter-operable components. Based on this idea, the KEA-Mod platform will provide e-assessment capabilities for various graph-based modeling languages such as Unified Modeling Language (UML), Entity-Relationship diagrams (ERD), Petri Nets, Event-driven Process Chains (EPC) and the Business Process Model and Notation (BPMN) and their respective diagram types.

Keywords: domain specific e-assessment; diagrams; modeling; micro-service

1 Introduction

In software engineering, modeling in the form of graph-based diagrams plays an important role. This is also valid for computer science and information systems. As modeling is a crucial aspect for the qualification of software engineering experts, teaching modeling approaches at universities, especially using e-learning infrastructures gains more and more importance. In the above mentioned disciplines, learning and teaching modeling languages such as Unified Modeling Language (UML), Entity-Relationship diagrams (ERD), Petri nets, Event-driven process chains (EPC) or Business Process Model and Notation (BPMN), are essential core components of the curriculum [As18].

However, modeling is a complex topic as several different aspects need to be addressed (syntactic and semantic correctness and also pragmatic aspects). Furthermore, naturally there might be more than one correct solution for a particular modeling problem. In large university courses with a lot of participants, teachers struggle to provide individual feedback and coaching for every student. E-assessment systems can significantly support this and provide valuable feedback on student diagrams. While several different approaches regarding an automated diagram assessment have been proposed in recent years (e.g.,

[1] Karlsruhe Institute of Technology (KIT), Institute of Applied Informatics and Formal Description Methods (AIFB), Kaiserstr. 89, 76133 Karlsruhe, firstname.lastname@kit.edu
[2] German Research Center for Artificial Intelligence (DFKI) and Saarland University, Institute for Information Systems (IWi), Campus D3 2, 66123 Saarbrücken, firstname.lastname@dfki.de
[3] University of Duisburg-Essen, paluno - The Ruhr Institute for Software Technology, Gerlingstr. 16, 45127 Essen, tobias.stottrop@uni-due.de

[TWS06, SG14, Th16, CLP18]), these solutions have, however, remained stand-alone applications, often only in the form of prototypical implementations.

In the collaborative research project KEA-Mod[4], a methodically grounded, competence-oriented and modular e-assessment platform is developed. It is applicable to multiple assessment scenarios (formative/summative) and supports various modeling languages. The platform is supposed to merge existing technical applications of the participating project partners into a uniform, adaptable and also extensible concept. Therefore, the platform architecture has a modular structure and is based on different inter-operable micro-services. In this paper, the KEA-Mod platform architecture is introduced. The remainder of this article is structured as follows. In Section 2, related work will be presented, before the KEA-Mod platform architecture and components are described in Section 3 while explaining both non-technical requirements and the current technical realization. Section 4 closes the paper with an outlook on future complementing work.

2 Related work

Existing approaches regarding the diagram assessment domain described in academic literature can be categorized basically in two classes.

High level approaches describe architectures for diagram assessment with the goal to provide universal solutions for various diagram types and the respective modeling languages. Furthermore, they are often concerned with the integration of assessment functionality into a larger e-learning context. One example is the CourseMarker system [HB06]. Main components are different environments for authoring (i) diagram notations and (ii) exercises as well as a customizable student diagram editor. The marking component can be configured through a marking scheme describing how to to call specific marking tools. Another more recent system is Mooshak [CLP18], which has been extended to cover diagrammatic languages. An integrated learning environment offers a diagram editor as well as a diagram evaluator. A bridging component is designed to deliver feedback created by the diagram evaluator component to the student through the diagram editor.

In contrast, *low level approaches* are focused mainly on the process of assessment itself, mostly in a rather specific use case only, e.g. identifying defects in UML class diagrams [HR11]. Actually, much effort has been put into the assessment of UML class diagrams [SG11, RK19], other UML diagram types [SG14, TSW08] and ERD [TWS06, STW13] whereas only few approaches assess business process models like Petri nets [WFS13] or EPCs [Th16] in an educational context. Even though BPMN is considered a widely spread modeling language, no publications about assessing BPMN models in an educational context could be identified.

[4] Kompetenzorientiertes E-Assessment für die grafische Modellierung (KEA-Mod), funded by the German Federal Ministry of Education and Research (BMBF) (FKZ: 16DHB3022-16DHB3026), https://keamod.gi.de/

While the various approaches from the low level category cover many individual important aspects which should be assessed in student diagrams, systems from the high level category cover only a few diagram types up to now (ERD in CourseMarker [HB06] and ERD/UML-class diagrams in Mooshak [CLP18]). This underlines the need for a comprehensive (high-level) diagram assessment platform covering multiple diagram types from the considered disciplines. As the diagram types of concern are all based on graph structures (nodes connected via edges), existing (low-level) approaches for a specific diagram type can be transferred to other modeling languages or even adapted towards a cross-language assessment solution. The basic technical feasibility has been shown in e.g., [TSW08, Fe16].

3 Platform architecture

An overview of the constituent elements of the KEA-Mod platform is depicted in the UML component diagram shown in Fig. 1. The architectural design features a micro-service

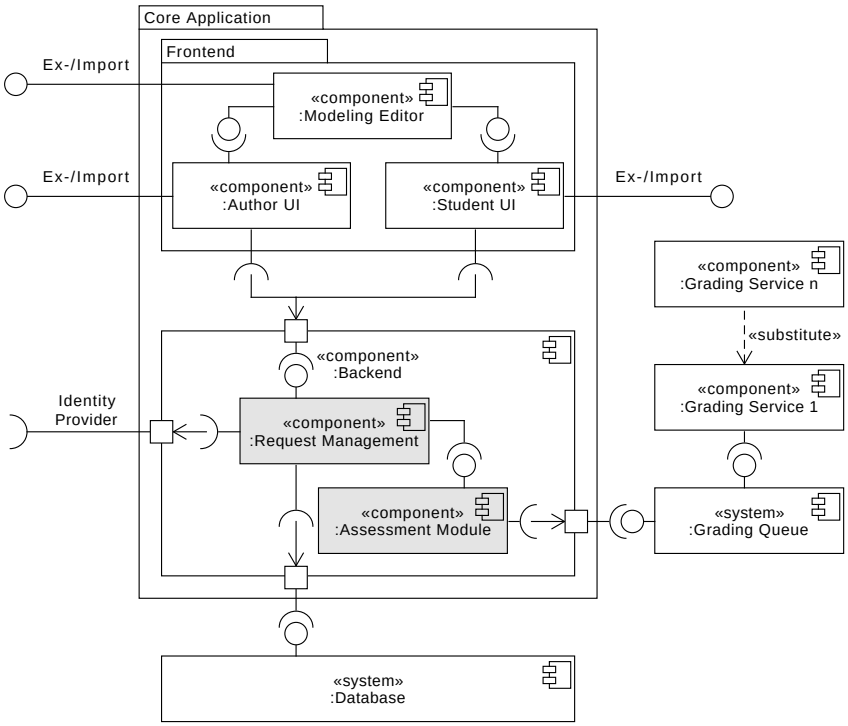

Fig. 1: KEA-Mod platform component model

structure which is based on different inter-operable components and systems. The Core Application and Frontend groups (visualized as packages) are each comprised of several

components. The Backend component has been refined into two significant internal sub-components (visualized as grey components). In the following sections, all components of the platform architecture are described in more detail.

3.1 Frontend

The Frontend is comprised of three components each serving as an independent web application: Author UI, Student UI and the Modeling Editor. The strict separation of those components has several advantages. It is possible to use the modeling editor as a standalone application for training or production purposes without having to log in beforehand and devoid of assessment functionality. Moreover, having separate user interfaces for the two main user groups of the platform provides a clear division of roles and their respective actions plus it requires a minimal set of authorization checks. The integrated export and import interfaces enable optional external support (see Sec. 4).

Users with the role *author* are granted access to the Author UI (e.g., lecturers, course instructors or tutors). The Author UI is a web application offering the functionality to create, configure and review assignments, exercises and grading schemas for the automatic assessment. An assignment contains a set of different exercises that should be solved by students (like a typical exercise sheet or written exam). It is possible to add established exercises to a public exercise pool. Thus, others can simply re-use already existing exercise material. The platform supports different exercise types which are designed to address student's modeling competences like diagram understanding and diagram creation. While diagram understanding exercises can be implemented with single or multiple choice questions, diagram creation exercises require students to draw a diagram using a modeling editor. To control how student's submissions to an exercise shall be automatically assessed, each exercise includes a specific grading schema. Using such grading schemas, the assessment can be adapted flexibly to a specific context as well as reflect individual grading preferences. Configuring the grading schema involves the selection of available grading services (see also Sec. 3.3) and choosing their desired weighting. While reviewing automatically assessed submissions, it is possible to add a manual note with individual remarks or to adapt the grading schema and rerun the assessment procedure.

The Student UI is a web-application intended for users of the role *student*. This functionality includes working on assignments by entering and handing in submissions to exercises and displaying the corresponding assessment results. The results are prepared by the Assessment Module in the Backend component (see also Sec. 3.2) according to the grading schema configured through the Author UI. Especially in a formative setup (self-directed learning), feedback plays an important role in gradually guiding students towards a correct submission. For this, textual feedback is generated by the grading services. In the case of exercises regarding diagram creation, textual feedback is also linked to the related diagram elements and displayed accordingly (e.g., using colours to highlight incorrect elements). For the sake of transparency, the assessment of competences through the grading schema should be

presented particularly in a summative setup (e.g., midterm or final exam). This implies not only showing total scores, but also partial scores and weightings together with a description of the involved grading services.

The Modeling Editor does not require a specific user role. Thus, it can be tightly integrated into both the Author UI and the Student UI. Furthermore, it can be be used independently as a standalone web-application. By offering graphical modeling elements of various modeling languages and diagram types, the modeling editor supports merging existing technical applications of the participating project partners and the development of cross-language grading services. To achieve this, a proprietary, generic data exchange format for the modeled diagrams is used. In addition, the user activities during the modelling process are recorded (completely anonymously) to study the modelling process. As mentioned before, the modeling editor can be configured through the Author UI to restrict or expand the available syntactical elements of modeling languages (or rather diagram types) and enable or disable live syntax checking prohibiting users from entering syntactically incorrect solutions. Such syntax checking might not be desirable if the learning objective to be assessed is the knowledge of a specific modeling language syntax.

3.2 Backend

One of the two core sub-components of the Backend component is the Request management which processes incoming requests via the Frontend components. The authentication and authorization of incoming requests is performed by querying external identity providers of the participating institutions and storing a pseudonymized identifier in the platform database. Furthermore, requests are either basic create, read, update and delete (CRUD) operations which will be passed on to the Database system, or they trigger the automatic assessment of solutions performed by the Assessment Module.

The Assessement Module is the second of the two core sub-components of the Backend. It takes the grading schema of an exercise and a student solution as input in order to automatically generate an assessment. For this purpose, the selected grading services are triggered by adding a corresponding request to the Grading Queue system (see also Sec. 3.3). As soon as all results are available and pulled from the Grading Queue, a total score of a submission is calculated based on the provided weights. The Assessment Module then returns the aggregated results together with the feedback generated by the grading services to the Request Management component, which passes it on to the Database system.

3.3 Grading Queue and Grading Services

The Grading Queue system stores requests to assess submissions from the Assessment module. Those requests are consumed by an appropriate Grading Service. A Grading Service

is an application which runs a specific analysis on a submission and returns the results (numerical score and feedback list) back to the Grading Queue. While some grading services might only be applicable to certain diagram types, others can be used for any diagram type due to the generic exchange data format. Examples are pragmatic quality metrics measuring the understandability of diagrams through e.g, counting edge crossings or checking if the flow direction established through the arc orientation is consistent within a given process model. The scope of grading services ranges from simple ones, which e.g. only check for syntactic correctness, to complex ones, which e.g. check a submission for semantic correctness, which might require the comparison of a student diagram with a sample solution diagram.

3.4 Non-functional requirements

The core non-functional requirements which are taken into account during the development of the KEA-Mod platform concern the usability design, data protection/security and technical expandability. The graphical design of the Frontend user interfaces should be simple, intuitive and consistent to allow for an efficient and easy usage. To this effect, media-pedagogic principles and usability guidelines are considered and a usability evaluation will be performed. To ensure data protection, the platform accesses no personal data at all and stores only an identifier given by the external identity provider. The KEA-Mod platform as such should be geared towards expandability. More precisely, it should be as easy as possible to extend the platform further by integrating new diagram types or grading services.

3.5 Technical realization

Regarding the implementation, modern, popular and platform independent technologies were chosen to increase future development by a broad basis of the community once the source code is published. The Frontend components Modeling Editor, Author UI and Student UI are designed as web applications with the JavaScript framework *React*, which allows the usage on various devices. For internationalization purposes the *i18next* framework is used. Besides that, the Frontend also utilizes *axios* as HTTP client and *Bulma* for layouting and design. The Modeling Editor makes use of the *JointJS* diagramming framework. The Backend is also written in JavaScript using *Node.js* and *Express*. It implements a RESTful API for communication with the other platform components. For the Database, we used the relational DBMS *MySQL*, which is both powerful and widely used. *Sequelize* provides an object-relational mapping (ORM) to the database. For realizing the Grading Queue, *Apache Kafka* and *Protocol Buffers (Protobuf)* were used. All KEA-Mod platform components shall be published under an open source license both as source code and easy to set-up *Docker* container images to ensure transparency and foster the development of extensions and further distribution of the platform in the higher education sector.

Due to the micro-service architecture, instances of a Grading Service can be implemented using any technology as long as they implement an appropriate communication interface. The grading service for assessing EPCs has already been realized in a comprehensive tool for automated process model analysis which also allows for automated model evaluation: the *RefMod-Miner* (RMM)[5]. Existing EPC assesment functionalities [Fe15, RFL17] from the RMM are migrated to the RMM4Py [Kl20] – a new python library – and will be used as grading service instances in the KEA-Mod application. While the previous version mainly used a CLI, the newly implemented version can be integrated as a python library. The widely recognized e-assessment platform *JACK*[6] serves as basis for the implementation of grading services for various UML diagrams, namely class, activity and sequence diagrams. These services implement different techniques like trace checking [SG14] or graph-based analysis based upon GReQL, an expression-based query language for graphs [BE08].

4 Outlook

Besides the development of the presented platform, the transfer into the practical application is also an important mission of the KEA-Mod project. In an upcoming pilot phase, the platform will be used in different courses of the participating project partners. Other institutions have already signalled interest using the platform during the final transfer phase of the project. One of the main challenges expected here is the integration of the platform into existing learning management systems (LMS) or other e-assessment systems.

Furthermore, the data collected through the platform can be subject of further research under strict data protection measures. A possibility already mentioned is the study of the modeling process observed and logged through the modeling editor. Moreover, the application of machine learning approaches bears tremendous potential for the development of sophisticated grading services. Here, automatically generated assessments (that have been manually reviewed and approved) might serve as training data.

An auxiliary prospective add-on comes in the form of exercise generators for parametric diagram understanding exercises. Here, answer options for multiple choice questions suitable for a specific diagram are generated automatically. This saves hours of tedious manual work, when authors are obliged to create a larger quantity of variations of similar exercises.

Bibliography

[As18] Association for Computing Machinery (ACM): , Curricula Recommendations. Online [https://www.acm.org/education/curricula-recommendations], 2018.

[BE08] Bildhauer, D.; Ebert, J.: Querying Software Abstraction Graphs. In: Working Session on Query Technologies and Applications for Program Comprehension (QTAPC). 2008.

[5] The RefMod-Miner website and a full function description can be found here: `https://refmod-miner.dfki.de`

[6] *JACK* has a demo platform, which can be accessed without registration: `https://jack-demo.s3.uni-due.de`

[CLP18] Correia, H.; Leal, J. P.; Paiva, J. C.: Improving Diagram Assessment in Mooshak. In: Proc. Technology Enhanced Assessment (TEA). pp. 69–82, 2018.

[Fe15] Fettke, P.: Integration von Prozessmodellen im Großen: Konzept, Implementierung und experimentelle Anwendungen. In: Proc. Wirtschaftsinformatik (WI). pp. 453–467, 2015.

[Fe16] Fellmann, M.; Fettke, P.; Houy, C.; Loos, P.; Oberweis, A.; Schoknecht, A.; Striewe, M.; Thaler, T.; Ullrich, M.: Evaluation automatisierter Ansätze für die Bewertung von Modellierungsaufgaben. In: Proc. E-Learning Fachtagung Informatik (DeLFI). pp. 203–214, 2016.

[HB06] Higgins, C.A.; Bligh, B.: Formative computer based assessment in diagram based domains. In: Proc. Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE). pp. 98—102, 2006.

[HR11] Hasker, R.W.; Rowe, M.: UMLint: Identifying defects in UML diagrams. Proc. Annual Conference and Exposition (ASEE), pp. 22.1558.1 – 22.1558.14, 2011.

[Kl20] Klein, S.; Lahann, J.; Mayer, L.; Neu, D.; Pfeiffer, P.; Rebmann, A.; Scheid, M.; Willems, B.; Fettke, P.: Business Process Intelligence Challenge 2020: Analysis and evaluation of a travel process. In: 10th Business Process Intelligence Challenge at the Int. Conf. on Process Mining (ICPM). 2020.

[RFL17] Rehse, J.-R.; Fettke, P.; Loos, P.: A graph-theoretic method for the inductive development of reference process models. Softw. Syst. Model., 16(3):833–873, 2017.

[RK19] Reischmann, T.; Kuchen, H.: A web-based e-assessment tool for design patterns in UML class diagrams. In: Proc. ACM/SIGAPP Symposium on Applied Computing. pp. 2435–2444, 2019.

[SG11] Striewe, M.; Goedicke, M.: Automated checks on UML diagrams. In: Proc. Annual SIGCSE Conf. on Innovation and Technology in Computer Science Education (ITiCSE). pp. 38–42, 2011.

[SG14] Striewe, M.; Goedicke, M.: Automated Assessment of UML Activity Diagrams. In: Proc. Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE). p. 336, 2014.

[STW13] Smith, N.; Thomas, P.; Waugh, K.: Automatic Grading of Free-Form Diagrams with Label Hypernymy. In: Proc. Learning and Teaching in Computing and Engineering (LaTiCE). pp. 136–142, 2013.

[Th16] Thaler, T.; Houy, C.; Fettke, P.; Loos, P.: Automated Assessment of Process Modeling Exams: Basic Ideas and Prototypical Implementation. In: Proc. Workshop zur Modellierung in der Hochschullehre (MoHoL). pp. 63–70, 2016.

[TSW08] Thomas, P.; Smith, N.; Waugh, K.: Automatic Assessment of Sequence Diagrams. In: Proc. Computer Aided Assessment Conference (CAA). 2008.

[TWS06] Thomas, P.; Waugh, K.; Smith, N.: Using Patterns in the Automatic Marking of ER-diagrams. In: Proc. Annual SIGCSE Conf. on Innovation and Technology in Computer Science Education (ITiCSE). pp. 83–87, 2006.

[WFS13] Westergaard, M.; Fahland, D.; Stahl, C.: Grade/CPN: A tool and temporal logic for testing colored Petri net models in teaching. In: Transactions on Petri Nets and Other Models of Concurrency VIII. pp. 180–202, 2013.