



Supporting Lecturers in Properly Using Digital Learning Environments

The stARS Approach

Tommy Kubica, M.Sc.

Dissertation

submitted to

Technische Universität Dresden, Faculty of Computer Science

in partial fulfillment of the requirements in order to obtain the academic degree

Doktor-Ingenieur (Dr.-Ing.)

Dresden, December 2021



Supporting Lecturers in Properly Using Digital Learning Environments

The stARS Approach

Tommy Kubica, M.Sc.

born on June 10, 1991 in Elsterwerda, Germany

Dissertation

submitted to

Technische Universität Dresden, Faculty of Computer Science

in partial fulfillment of the requirements in order to obtain the academic degree

Doktor-Ingenieur (Dr.-Ing.)

Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Technische Universität Dresden, Germany

first reviewer

Univ.-Prof. Dipl.-Ing. Dr. Andreas Bollin

Alpen-Adria Universität Klagenfurt, Austria

second reviewer

Prof. Dr.-Ing. Thorsten Strufe

Karlsruhe Institute of Technology, Germany

advisor

Date of Submission

October 8, 2021

Date of Defense

November 26, 2021

Abstract

In recent years, the adoption of *digital learning environments* has been proven as a suitable complement to traditional lectures, allowing to involve students more actively. However, current approaches lack at supporting both lecturers' individual teaching scenarios and collaborative activities. Thus, this thesis introduces an *adaptable collaborative learning environment* that enables lecturers to model and execute customized teaching scenarios. In addition to expressive means of adaptation, it includes collaborative functions which support group and peer interactions. The approach was implemented in a role-based prototype called scenario-tailored Audience Response System (stARS), demonstrating its applicability through seven well-known teaching scenarios. Furthermore, a thorough evaluation based on various user studies and lecture experiments confirmed the ability to support lecturers' individual teaching scenarios and integrate advanced collaborative activities into *digital learning environments*.

Statement of Authorship

I hereby certify that I have authored this thesis entitled “Supporting Lecturers in Properly Using Digital Learning Environments – The stARS Approach” independently and without undue assistance from third parties. No other than the resources and references indicated in this thesis have been used. I have marked both literal and accordingly adopted quotations as such. For the development of the utilized prototype, I received assistance from the following persons: Ilja Shmelkin, Robert Peine, Lidia Roszko, Sinthujan Thanabalasingam, Niclas Zellerhoff and Chang Hong. There were no additional persons involved in the spiritual preparation of the present thesis. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Dresden, 8th October 2021

Tommy Kubica, M.Sc.

Contents

Abstract	iii
Statement of Authorship	v
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	2
1.3 Objectives	3
1.4 Research Context	4
1.5 Outline	5
2 Fundamentals	7
2.1 Educational Foundations	7
2.1.1 Types of Lectures	7
2.1.2 Teaching Scenarios	9
2.1.3 Educational Technology	14
2.2 Conceptual Foundations	14
2.2.1 The Concept of Roles	14
2.2.2 The Fundamentals of Modeling	19
2.3 Summary	20
3 State of the Art and Related Work	21
3.1 Classification of Digital Learning Environments	22
3.2 The Functional Scope of Digital Learning Environments	23
3.2.1 Interaction Initiated by the Lecturer	24
3.2.2 Interaction Initiated by the Student(s)	27
3.3 Analysis of Existing Systems	28
3.3.1 Methodology: An Unstructured Literature Review	29
3.3.2 Findings	31
3.3.3 Problem Statement	32
3.4 Related Work	33
3.4.1 Methodology: A Two-Step Structured Literature Review	33
3.4.2 Didactic Strategies and Confirmation of the Problem Statement	37
3.4.3 Current State of Lecturer Support	40
3.4.4 Current State of Adaptation	43
3.4.5 Current State of Collaboration	45
3.5 Summary	47

4	An Adaptable Collaborative Learning Environment	53
4.1	Conceptual Idea	53
4.2	(Meta-)Model	55
4.2.1	Choosing an Appropriate Underlying (Meta-)Model	56
4.2.2	Structural Blocks	58
4.2.3	Transition Blocks	59
4.2.4	Functional Blocks	60
4.2.5	Blocks for Visualization	65
4.2.6	Preliminary Evaluation of the (Meta-)Model	67
4.3	Graphical Editor	70
4.3.1	Basic Concepts for Usability	71
4.3.2	General Design and Concept Decisions	71
4.3.3	Supporting Concepts	79
4.4	Runtime Environment	88
4.4.1	Frontend	90
4.4.2	Backend	92
4.4.3	Runtime-Cloud	93
4.5	Integration of Role Concepts	94
4.5.1	The Importance of Roles in Learning Environments	94
4.5.2	Role-based Modeling of an Adaptable Collaborative Learning Environment	96
4.5.3	Role-based Runtime for an Adaptable Collaborative Learning Environment	99
4.6	Adapting Scenarios at Runtime	104
4.7	Didactic Opportunities	107
4.8	Summary	108
5	stARS – The scenario-tailored Audience Response System	109
5.1	Global Picture on the stARS Components	109
5.2	(Meta-)Model	110
5.2.1	Choosing a Technology to Implement the (Meta-)Model	111
5.2.2	Implementing the (Meta-)Model	113
5.2.3	(Meta-)Model-Transformation to JSON	117
5.2.4	Define Constraints in the (Meta-)Model	119
5.3	Graphical Editor	120
5.3.1	Choosing a Suitable Library for a Graphical Editor	121
5.3.2	Structure and Functions of the Graphical Editor	123
5.3.3	Supporting Concepts	128
5.4	Runtime Environment	133
5.4.1	Backend	133
5.4.2	Runtime-Cloud	134
5.4.3	Frontend	135
5.5	Integration of Role Concepts	138
5.5.1	Implementing the Role Concept in stARS	138
5.5.2	Different Use Cases that Implement the Role Concept	140

5.5.3	An Outlook on the Tradeoffs of Using the Role Concept	141
5.6	Adapting Scenarios at Runtime	142
5.7	Example Scenarios	145
5.7.1	Interactive Learning Questions	145
5.7.2	Peer Instruction	146
5.7.3	Jigsaw Classroom	147
5.7.4	Think-Pair-Share	148
5.7.5	Learning Stations	149
5.7.6	Peer Feedback	151
5.7.7	Learners-as-Designers	152
5.8	Summary	156
6	Evaluation	159
6.1	User Studies	159
6.1.1	User Study on Usability (Mid 2020)	160
6.1.2	User Study on Teaching Scenarios (Early 2021)	163
6.1.3	User Study on Runtime Adaptation (Mid 2021)	171
6.2	Lecture Experiments	176
6.2.1	Lecture Experiment on Group Interactions (June 14, 2021)	176
6.2.2	Lecture Experiment on Peer Interactions (June 18 – June 28, 2021)	181
6.2.3	Lecture Experiment on Peer Interactions (June 23, 2021)	183
6.2.4	Lecture Experiment on Group Interactions (July 22, 2021)	187
6.3	RT1: Modeling Adaptation for Customized Teaching Scenarios	190
6.4	RT2: Runtime Adaptation for Adjusting Running Teaching Scenarios	194
6.5	RT3: Role Concept as a Promising Extension for Integrating Adaptation	196
6.6	Summary	197
7	Conclusions	199
7.1	Findings of this Thesis	200
7.2	Outlook	203
	Bibliography	207
	List of Figures	219
	List of Tables	223
	List of Listings	225
A	(Meta-)Model Parameters	A
B	Build Schemes	G
B.1	The Build Schemes of the Group Builder	G
B.2	The Build Schemes of the Peer Builder	I

C	User Study on Usability (Mid 2020)	K
C.1	Slides of the User Study	K
C.2	Questionnaire of the User Study	L
D	User Study on Teaching Scenarios (Early 2021)	S
D.1	Questionnaire of the User Study	S
D.2	Detailed Results of the Sorting Questions	AS
E	User Study on Runtime Adaptation (Mid 2021)	AY
E.1	Questionnaire of the User Study	AY
F	Questionnaire for the Lecture Experiments	BI
	Acknowledgments	BQ

“I love using technology to help students bring their unique perspectives & experiences to the conversation.”

Derek Bruff, 2019

1 Introduction

For decades, digitalization influences different areas of our daily lives and did not stop at the educational sectors. Especially in the constantly increasing field of higher education (cf. [UNE20]), it provides a promising opportunity to transform the way in which study programs are offered and courses are held. Distance learning, for instance, is no longer a utopia and already took 6 percent of all students in Germany in 2018 (cf. [Hoc20]), which is a total number of 160.000. Moreover, the recent CoViD-19 pandemic has proven the importance of digitalization to traditional courses, too (cf. figure 1.1 that visualizes the growing interest of the topic “E-Learning” in the past five years). Only by using digital tools was it possible to shift the latest semesters predominantly into virtual space and enable education despite curfews or social restrictions [AR20; Cra+20; Ebn+20; Rad+20].



Figure 1.1: The global search interest of the topic “E-Learning” over the past five years: The numbers represent the global search interest relative to the highest point on the chart for the given time [Tre21].

1.1 Motivation

Although digitalization advances continuously, universities tend to offer their courses as traditional readings, tutorials, practicals, or seminars. As a result, even during the pandemic, a lot of lectures were transferred nearly entirely synchronously into virtual space. Innovative teaching formats such as *inverted classroom scenarios*, in which synchronous formats (i.e., both lecturer and students are present at the same time) are

combined with asynchronous formats (i.e., the students learn independently using online posted material), have rarely been used.

This leads to known problems: Due to the interaction taking place mainly from the lecturer to the students, students continuously act as passive listeners, trying to absorb as much information as possible. Active learning processes, which lead to increased student performance (cf. [Pri04; Fre+14]), do not take place. Moreover, students are cognitively overloaded due to the amount of information they receive in a nonstop 90-minute lecture (cf. [SR78]).

Digital learning environments can solve these problems either by breaking up the lecture into smaller chunks through interactive activities such as answering multiple-choice questions (i.e., *audience response*), or by allowing students to discuss their open questions in a *digital backchannel* of the lecture (cf. [Mar07; Ebn+14]).

1.2 Problem Statement

While the benefits of *digital learning environments*, such as involving students more actively or improving students' learning performance, are known (e.g., [KL09; NE18]) and students accept these systems (e.g., [Wol+11; OK13]), they have not yet been used in the vast majority of courses. This is partially caused by the heterogeneity of existing approaches: Although there are numerous systems (cf. [Kub+19]), no standard has yet been defined. Instead, several systems provide their own individual approaches that are targeted to specific scenarios. Thus, despite the fact that lecturers coming to class have their own teaching strategy in mind [Bru19], they have to choose one or more systems to support it [Sch16], or even adjust their strategy to integrate an appropriate approach. Moreover, they have to create content such as suitable multiple-choice questions and schedule enough time, which involves content cuts in the lecture, to execute those activities and evaluate the results comprehensibly. In practice, this huge workload and lecturer's uncertainty on the system's functionality to support their personal teaching strategy often lead to misuse (e.g., students are not given enough time to answer a question), or no use at all [KL09; Kub+17].

Furthermore, it is not possible to support arbitrary teaching scenarios. Especially short-term group activities, which are intended to stimulate collaborative learning (cf. [Dil99]), can often only be carried out offline (cf. [Shm18]). Thus, they have rarely been used in the presence of the CoViD-19 pandemic, and even if they have been used, they were limited to class-wide chat applications (later referred to as *backchannel* approaches), in

which students only support each other voluntarily if they pursue a common goal, or *breakout rooms* that allow for discussions in manually formed groups.

1.3 Objectives

We intend to use the means of adaptation to allow lecturers with varying teaching strategies to integrate interactive activities into their individual lectures. Consequently, the main research question of this thesis is formulated as follows:

How can different levels of adaptation support the lecturer in properly using digital learning environments?

In order to provide this support, lecturers must not only be given the opportunity to represent their teaching strategy within the system, but rather they must be able to adjust it on the fly if the results of previous activities demand it. Thus, the above-posted research question is decomposed into the following two research theses:

RT1) Modeling adaptation allows lecturers to create customized teaching scenarios that support their individual teaching strategies.

RT2) Runtime adaptation allows adjusting teaching scenarios on the fly in order to respond to real-time results.

Furthermore, a third research thesis is formulated that considers the concept and implementation of an appropriate approach:

RT3) The concept of roles provides a promising extension to integrate the means of adaptation in digital learning environments.

RT1 and RT2 are evaluated by university lecturers from different disciplines and with varying prior knowledge on both *teaching* and *digital learning environments*. This is expected to produce the most meaningful results. In contrast, RT3 is evaluated by an analysis of a prototypical implementation of promising use cases.

1.4 Research Context

The research in this thesis mainly focuses on synchronous lectures¹, which are still used by the majority of university courses. Moreover, especially in such settings, the integration of interactive activities by *digital learning environments* provides a promising extension and therefore represents an interesting research area.

This thesis presents an approach of an adaptable learning environment that allows lecturers to support their individual teaching strategies through interactive activities. In the following, an overview of the main contributions (and publications) is presented:

- The concept of an adaptable collaborative learning environment [Kub19a; Kub19b],
- A (meta-)model as a development methodology to realize this concept [KSS19],
- A prototype of an adaptable collaborative learning environment, in which lecturers can model their individual teaching scenarios using a graphical editor [KRT20], that allows supporting both traditional classroom scenarios [Kub+20b] and live-stream lectures [Kub+20a],
- The proposal of role-based group formations and interactions to foster collaboration in these settings [KPB20],
- Two user studies that investigate lecturers' opinions regarding the modeling task and the comprehensibility of created models [Kub+21],
- A user study that evaluates a concept and implementation of runtime adaptation in order to extend scenarios on the fly.
- The usage of the prototype in four real teaching scenarios as well as the opinion of both lecturers and students.

However, the research is not limited to these contributions. Further investigations were made that will be presented throughout this thesis.

¹ In synchronous lectures, both lecturer and students are present at the same time. Thus, the learning takes place simultaneously.

1.5 Outline

The remainder of this thesis is structured as follows: Chapter 2 establishes the *fundamentals*, both from an educational and conceptual point of view. Afterward, the *state of the art* and *related work* are investigated. Thereby, the functional scope of existing systems as well as the importance of the research theses, which were listed in section 1.3, are summarized, and *related research* is presented, before the research theses are further decomposed and requirements are defined. Next, chapter 4 motivates the *concept of an adaptable collaborative learning environment* that is supported by the *paradigm of roles*. Afterward, chapter 5 describes its *implementation*. Then, in chapter 6, the *evaluation* is presented, in which the user studies and experiments conducted are presented, and the validity of the sub research theses is examined. Finally, the thesis concludes by verifying the *research theses* and *answering the main research question* before giving an *outlook* on promising topics to be addressed in the future.

2 Fundamentals

In this chapter, the fundamentals of this thesis are briefly presented. Therefore, first, an overview of the educational foundations is given before the conceptual principles are introduced that will later be applied when designing and implementing a solution.

2.1 Educational Foundations

This section starts by presenting the types of lectures followed by a variety of selected teaching scenarios with their underlying didactic principles and an outlook on the domain of educational technology. The goal is to contextualize the topic of this thesis into teaching and introduce key terms that are used later.

2.1.1 Types of Lectures

In general, lectures can be distinguished between synchronous formats, in which the learning takes place simultaneously, and asynchronous formats, in which it takes place at different times. Furthermore, hybrid formats exist that combine both synchronous and asynchronous ones.

Synchronous Lectures

The most common types of synchronous lectures are traditional *face-to-face* lectures, such as readings, tutorials, practicals, or seminars, in which the lecturer and his/her students meet at a specific time in a physical room. An alternative to this face-to-face format that is frequently used during the CoViD-19 pandemic are *live-stream* lectures, in which this physical room is replaced by a virtual room. However, the main characteristic is the same: Both lecturer and students are present at the same time [CBP21].

Asynchronous Lectures

In asynchronous lectures, learning takes place individually. Therefore, the teaching material, such as *lecture videos* (e.g., *lecture recordings* or *text over slides*) or *lecture scripts*, is shared with the students, who process it themselves “at their own pace and at times of day which are convenient for them.” [CBP21]

Thus, the quality of asynchronous lectures strongly depends on the variety and types of media included, as well as the means of interactions provided (e.g., questions for the preparation or post-processing of specific topics, as will be described in section 3.2).

Hybrid Lectures

Hybrid lectures combine the advantages of both synchronous (i.e., the ability to interact in real-time) and asynchronous lectures (i.e., working at own pace and at any time). A well-known method to realize such lectures is represented by the *inverted classroom model* (ICM)¹, in which “events that have traditionally taken place inside the classroom now take place outside the classroom and vice versa” [LPT00], e.g., the content is acquired (asynchronously) by the students while the synchronous phase is used to practice and strengthen it². Although using ICM has, for instance, the potential to promote students’ critical thinking [OP15], several challenges exist, which are mostly related to out-of-class activities, such as students have not prepared adequately (cf. [AA18]).

Note on the Importance of Synchronous Formats

While the benefits of asynchronous learning cannot be understated, using synchronous formats can be critical to motivate students and foster collaboration either between the students or between the lecturer and the students [CBP21; Hra08]. Furthermore, it allows lecturers to react to students’ progress of learning immediately and thus, influence their learning. These are some of the reasons why synchronous formats are still used by the majority of university courses and will be the main focus of this thesis.

¹ In the context of primary and secondary education, the term *flipped classroom* is often used [BS12].

² There is no single model for *inverted classrooms*. Instead, core features can be recognized (cf. [OP15]).

2.1.2 Teaching Scenarios

Although there exists a large variety of teaching scenarios (cf. [Rei12; Pro21]), throughout this thesis, we will focus on seven well-known scenarios with different amounts of complexity, which should help to visualize the adaptability and expressiveness of our approach. Therefore, we have specifically selected teaching scenarios that go beyond traditional teaching methods such as *frontal teaching*, *PowerPoint presentations*, or *student presentations*. Each of the selected scenarios is briefly presented in the following.

Interactive Learning Questions

A common method to stimulate an active engagement with the learning content is presented by *interactive learning questions* [KNP04; Kap+14]. They can be used in different stages of the learning process with different functions [KNP04]. For example, within a synchronous lecture, they can be integrated at three points in time (cf. [Kap+14]): *At the beginning* of the lecture, *interactive learning questions* are used to activate the prior knowledge of students. When used *during* the lecture, students can practice the previously learned concepts and receive feedback on their current learning progress. Similar goals are provided when integrating *interactive learning questions at the end* of the lecture, i.e., important concepts can be repeated, and the learning progress can be assessed.

Note on the Relation to Self-Regulated Learning

Self-regulated learning can be understood as “a targeted, cyclical process during which the learner systematically applies cognitive, meta-cognitive, and motivational strategies to plan, execute, and reflect on the learning process, thus optimize his or her learning process” (translated from [PD20] and referring to [Zim00]). Using *interactive learning questions* allows supporting the learning process on both a cognitive and a meta-cognitive level [Kap+14; Kap14].

Peer Instruction

A well-known method to actively engage students and uncover difficulties with the material is *Peer Instruction* and typically works as follows [CM01; MH97]: Each 90-minute lecture consists of three to four short presentations focusing on specific topics (*Brief Lectures*). Each of these presentations is followed by a conceptual question (*ConcepTest*),

in which students' gained knowledge is assessed and presented to the lecturer. After the *ConcepTest*, the students discuss their given answers with the ones sitting around them (*Peer Discussion*). The goal is to convince each other that the personally given answer is the correct one. Finally, the *ConcepTest* is repeated, whose results should have changed due to the discussion. *Peer Instruction* has been proven to increase students' learning [CM01]. In literature, there exist numerous variants to implement *Peer Instruction*. A schematic representation created by [LMW08] is visualized in figure 2.1 and extends the process described above as follows: Depending on the results of the *ConcepTest*, the lecture proceeds differently. If a low number of students answered correctly ($< 30\%$), the *Brief Lecture* is revisited. If a high number of students answered correctly ($> 70\%$), the solution is explained and the next topic follows, and if some students answered correctly and others did not, the *Peer Discussion* is executed.

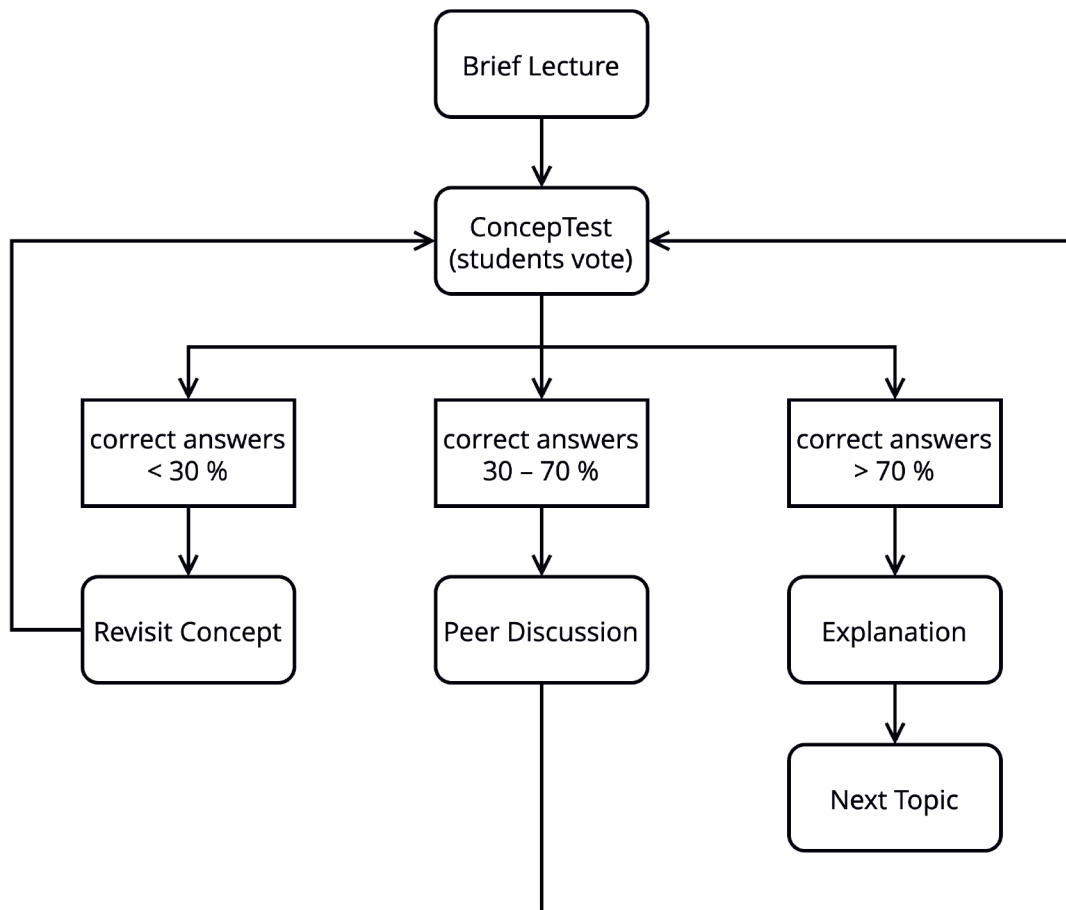


Figure 2.1: The schematic representation of a variant of *Peer Instruction* [LMW08]. Please note that the percentages presented are estimates, as those strongly depend on both the topic and the student population.

Jigsaw Classroom

Another alternative to traditional lectures is presented by the cooperative learning method³ called *Jigsaw Classroom* [Aro78], which has been considered effective in increasing positive educational outcomes [MX10]. Instead of grouping the entire class around the lecturer, the students work in smaller, interdependent groups. This is commonly done in three steps:

1. In the example visualized in figure 2.2, the students are first split into five *home groups* and each individual group member is assigned a topic to be studied in order to become an “expert” of it.
2. Next, the group members that have been assigned the same topic meet each other in an *expert group*, which allows comparing and discussing the results.
3. Finally, the “experts” return to their *home group* and present the results of their individual topic to the other group members.

The goal is to solve larger problems consisting of different sub-topics while actively involving all students.

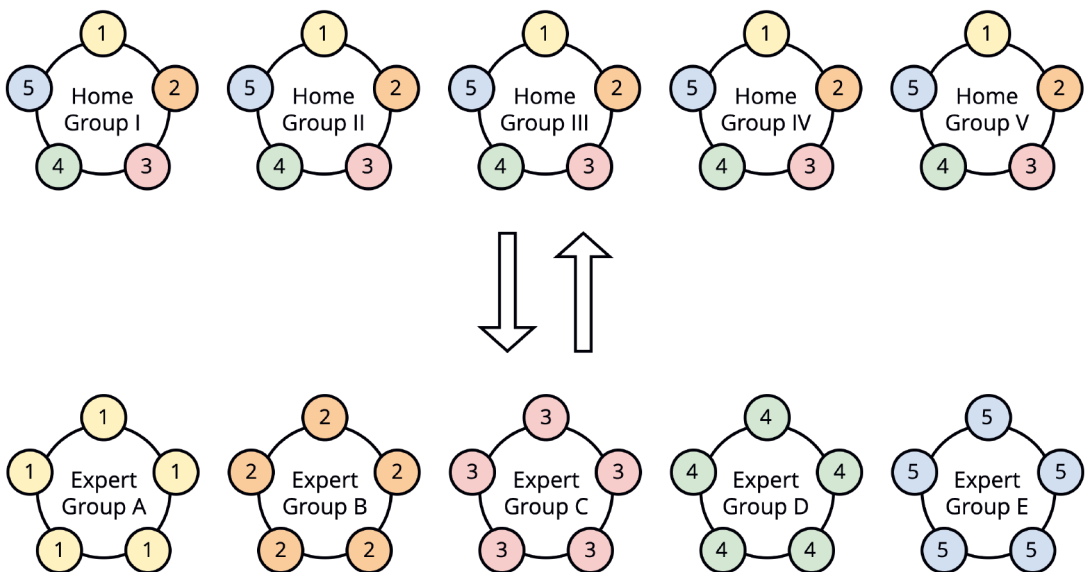


Figure 2.2: The formation of home groups (I to V) and expert groups (A to E) in a Jigsaw Classroom for 25 students (redrawn from [Nus+19]).

³ The term *cooperative learning* “refers to classroom techniques in which students work on learning activities in small groups and receive rewards or recognition based on their group’s performance” [Sla80].

Think-Pair-Share

Think-Pair-Share is another possible implementation of the cooperative learning method. Even though it also consists of three stages (similar to the *Jigsaw Classroom*), it works quite differently [Kad13; Rob06]:

1. Instead of splitting a task into several sub-tasks that are studied by different “experts,” the lecturer “provokes students’ thinking with a question, prompt, or observation” [Kad13]. The students receive several minutes to *think* about this question.
2. Afterward, the students *pair* up to discuss the answer they found. The goal is to identify the answer they think is the most convincing or unique one. This step can also be repeated multiple times, i.e., two pairs build a group of four students and discuss the answers they identified previously.
3. Finally, the pairs (or groups) are asked to *share* their opinion with the whole class.

This method has been proven to promote students’ critical thinking [Kad13].

Learning Stations

Learning Stations describe a strategy for promoting *active learning* (cf. [Bra16]) during all stages included in the educational process. In general, *learning stations* can be understood as physical locations in the classroom, in which students (or groups of students) have to complete an activity, such as solving a problem or answering several questions using the provided material. These activities have to be rather straightforward to enhance students’ learning effectively [Sch95]. After completing a station (or an impulse was given by the lecturer), students move forward to the next station. Using *learning stations* does allow lecturers both to reduce the amount of material and time required to set up, as well as help students with complex concepts [Sch95].

Peer Feedback

Following [Nar08], feedback can be defined as “all post-response information that is provided to a learner to inform the learner on his or her actual state of learning or performance,” while the source of feedback can be either external (e.g., peer or lecturer) or internal (the learner). Thus, the term *peer feedback* “refers to an activity in which

students offer each other comments, critiques, and advice about each other's work" [Dem15; LC06]. An exemplary scenario could proceed as follows:

1. First, the students upload their submissions, for which the feedback should be received.
2. Next, these submissions are distributed to one or several students that review them and submit their feedback.
3. Finally, it is sent back to the creator of the initial submission, who is able to edit his/her submission accordingly.

It is important to note that students do not necessarily only benefit from receiving feedback. Instead, providing feedback might promote higher-order and critical thinking skills [Dem15; LY11], which provides the necessary means to self-regulate their individual performance [NM06].

Learners-as-Designers

Learners-as-Designers describes a student-centered approach, which is based on an observation made by [Jon96] that "ironically, the people who seem to learn the most from systematic instructional design of instructional materials are the designers themselves." Consequently, students are taken in the role of an instructional designer, who will develop computer-based representations of topics (for example, a digital wallpaper) that can be used by other students for learning purposes [Dam17]. The process of acquiring content knowledge can be divided into four phases [Dam17; LR97]:

1. First, in the *planning* phase, the media is planned by the students, e.g., the topic of the project is defined, or source materials are searched and read.
2. Next, in the *design* phase, the students organize and structure information to define an outline for their project.
3. Afterward, different representations of the project are created in the *production* stage.
4. Finally, the project is evaluated and revised (*revising* stage).

The order of processing these stages is not always the same. Instead, the current stage, as well as prior stages, can be repeated. This also includes spontaneous adjustments, e.g., the implementation of a presentation, if the results indicate its necessity.

2.1.3 Educational Technology

The integration of technology provides a promising opportunity to facilitate learning. A discipline closely related to this study is described by *educational technology* (often used synonymously with *instructional technology* [Rei18]) that can be defined as “the study and ethical practice of facilitating learning and improving performance by creating, using, and managing appropriate technological processes and resources” [JM13]. There exist a vast amount of concepts targeting to improve different teaching scenarios.

Note on Terminology Used

In this thesis, the term *digital learning environments* is used to describe the vast amount of existing systems that enable interactions with the students or between them, while especially focusing on synchronous lectures.

Referring to the teaching scenarios presented in the last subsection, *digital learning environments* provide the means to actively involve students. For example, approaches such as *Audience Response Systems* (also called *Student Response Systems* or *Classroom Response Systems*) (cf. [KL09]) can be used to realize *Interactive Learning Questions* or the *ConcepTest* of *Peer Instruction*. Another example is provided by *Backchannel Systems* [Ebn+14; Yar08], allowing to initiate discussions in a digital backchannel of the ongoing lecture.

The functional scope of *digital learning environments* is explored in section 3.2.

2.2 Conceptual Foundations

In this section, the conceptual foundations are presented that are used in the course of this thesis to design and implement a solution. Therefore, the concept of roles is motivated as a promising opportunity to implement the means of adaptation before the fundamentals of modeling are described.

2.2.1 The Concept of Roles

In 1977, [BD77] presented the *role data model* as an extension of the network model, commonly known as the first data model that explicitly introduced the notion of roles. Since then, *role-based modeling* has been continuously studied as a means to model

complex and dynamic systems. As by 2000, the concept of roles was used in various fields in computer science, [Ste00] surveyed its state of the art. While he could identify a list of 15 features of roles to compare these approaches⁴, he also noticed that “the influence of the role data model on modelling has at best been modest” [Ste00]. In 2014, another survey was presented [Küh+14] that extended this list by additional features to describe the context-dependent nature of roles that approaches since 2000 have been shifting to – the complete list of role features can be retrieved from table 2.1.

However, the authors could also show the “discontinuity and fragmentation of the whole research field,” which is caused by the problem that existing modeling languages were often limited to the behavioral, relational, or context-dependent nature of roles instead of combining them [Küh+14]. As a solution, they proposed a family of (meta-)models for role-based modeling languages, called Compartment Role Object Model (CROM) (cf. [Küh+14; Küh17]). The types of languages are summarized by [Leu17]:

- The *relational nature* of roles is described by languages such as UML, wherein roles are named places in relations, e.g., a *person* can *take a course* and thus, is in the role of a *student*.
- Languages that focus on the *behavioral nature* of roles describe the adaptation of objects’ behavior and structure. For instance, if a *person* has the role of a *student*, he/she has a *matriculation number* and can *access the script of the course*. A well-known example for those languages is the contextual role language OT/J [Her05] that adapts the behavior and structure of objects regarding specific situations.
- There is currently no known language that focuses solely on the *contextual nature* of roles, i.e., the ability to play a role depends on its availability in the current context. For example, in order to play the role of a *student*, the *person* has to be in the context of a *university*⁵.
- With HELENA [HK14], an approach exists that *combines the relational and contextual nature* of roles. Therefore, the role (i.e., the association between objects) is embedded within a context.
- Furthermore, there also exist approaches that *combine the relational and behavioral nature* of roles, e.g., ORM [Hal98].

⁴ This is to be understood as a list of features of available approaches at this time.

⁵ Of course, similar contexts such as *schools* would also be suitable.

Table 2.1: A summary of the features of roles (slightly adjusted version of [Küh+14]’s extension of the role features list from [Ste00]). *M1* and *M0* are appended to denote whether the type or the instance level is affected by a feature.

<1>	Roles have properties and behaviors.	(M1, M0)
<2>	Roles depend on relationships.	(M1, M0)
<3>	Objects may play different roles simultaneously.	(M1, M0)
<4>	Objects may play the same role (type) several times.	(M0)
<5>	Objects may acquire and abandon roles dynamically.	(M0)
<6>	The sequence of role acquisition and removal may be restricted.	(M1, M0)
<7>	Unrelated objects can play the same role.	(M1)
<8>	Roles can play roles.	(M1, M0)
<9>	Roles can be transferred between objects.	(M0)
<10>	The state of an object can be role-specific.	(M0)
<11>	Features of an object can be role-specific.	(M1)
<12>	Roles restrict access.	(M0)
<13>	Different roles may share structure and behavior.	(M1)
<14>	An object and its roles share identity.	(M0)
<15>	An object and its roles have different identities.	(M0)
<16>	Relationships between roles can be constrained.	(M1)
<17>	There may be constraints between relationships.	(M1)
<18>	Roles can be grouped and constrained together.	(M1)
<19>	Roles depend on compartments.	(M1, M0)
<20>	Compartments have properties and behaviors.	(M1, M0)
<21>	A role can be part of several compartments.	(M1, M0)
<22>	Compartments may play roles like objects.	(M1, M0)
<23>	Compartments may play roles which are part of themselves.	(M1, M0)
<24>	Compartments can contain other compartments.	(M1, M0)
<25>	Different compartments may share structure and behavior.	(M1)
<26>	Compartments have their own identity.	(M0)
<27>	The number of roles occurring in a compartment can be constrained.	(M1)

- At the time of writing, only one approach exists that *combines all three natures* of roles, namely CROM [Küh17].

In the following, the fundamental concepts of CROM are summarized to understand the principles of the concept of roles.

In general, CROM specifies four *meta-types*, i.e., *natural types (NT)*, *compartment types (CT)*, *role types (RT)* and *relationship types (RST)*, that can be distinguished by investigating the following ontological *meta-properties* (cf. [AG13]), as summarized in table 2.2:

- *Rigidity*: An instance must be a member of this type for its whole lifetime.
- *Foundedness*: A type cannot exist without the existence of another type.
- *Identity*: The *identity* of a type's instance can either be unique, derived or composed.

Table 2.2: A summary of CROM's *meta-types* that are distinguished by their ontological *meta-properties* (adjusted version from [Küh17] that was created by [Leu17]).

Meta-Type	Rigidity	Foundedness	Identity	Example(s)
Natural Types	rigid	non-founded	unique	<i>person</i>
Compartment Types	rigid	founded	unique	<i>university</i>
Role Types	anti-rigid	founded	derived	<i>student</i>
Relationship Types	rigid	founded	composed	<i>takes course</i>

These four *meta-types* are connected by *relations* and *functions* (cf. [Küh17; Leu17]):

- The *fills relation* binds player types with role types, i.e., it defines that objects with specific player types can only play roles of a specific type. E.g., the role *student* can only be played by the player type *person*.
- The *parts function* maps each role type to a compartment type, e.g., the role type *student* is mapped to the compartment type *university*.
- The *rel function* maps relationship types to specific role types, which are part of the same compartment type. E.g., the relationship type *takes course* is mapped to the role type *student* – both are located inside the compartment type *university*.

To summarize, CROM is defined as a tuple over the *meta-types*, *relations* and *functions*: $M = (NT, RT, CT, RST, fills, parts, rel)$.

Moreover, an instantiated version of CROM, the Compartment Role Object Instance (CROI) was presented ([Küh17]), which is defined as a tuple: $I = (N, R, C, type, plays, links)$, whose concepts are described as follows (cf. [Leu17]):

- *Natural* (N), *Role* (R) and *Compartment* (C) are the instantiated versions of their specific types (i.e., NT , RT , CT), e.g., the instance of a *person*, the instance of a *student*, or the instance of a *university*.
- The *type* function is used to return the type of an instance (i.e., NT , RT , CT), e.g., calling *type* for the instance of the *student* returns the type *student*.
- The *plays* relation assigns a player to a role and thus, describes the instantiated version of the fills relation. An example is the assignment of an instance of a *student* to an instance of a *person*.
- The *links* function is used to return the roles for a relationship type, e.g., calling *links* for the relationship type *takes course* returns the instances of the role *student*.

Finally, a Constraint Model was proposed [Küh+15] to constrain roles and relationships (cf. [Leu17]):

- In the *role-implied constraint*, an object that plays a role of type A must also play a role of type B , but not necessarily the other way around, e.g., when playing the role *student*, the role *attendee* must also be played. However, when playing the role *attendee*, *student* does not have to be played automatically, as external people may attend the course as well.
- In the *role-equivalent constraint* instead, the relationship is bidirectional, i.e., if the object that plays a role with type A is required to also play a role with type B , then it must also play A if it is attempting to play B . For example, when playing the role *lecturer*, the role *presenter* must also be played. Moreover, when playing the role *presenter*, the role *lecturer* has to be played as well.
- In the *role-prohibited constraint*, an object that plays a role of type A is not allowed to play a role of type B and vice-versa, e.g., when playing the role *student*, it is not allowed to play the role *lecturer*, and when playing the role *lecturer*, it is not allowed to play the role *student*.

- In the *role-dontcare constraint*, no constraint is applied for an object that plays a role of types *A* or *B*. For example, the roles of *student* and *questioner* do not constrain each other.

2.2.2 The Fundamentals of Modeling

When talking about *models*, we usually refer to the abstraction of an object or a structure in the real world, targeting to ease both the understanding and usage of it. Thus, in computer science, *models* are a common method to represent references of the real world in a digital manner [KR08; PSW08; Tha21]. If models and modeling themselves are subject of a modeling process, we refer to *(meta-)modeling*, whereas “meta” can be understood as applying an operation repeatedly [Str+98; Str16].

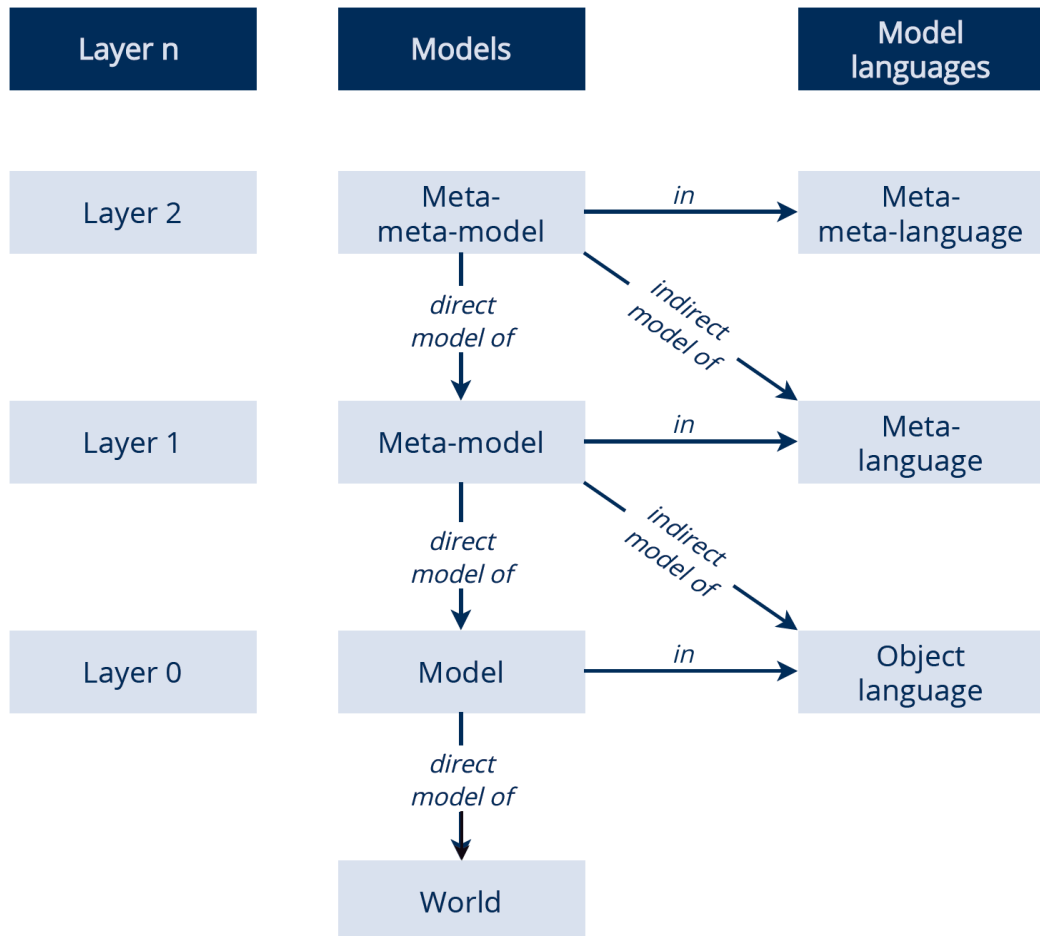


Figure 2.3: The definition of (meta-)models based on (meta-)model languages (translated and reduced version of [Str16] that is retrieved from [Tha21]).

As visualized in figure 2.3, each layer of abstraction n can be further abstracted by another layer $n + 1$ on top of it [Tha21]. When discussing a concrete model or modeling language (i.e., the *concrete syntax*), it is not specified which layer it is located. For example, the language UML (Unified Modeling Language) that is specified by the (meta-)language MOF (Meta Object Facility) can be used as *object language*, *(meta-)language*, or *(meta-)meta-language* (depending on the layer of the associated model) [Shm19].

Note on DSLs and DSMLs

The terms *domain-specific languages (DSLs)* and *domain-specific modeling languages (DSMLs)* are defined by three main components: *abstract syntax*, *concrete syntax*, and *semantics* (cf. [Kle08]). (Meta-)models are used to define the *abstract syntax*, i.e., the domain concepts and rules.

2.3 Summary

In this chapter, the fundamentals that will be used in the course of this thesis were briefly presented. Therefore, both educational and conceptual foundations were introduced:

Section 2.1 started by summarizing the types of lectures, namely synchronous, asynchronous and hybrid lectures. Afterward, seven well-known teaching scenarios (i.e., *Interactive Learning Questions*, *Peer Instruction*, *Jigsaw Classroom*, *Think-Pair-Share*, *Learning Stations*, *Peer Feedback*, and *Learners-as-Designers*) including their underlying didactic principles were described. Finally, an outlook on the domain of educational technology was provided.

Next, section 2.2 presented the conceptual foundations that will be used when designing and implementing a solution. Therefore, first, the concept of roles was introduced as a promising concept to realize the means of adaptation, before the fundamentals of modeling were briefly summarized.

3 State of the Art and Related Work

In this chapter, the research field of *digital learning environments* to support lectures in higher education is explored. We will focus on approaches following the *bring your own device* (BYOD) mantra, in which students use their personal mobile devices to participate actively through interactive activities. This will enable its implementation in any future lecture, as students inherently own the required hardware devices and only a stable Internet connection needs to be provided. Furthermore, it was already evaluated to be well accepted by both students and lecturers [CGC16; FS20].

This chapter is structured as follows: In the first section, different classifications of interactive activities in *digital learning environments* are presented and discussed regarding their application in this thesis. Next, common activities are summarized using the previously presented classifications. In the third section, an unstructured literature review is described, in which several problems of current *digital learning environments* are exposed. Afterward, the current state of research is systematically examined for each problem statement. Finally, a tabular summary is presented that shows how different approaches address different functional requirements. This will help to visualize the research gap that is tackled by this thesis. Moreover, both research theses and non-functional requirements are listed.

Note on the Requirements

In the course of this chapter, both functional (*FR*) and non-functional requirements (*NFR*) are retrieved that are used in chapter 4 to design the concept.

In order to specify functions that have to be supported to fulfill a *FR*, the following symbols are used:

- ▣ marks functions that have to be supported to partially support a *FR*.
- marks functions that have to be supported to fully support a *FR*.

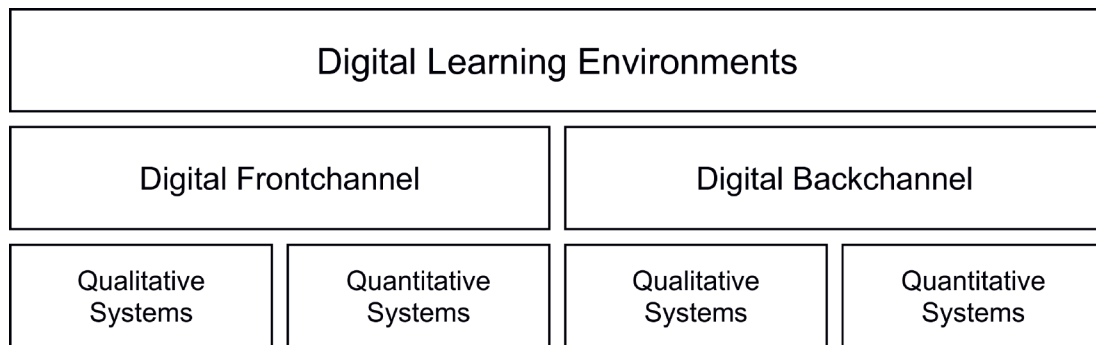


Figure 3.1: The classification of digital learning environments by the type of usage (translated and adjusted version of [Ebn+14]).

3.1 Classification of Digital Learning Environments

In 2014, [Ebn+14] proposed a classification of *Audience Response Systems* (i.e., *digital learning environments*) that allows different forms of systems to be distinguished. Therefore, the authors divide between *digital frontchannel* and *digital backchannel systems*, whereby each of them is distinguished between *qualitative* (i.e., open-ended feedback such as by input of freetext is supported) and *quantitative* (i.e., closed feedback by choosing from predefined choices is supported) systems (cf. figure 3.1). The term *system* can be transferred to the *individual functions of systems*, as today's systems usually combine a range of different functionalities. *Frontchannel* can be understood as functionality executed actively during the ongoing lecture and therefore requires a certain break to be added, e.g., to allow students to think about the possible correct answer when solving a task. In contrast, *backchannel* functions run in the background of the lecture and can be used by students without adding a break during the lecture.

While this classification is quite easy to understand and can also be applied to individual functions, several problems occur: Even though functions have a preferred type of application (i.e., *frontchannel* vs. *backchannel*), it strongly depends on the individual lecturer and his/her teaching strategy how a function is integrated into a lecture. For example, *learning questions*, which are commonly used in the *frontchannel* of the lecture, can also be executed in the background. Similarly, a traditional *backchannel* functionality such as *question & answer* can be implemented actively in the *frontchannel*, when its results should be discussed. Furthermore, some functions may also be classified as both *qualitative* and *quantitative*, e.g., a *single-choice question* (i.e., select one of several predefined options) that asks for students' courses of study can optionally allow the input of an open-ended answer.

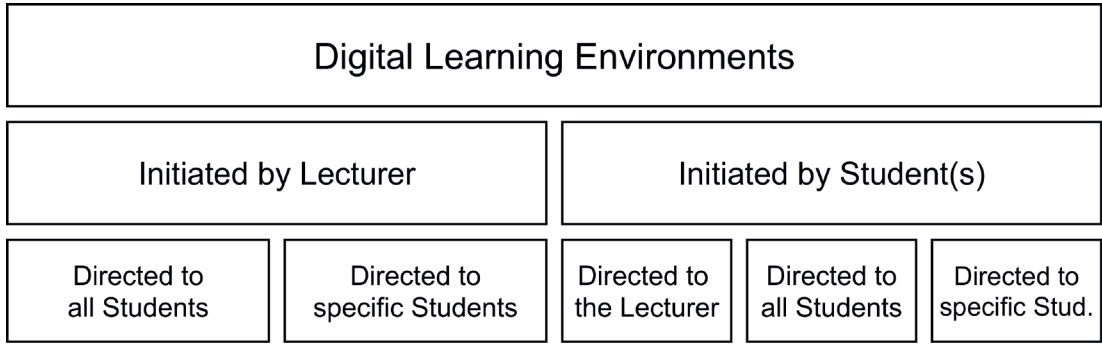


Figure 3.2: The classification of digital learning environments by the type of interaction.

Another classification is motivated by [Rob11], who presents a learning environment that allows various classroom interactions: *Teacher to individual student(s)*, *teacher to all students*, *student to teacher* and *student to student*. As the type of interaction is significant to identify the type of application, it can be used to classify systems and their functions. For instance, [Fli17] categorizes systems regarding their direction of interaction for feedback and distinguishes between *feedback from the learner to the auditorium*, *feedback from the learner to the lecturer*, and *feedback from the lecturer to the auditorium*.

We propose a classification that builds on top of these interaction types and provides information on both the initiator(s) and the receiver(s) of an interaction. The initiator can either be a lecturer or one or more student(s). Furthermore, we distinguish whether the receiver is the lecturer or if the receivers are all or specific students. Our proposed classification is summarized in figure 3.2.

Contrary to the classification presented first, the type of interaction is rather meaningful when selecting a suitable functionality that allows a specific interaction to be performed. However, the type of application can be used as an addition for lecturers to identify *best practices*. Thus, in the following, we will classify functions according to both their type of interaction and preferred way of use.

3.2 The Functional Scope of Digital Learning Environments

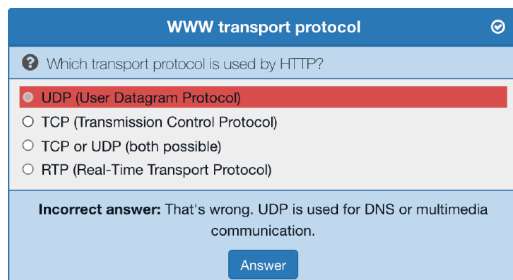
A variety of *digital learning environments* exist, which are listed in different overviews, e.g., [Ber15; Fli17; HPE14; Har16; Kub+19; Kun+13; MMN18]. However, as motivated in the previous section, their functional scope varies strongly depending on the intended type of application. Thus, this section gives an overview of common functionalities and their desired purpose of application, which will serve as the foundation of this thesis.

3.2.1 Interaction Initiated by the Lecturer

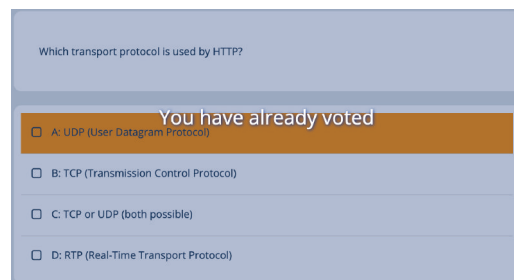
The most popular function to integrate and activate students during the lecture is presented by *polling* functionalities, which allow *all students* to answer previously prepared or spontaneously created questions. While early hardware-based systems were referred to as *Clickers*, web-based systems that allow students to answer using their personal mobile devices are known as *Audience Response Systems*, *Classroom Response Systems*¹, or *Student Response Systems*. Even though these terms often describe different functions, they are primarily understood as traditional *response* (i.e., *polling*) functionality.

Contrary to hardware-based systems, on which buttons are fixed, and therefore only simple question types (e.g., *single- or multiple-choice questions*) can be realized, web-based systems allow the implementation of advanced question types that require students to apply different cognitive skills in order to stimulate different learning processes. The range of types is listed by several studies, e.g., [Har16; Cer+19; MB19]. A summary is given by [MB19] that differentiates the following question types: *Choice*, *open answer*, *region*, *sketch*, *fill-in-the-blank*, *scale*, *order*, *sort*, *graph*, *text highlight* and *match*. In any case, the implementation of novel question types is still a subject of research.

In addition to their type, questions can be distinguished by their settings, which are determined according to both the intended purpose of application and the demand of students. For example, for *single-choice questions*, correct answers or hints can be defined to inform students in one or more feedback loops about the correctness of their given answer. However, in other application scenarios, the correct answer should not be revealed but discussed with or between students. An example of two questions with the same question type but different application purposes is displayed in figure 3.3.



(a) A question displaying feedback on the correctness of the given answer.



(b) A question without feedback on the correctness of the given answer.

Figure 3.3: The comparison of two questions with the same type but different purpose of application (screenshots taken on <https://amcs.website/> and <https://arsnova.eu/>).

¹ The term *Classroom Response Systems* might be the most appropriate term in educational settings [Bru09].

According to the classification presented in figure 3.1, the *polling* function is to be classified in the *frontchannel* because students require time to think about their answers. Furthermore, it is either *qualitative* or *quantitative*, depending on the input of the selected question type. In general, several application purposes exist for this functionality (e.g., as presented by [Qui16a; Kap+14]). In the following, we list several examples of used practices. Overlaps, as well as variations of individual practices, might occur:

- In *Just-in-Time Teaching*, *preparation questions* are posted in advance of the upcoming lecture to tailor its content based on the results. These questions can either ask for previous knowledge or assess it by means of *interactive learning questions* (see below). The correctness of an answer is either be displayed or discussed as a part of the lecture.
- *Interactive Learning Questions* are used to activate students at the beginning, the middle, or the end of the lecture [KK11]. These questions allow students to check their gained knowledge and provide feedback to the lecturer on whether certain topics have been understood or need to be repeated. They can either hide or reveal the correct answer, depending on the purpose of the application.
- In *Peer Instruction*, *conceptual questions* are posed during the ongoing lecture (i.e., *ConcepTest*). As, depending on the results, the possible correct answer should be discussed between peers (i.e., *Peer Discussion*), it shall not be revealed after answering. Moreover, the questions have to be phrased in a way that they are neither too easy nor too difficult, which allows a discussion to be executed.
- In *Self-Study Phases*, *interactive learning questions* should either reveal their correct answer or provide several feedback loops to support the students in finding and understanding the correct answer. *Self-Study Phases* can be used in advance, during, as well as in the post-processing of the lecture, e.g., to prepare for the upcoming exam.

The *polling* functionality has been evaluated in numerous use cases with varying purposes of application (cf. [NE18]) and has proven its added value to promote students' learning (cf. [Hun17; HAB16; KL09]). Nevertheless, in addition to the system used, the type of implementation is important. Although best practices can be modified, the students have to be taken into account. For example, it is essential to give students enough time to complete learning questions during the lecture and discuss the results in more detail. A variety of further tips exist that should be considered (cf. [Rob00]).

In addition to the *polling* functionality, students can be provided with *messages* that either directly provide or link further information on a certain topic. Moreover, concrete details about the lecture or the chair can be given, e.g., if theses are offered in the current subject area. Such messages are sent in the background of the ongoing lecture.

As the interaction from the lecturer to the students is an essential part of *digital learning environments*, the first functional requirement is formulated as follows (the symbolism was introduced in the motivation of this chapter).

FR1) Support the interaction from the lecturer to all students

- ▣ Either qualitative or quantitative types of questions are supported.
- Both qualitative and quantitative types of questions are supported.
- Allow sending messages to students (optional).

The functions presented above can also be limited to *specific groups of students* in order to target the diversity occurring in current classrooms (cf. [HK19]). As students with different courses of study (and thus with varying prior knowledge) often attend the same lecture, certain questions or messages may be hidden or their formulation adjusted. Therefore, *Pinnion* allows defining *prerequisites*² to ensure that certain events (i.e., one or more specific answers) occur before displaying a question. Moreover, [Kap+14] presents an approach to support specific groups of students by different *prompts* (i.e., targeted messages). In addition to messages that provide further material, *cognitive prompts*³ can be defined to give individual feedback during the lecture, and *metacognitive prompts*⁴ to support students in reaching their personal learning goals.

Limiting functions to specific user groups can master challenges occurring in teaching scenarios. Consequently, the following functional requirement is defined.

FR2) Support the interaction from the lecturer to specific students

- ▣ Support limiting of questions or messages to individual students.
- Support limiting of questions or messages to specific groups of students.

² The related article can be found on <https://www.pinnion.com/help-and-resources/main-help/survey-logic-question-prerequisites/> – last successful access on October 8, 2021

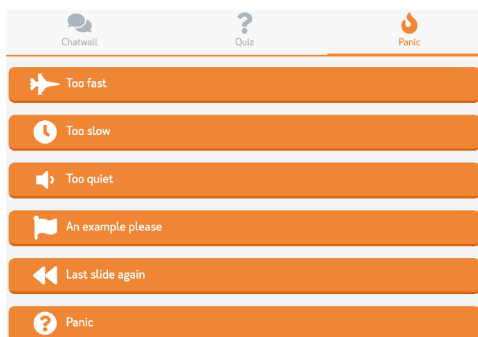
³ For example, students who answered a previous question incorrectly can receive a prompt when the topic is repeated.

⁴ For example, students who just want to pass the exam can receive a prompt when a relevant topic is explained.

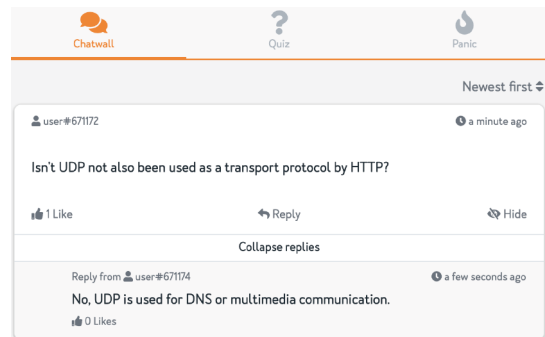
3.2.2 Interaction Initiated by the Student(s)

A common problem in traditional lectures occurs when students either have open questions or cannot follow up. Therefore, *backchannel* functions are investigated, allowing students to initiate their own interactions in *qualitative* and/or *quantitative* formats.

The most simple type of feedback initiated by the students is provided by *quantitative Instant Feedback*, which allows students to select from different feedback dimensions. For example, *Tweedback*⁵ offers the options “Too fast,” “Too slow,” “Too quiet,” “An example please,” “Last slide again,” and “Panic” in order to allow students to respond as soon as something went wrong (an example is depicted in figure 3.4a). While *Instant Feedback* is easy to process and can provide useful hints, different challenges arise, which are summarized by [CDV15]: The lecturer cannot check the screen every minute and thus does not receive the feedback immediately, which especially holds for “mobile lecturers” that wander in the class. Another challenge is the visual overload because the lecturer is split between the current slide, the students, as well as the organization of the lecture. Possible solutions are provided by smart gears (i.e., wearables) such as smart-watches (cf. [CDV15; UE16] or smart-glasses (cf. [Wol+17]). However, this creates challenges on its own, as the lecturer has to cope with the distraction introduced by such devices.



(a) The *Panic* function (i.e., *Instant Feedback*) to choose between different options.



(b) The *Chatwall* (i.e., *Question & Answer*) to ask and discuss open questions.

Figure 3.4: The comparison of two *backchannel* functions that allow students to initiate interactions (screenshots taken on <https://tweedback.de/?l=en>).

Another problem occurs if students have specific questions that cannot be expressed by the provided *Instant Feedback* options. Therefore, *Lecturer Questions* are used to allow students to ask their own questions to the lecturer that can be answered within the next break or during the introduction of the next lecture.

⁵ <https://tweedback.de/?l=en> – last successful access on October 8, 2021

The following functional requirement is defined to allow the interaction from the students to the lecturer:

FR3) Support the interaction from the students to the lecturer

- ▣ Either qualitative or quantitative feedback is supported.
- Both qualitative and quantitative feedback is supported.

As there can be a vast number of questions, *Lecturer Questions* are often replaced by *Question & Answer* functionalities that allow interactions between students to be initiated. In order to cope with a large number of questions, *rating* options give students the ability to highlight useful questions. Moreover, a *reply* function allows them to help each other by answering questions as soon as they arise, without interrupting the ongoing lecture. Finally, *down-votes*, *flags* to mark spam, or *moderated modes* (i.e., a moderator unlocks all promising questions) are used to avoid potential spam.

While the previously described function targets interactions to be directed to all participants, several systems allow initiating discussions between specific students. Therefore, they propose the extension of the *Question & Answer* functionality by *private messages* that allow moving ongoing discussions into separate spaces. This will not only avoid spam in the public discussion but furthermore also strengthen the collaboration between the students involved.

Note on Interactions between Students

The collaboration between students will be investigated in more detail in subsection 3.4.5. Thus, no requirement is defined at this point.

3.3 Analysis of Existing Systems

In this section, the overall goal is to get an understanding of open research issues occurring in the domain of *digital learning environments*. Therefore, an unstructured literature review is described, which examines 50 systems (that provide at least a free plan) and their possibilities for implementing the varying teaching strategies of lecturers. In addition, their individual limitations are identified, which will contribute to define the research gap that is targeted by this thesis. The results of this section are partly published within [Kub+19].

3.3.1 Methodology: An Unstructured Literature Review

In literature, there exist numerous overviews that list and classify a variety of *digital learning environments* based on a set of different criteria, e.g., [Ber15; Fli17; HPE14; Har16; Kun+13; MMN18; Vet+14]. However, these overviews are often quite limited in their investigated selection criteria and are targeted to specific use cases. Moreover, they include fully commercial systems, whose usage is not affordable by the majority of lecturers. Therefore, these overviews cannot be used to find a suitable system for a predefined scenario without great effort.

Thus, our target was to create a comprehensive overview of systems, which are either free-to-use or provide at least a free plan, in order to support lecturers in choosing an appropriate system to realize their desired teaching scenarios. Moreover, this overview should help to identify limitations and thus detect potential research topics. In addition to systems found in the overviews listed above, additional research was conducted using common search engines (e.g., Google Scholar⁶) and open repositories (e.g., Github⁷). As a result, a list of 50 systems, each of which is either freely usable or provides a free plan, could be identified.

In the next step, a large variety of selection criteria from both proprietary and didactic view was defined. In order to provide a comprehensive processing and easy-to-understand filter method, the metaphor of index cards was used. The result of the index card for choosing an appropriate *digital learning environment* is depicted in figure 3.5a and combines a variety of selection criteria. However, this approach is easy to extend, as visualized by the ellipses.

Afterward, each system was categorized by this index card. An excerpt of the resulting tabular view is displayed in figure 3.5b (the complete list can be found in [Kub+19]). In order to simplify the selection process even further, a web-based tool was created that allows reducing the list of systems by selecting an arbitrary amount of classification criteria. Moreover, it offers the advantage of updating the list of systems regularly. The tool can be accessed on <https://www.rn.inf.tu-dresden.de/arselector/>⁸.

⁶ <https://scholar.google.com/> – last successful access on October 8, 2021

⁷ <https://github.com/> – last successful access on October 8, 2021

⁸ This URL was last successfully accessed on October 8, 2021.

Proprietary View					Didactic View				
Costs					Content / Type	Web Application	Mobile Application	Presentation Application	Other
Free		Commercial	Commercial with Free Plan		Content Creation				
Open Source					Content Execution				
Language					Content Presentation				
Chinese	Spanish	English	...	German	Content Evaluation				
Server Setup					Student		Lecturer		
Public Server		Extension of External System		Own Server	Functional Scope				
Type / Format	GIFT	IMS QTI	Moodle XML	Other	Qualitative Front Channel	Quantitative Front Channel	Qualitative Back Channel	Quantitative Back Channel	
Import					Scenario				
Export					School		University		Event

(a) An index card to select an appropriate *digital learning environment*.

	Costs			Language			Server Setup		Import		Export		Content Creation		Content Execution		Result Presentation		Evaluation		Functional Scope		Scenario														
	Free	Commercial	Commercial with Free Plan	Open Source	Chinese	Spanish	English	German	Public Server	Extension of external System	Own Server	GIFT	IMS QTI	Moodle XML	Other	GIFT	IMS QTI	Moodle XML	Other	Web Application	Mobile Application	Presentation Software	Other	Web Application	Mobile Application	Presentation Software	Other	Student	Lecturer	Qualitative Front Channel	Quantitative Front Channel	Qualitative Back Channel	Quantitative Back Channel	School	University	Event	
AMCS																																					
AnswerGarden																																					
ARSenic																																					
ARSnova.app																																					
ARSnova.click																																					
ARSServer																																					
AskVote																																					
AuResS																																					
Backchannel																																					
Backstage 2.0																																					
ClassClicker																																					
ClassQuestion																																					
Cliqr																																					
Crowdsignal																																					
DirectPoll																																					
Feedbackr																																					
Formative																																					
FreeMobilePolls																																					

(b) An excerpt from the tabular summary of applying the index card to existing systems.

Figure 3.5: The filter method used as well as an excerpt of the resulting classification of 50 systems.

3.3.2 Findings

The resulting classification table, whose excerpt is displayed in figure 3.5b, reveals that the majority of systems offer a *web-based solution*, which is commonly made accessible over a public server. One reason for using web-based solutions is that the barriers to join the system's functionality should be low in order to allow almost any student, independent of the device used, to participate. Another reason is the easier development, as one code base is sufficient to deliver the application to nearly any device. Finally, *digital learning environments* only partially require access to native device functionalities such as push notifications, and even these are supported by the majority of modern web browsers. A few well-known systems, such as Kahoot⁹ or Polleverywhere¹⁰, offer *native apps* anyway to allow students to choose how they want to use the respective system. The same applies to extensions to integrate results into presentations in real-time.

Regarding the *range of functions*, it is noticeable that most of the systems provide frontchannel functionalities (i.e., functions that are used actively during the lecture and therefore require a certain break). Several other systems provide backchannel functions – however, these are usually offered in combination with frontchannel functions. Moreover, even though the supported functionality of these systems is quite similar by mainly providing questions to be answered, import and export functions are only partially supported, and even if they are, they are limited to custom formats. The provision of uniform formats to exchange questions between different systems is rarely supported.

The *options to present results* are primarily focused on real-time evaluations of the results during the ongoing lecture. Even though the results are usually visible after the lecture, more in-depth analyses (e.g., linking results of different questions to analyze specific student groups) are rarely provided [Bra+18]. Furthermore, students themselves often cannot access their given answers after the lecture, as the participation was conducted anonymously. In the university setting, it is therefore quite useful to combine the participation with the creation of an account – in [Bra+19], for example, an approach was presented, in which students submit their confidence during answering that can later be used to repeat promising questions. In this way, the *digital learning environment* cannot only be used to increase the interaction during the lecture but also to prepare for an upcoming exam.

⁹ <https://kahoot.com/> – last successful access on October 8, 2021

¹⁰ <https://polleverywhere.com/> – last successful access on October 8, 2021

3.3.3 Problem Statement

In addition to providing support for lecturers and analyzing the systems' features, another goal was to determine the limitations of systems when targeting to implement different teaching strategies. It could be recognized that systems often rely on a single supported didactic concept and therefore have predefined limitations. For example, [Rei+12] or [Qui16b] describe approaches that aim for cooperation by supporting *Peer Instruction* – therefore, questions can only be answered once in a row and do not reveal their correct answer(s). In comparison, [Kap+14] describes an approach to support students in regulating their learning using principles of *Self-Regulated Learning*, resulting in questions that can be answered twice and feedback provided after selecting certain choices. Only a few approaches allow adapting the functional scope and settings accordingly, e.g., [Kub+17].

Although different systems can be used to implement different didactic concepts, most systems have to be classified as purely technical solutions. While functional restrictions can be recognized, they are rarely described using the underlying didactic principles or are not even made clear as those. This furthermore results in lecturers being poorly supported in the didactically correct implementation of the system. Instead, they are often left alone with the system or only supported by best practice guides.

In order to investigate further issues, in [Shm18; Shm19], the components of *digital learning environments* to support different well-known didactic concepts were studied. Therefore, 20 out of the list of 50 systems (cf. [Kub+19]) were randomly chosen and a total of six common components could be identified to support five didactic concepts. It was particularly noticeable that, although collaboration is a key component of several didactic concepts, it is rarely supported by *digital learning environments*. This especially holds for the functionality to divide students into meaningful groups and provide them with opportunities to interact in these groups. Rather, the collaboration in groups is often limited to offline scenarios or requires lecturers to form groups of students manually.

As motivated earlier, systems rarely provide the opportunity to adapt either the range of functions or individual functions to support specific didactic concepts. Consequently, they also do not suggest a proper functional scope adaptively. However, we strongly believe that adaptation can be an essential requirement to tackle both problems, the missing support of the lecturer as well as the limited means of collaboration. By proposing collaborative components adaptively, the lecturer can be supported in using best practice didactics. Furthermore, the adaptation of the functional scope and individual

functionalities is crucial to support the diversity of didactic concepts. In summary, the following problem definitions and objective were defined, as listed in section 1.2:

- Lecturers are not supported by *digital learning environments* when implementing different didactic concepts.
 - Although collaboration is an essential component of different didactic concepts, advanced collaborative functions such as goal-oriented group formations and interactions are not supported by these systems.
- ⇒ The means of adaptation will help to target both the functional scope and individual functionalities, as well as to propose and integrate useful components.

A detailed investigation of all three research directions in the context of *digital learning environments* will be described in the following section.

3.4 Related Work

While the functional scope and limitations of existing *digital learning environments* were investigated in the previous sections, this section will explore the current state of the art of *lecturer support*, *collaboration* and *means of adaptation*. Therefore, the application of a two-step structured literature review is introduced and its results are presented: First, the problem definitions should be confirmed before related approaches to solve them will be investigated. Second, for each group of related work, a requirement is defined that will be used in the next section to identify the research gap targeted by this thesis.

The chosen methodology is based on the methods used by [CI14; Sch+17].

3.4.1 Methodology: A Two-Step Structured Literature Review

In the first step, the didactic strategies that are supported by *digital learning environments* were investigated. Therefore, a search query was defined combining both synonyms for *audience response systems* (i.e., *digital learning environments*) and *didactic concepts*. The relevance of synonyms was determined by the number of search results in Google Scholar¹¹ – irrelevant synonyms were omitted. In total, four synonyms for the term *audience response system* and 17 synonyms for *didactic concept* were defined. The individual

¹¹ <https://scholar.google.com/> – last successful access on October 8, 2021

synonyms were concatenated using *OR* and both groups were connected using *AND*. The following search query was created:

(bring your own device OR audience response system OR student response system OR classroom response system OR personal response system)

AND

(didactic concept OR didactic strategy OR didactic method OR didactical concept OR educational concept OR educational strategy OR educational scenario OR educational method OR teaching concept OR teaching strategy OR teaching scenario OR teaching method OR pedagogic strategy OR pedagogic method OR pedagogical concept OR pedagogical strategy OR pedagogical scenario OR pedagogical method)

As the primary goal was to search for *peer-reviewed* papers, the search was performed using the five most frequently used academic databases in the field of *Tech-Enhanced Learning (TEL)*: ACM¹², IEEE Xplore¹³, ScienceDirect¹⁴, SpringerLink¹⁵ and Wiley¹⁶. In addition, the academic database ERIC¹⁷ was added to find papers from the field of psychology, too. In order to ensure that the search is complete and no essential paper has been missed, the academic search engines Scopus¹⁸ and Web of Science¹⁹ were also added to the search – however, the search was limited to *title*, *abstract* and *keywords* to only find highly relevant papers and avoid less relevant duplicates already found. Each paper (if accessible) was manually retrieved and checked for its relevance. If a paper was found to be partially relevant, it was added to a list sorted by its year, and a short description, the origin of the paper and the number of citations were added. In order to also cover the area of relevant *gray literature*, a manual search in *Google Scholar* was performed, in which the titles were scanned. The search queries had to be split into several sub-queries (similarly to *ScienceDirect*), as *Google Scholar* has a character limit of 256 characters. The results of each search term were limited to the first 200 search results. A summary of the first step of the structured literature review and its number of individual results is depicted in figure 3.6.

¹² <https://dl.acm.org/> – last successful access on October 8, 2021

¹³ <https://ieeexplore.ieee.org/Xplore/home.jsp> – last successful access on October 8, 2021

¹⁴ <https://sciencedirect.com/> – last successful access on October 8, 2021

¹⁵ <https://link.springer.com/> – last successful access on October 8, 2021

¹⁶ <https://onlinelibrary.wiley.com/> – last successful access on October 8, 2021

¹⁷ <https://eric.ed.gov/> – last successful access on October 8, 2021

¹⁸ <https://scopus.com/> – last successful access on October 8, 2021

¹⁹ <https://apps.webofknowledge.com/> – last successful access on October 8, 2021

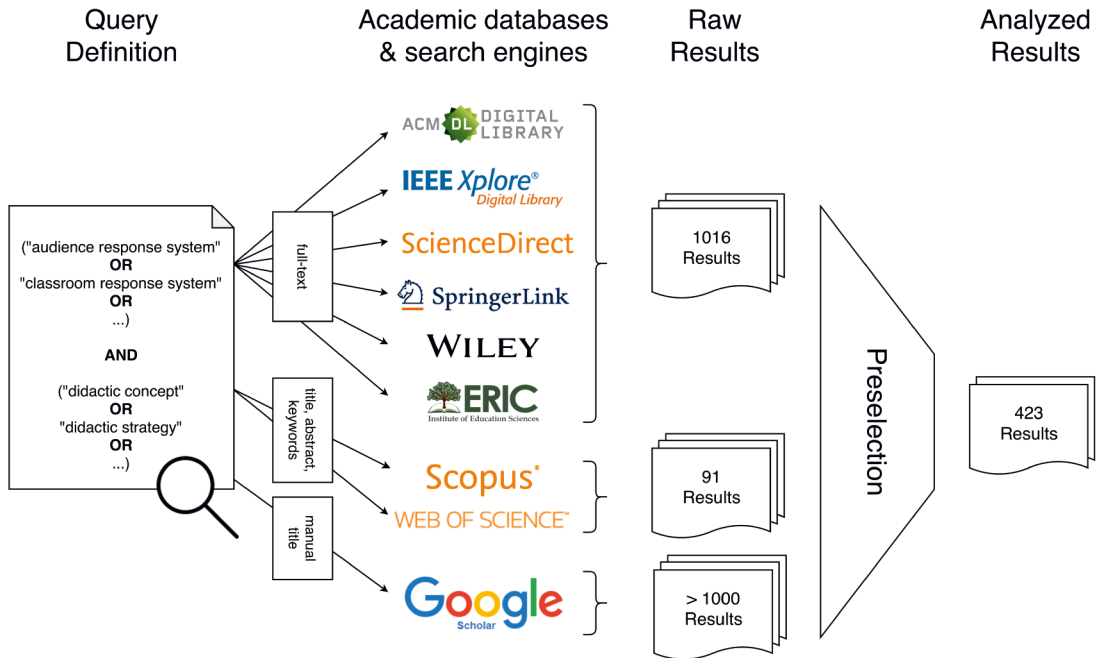


Figure 3.6: A summary of the first search query to confirm the problem statements that were described in subsection 3.3.3 and make statements about didactic strategies used.

Results of Query 1

In total, 423 papers were analyzed to confirm the problem definition and make statements about didactic strategies used. The results are presented in more detail in subsection 3.4.2.

In the second step, the current state of the art of the problem statements (i.e., missing *lecturer support* and *collaboration*) and the objective (i.e., usage of the *means of adaptation*), which were identified in subsection 3.3.3, should be explored. Therefore, relevant literature found in the first query was extended by a second query. In addition to the search term of *audience response systems*, three search terms (with their according synonyms) were identified that were targeted to find collaborative, adaptable *digital learning environments*: *group formation*, *collaborative learning* and *customizable*. While these terms consider both (*group*) *collaboration* and the *means of adaption*, we decided not to add separate search terms for the *lecturer support*, as this is strongly related to the search terms already presented as well as the results of the first query. The synonyms of the previously mentioned three search terms were concatenated using *OR* and connected with the synonyms of *audience response systems* (defined in the first query) by *AND*. This results in the following search query:

(bring your own device **OR** audience response system **OR** student response system **OR** classroom response system **OR** personal response system)

AND

(group formation **OR** group design **OR** group composition **OR** group organization **OR** team formation **OR** team composition **OR** team organization **OR** collaborative learning **OR** cooperative learning **OR** social learning **OR** group learning **OR** team learning **OR** customizable **OR** customize **OR** personalize **OR** configurable **OR** adaptable **OR** adjustable)

The search was performed similarly to the first query with minor adjustments: First, we limited it to research conducted since 2009, as web-based *digital learning environments* started to grow around this year. Second, we used the *title, abstract* and *keywords* search for each database, if possible²⁰, as this provided the best results. Finally, we added an iterative search to the *preselection* stage in order to find the most recent publications of different approaches. A summary of the second step of the structured literature review and its number of individual results is depicted in figure 3.7.

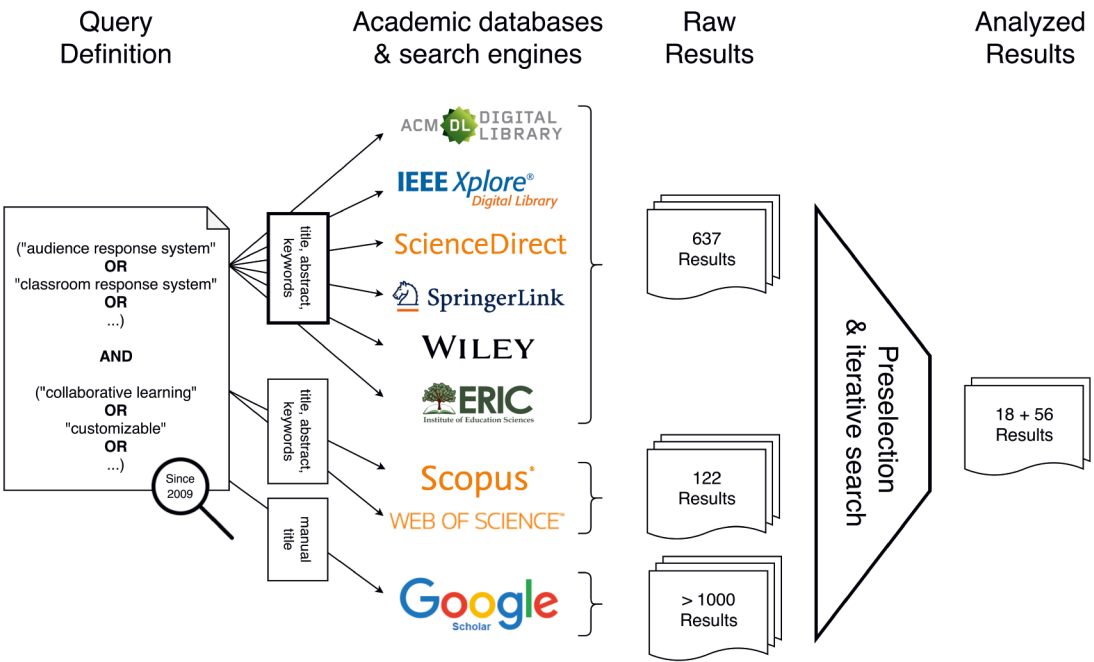


Figure 3.7: A summary of the second search query to explore the state of the art of the problem statements and objective (bold lines indicate changes made in the search process).

²⁰ In SpringerLink, a full-text search had to be conducted, and in both ScienceDirect and Google Scholar, the search had to be split into several sub-queries.

Results of Query 2

In addition to 19 papers identified as highly relevant in the first query, 55 new papers were found, resulting in a total of 74 papers.

3.4.2 Didactic Strategies and Confirmation of the Problem Statement

With the increasing availability of wireless Internet in universities and the rising number of mobile devices among students, the importance of *digital learning environments* is growing continuously. This can be seen, for example, in the number of papers per year from our first query. While 31 papers were found for 2014, there were 43 for 2015 and 66 for 2016. This number is similar for 2017 with 52 and 2018 with 57 papers, which demonstrates the importance of the research area and is certainly continuing to grow as a result of the CoViD-19 pandemic.

In order to evaluate this large number of papers, we divided them into six groups (the total number of papers per group is listed in parenthesis), as can be seen in figure 3.8:

- *Advantages by clicker*: the application of hardware-based *clickers* is described and evaluated (75),
- *Advantages by ARS*: the application of web-based *audience response systems* (i.e., *digital learning environments*) is described and evaluated (31),
- *Didactic terms*: the application of didactic strategies is presented (77),
- *Course design*: the design of courses, in which *digital learning environments* are involved, is described (54),
- *Applications*: novel applications (i.e., prototypes of *digital learning environments*) are presented (47),
- *Strongly related*: *digital learning environments* that support collaboration, offer means of adaptation or support the lecturer are described (19).

120 of the 423 papers were classified as “other papers” and thus are not relevant for this thesis, resulting in a total of 303 classified papers. Based on this classification, the following conclusions can be drawn:

- Hardware-based clickers have been increasingly replaced by web-based systems.

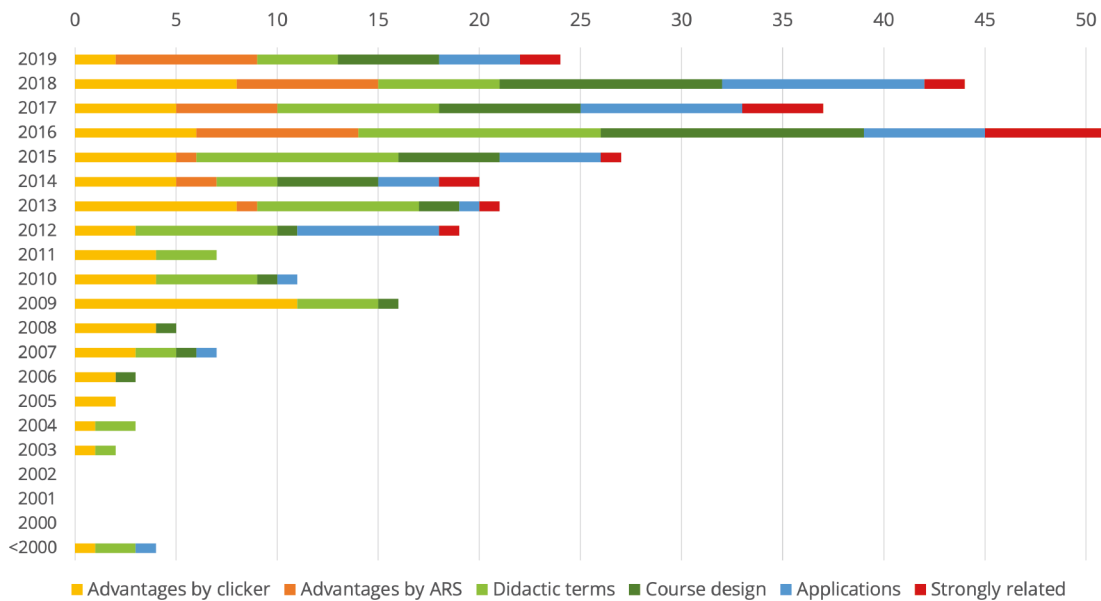


Figure 3.8: A summary of research fields that were investigated by the state of the art.

- The support of didactic strategies by *digital learning environments* was continuously investigated.
- Since 2012, the creation of prototypes to provide novel functionalities has been increasingly investigated.
- Since 2014, lectures have been more frequently being restructured, with *digital learning environments* having a leading role.

As the literature review was mainly focused on the use of *digital learning environments* to implement or support different didactic strategies and was targeted to confirm the problem statements, the papers of the groups *didactic terms*, *course design* and *applications* were investigated further. The results are summarized in the following statements:

- *Digital learning environments* can support many different teaching methods, including both traditional and inverted classroom scenarios, and have been proven, for instance, to stimulate students' *critical thinking* [Liu+17; Swa17] or to promote *active learning* [Álv+17; Gon18; El 17; Wol+15].
- With *clicker-integrated instruction* [CCC16], *question-driven instruction* [LKF15] and *peer instruction* [Arn+13; Kun+12; Len+15; Rei+12; SM15; Sch11; Siv+19], several teaching methods exist, in which the use of *digital learning environments* represents a crucial role.

- Moreover, *digital learning environments* are often used as a starting point of other collaborative teaching strategies, e.g., the results of a learning question can be used to initiate a *peer discussion* [LSL17; Lew+16; Lin+14; Mil+15] in *peer instruction*.
- While *digital learning environments* are used to initiate those collaborative strategies, the collaboration itself is mainly executed offline – very few approaches exist that propose to transfer several collaborative activities into virtual space, e.g., [Har16]. However, the multitude of collaborative scenarios, e.g., *Jigsaw* [Wol+15] or *Think-Pair-Share* [Wol+15], are, to the best of our knowledge, not yet supported by *digital learning environments*.
- In addition, lecturers have to adapt to the system's limitations (e.g., whether a possible correct answer is revealed during answering or the number of repetitions students got to answer a question [Kub+17]) when implementing a specific teaching method. Sometimes, lecturers are supported by best practice guides – however, those are tailored to the currently used system. The creation of own sequences, on the other hand, is very rarely supported.
- In summary, lecturers have to assess the systems and their supported range of functions themselves and have to decide, whether these are capable of implementing their desired scenario. Furthermore, they are left alone during the actual implementation – they do not receive feedback on possible useful extensions of their chosen functions.

The investigations have confirmed our problem statements defined in subsection 3.3.3: *Digital learning environments* do not support lecturers in implementing their individual teaching scenarios. This especially holds for the definition of customized scenarios as well as for the support of collaborative activities. We strongly believe that the means of adaptation can solve these problems. Consequently, in the following three sections, we will investigate the state of the literature for *lecturer support*, *collaboration* and *means of adaptation* in *digital learning environments*.

Note on the Sections' Order

As the *means of adaptation* might be useful for both *lecturer support* and *collaboration*, it will be investigated between them.

3.4.3 Current State of Lecturer Support

There were identified three groups of functions to support the lecturer in *digital learning environments*: *Predefined scenarios*, *scenario proposals* and *functional proposals*. For each group, a requirement is defined and the current state of technology will be presented.

FR4) Provide *predefined scenarios* to support the lecturer in finding best practices

- ▣ A description of functions is provided to decide which of them is appropriate for a specific scenario.
- A list of available scenarios is displayed.
- Each scenario unlocks different functions, and settings for specific functions.

In [HMB19], the composition of several core components into teaching methods as well as the composition of teaching methods into learning and teaching formats is described (as shown in figure 3.9). For example, for *audience response*, both *input interactions* (i.e., the actual question) and *learning analytics* (i.e., the evaluation) are required. Moreover, *large class teaching* benefits from both *audience response* and *peer review*. This guide can help lecturers to implement different teaching formats.

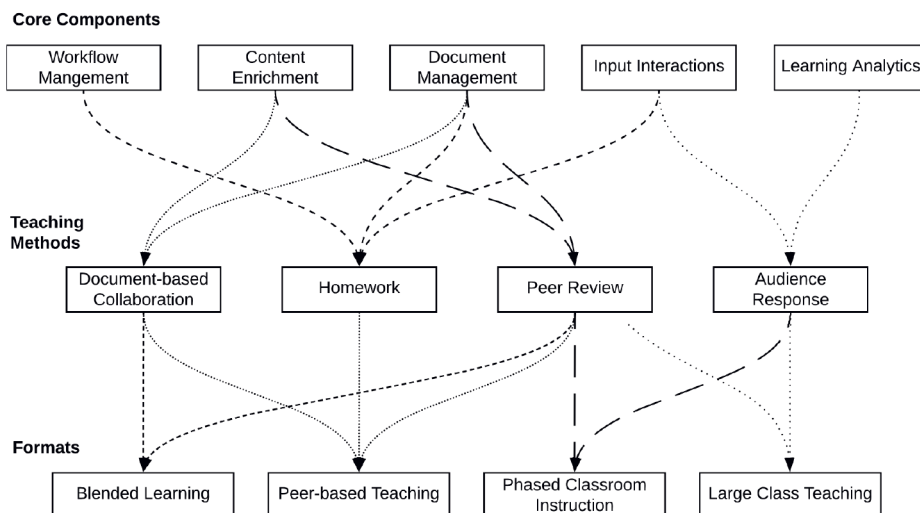


Figure 3.9: The composition of core components into teaching methods that are composed into teaching formats [HMB19].

Another approach is presented by [Qui16b], who integrates an overview of predefined scenarios into a *digital learning environment* from which lecturers can select. Each scenario provides a short description and enables a specific functional scope after selection. Furthermore, it is possible to select own combinations of different functions.

FR5) Make *scenario proposals* to initially support the lecturer

- Provide proposals of appropriate teaching methods, functions or settings.

[Kub+17] describes an approach that supports lecturers in choosing an appropriate range of functions for their specific use cases. Therefore, in a first step, values for different predefined influence factors (e.g., the approximate number of students or the time available during the lecture) have to be input (see figure 3.10a) before the proposal of an appropriate functional scope is made in the second step (see figure 3.10b). Each proposal consists of several sub-functions that can be used to implement different use cases. For each selected use case, an example is displayed. It is important to note that the range of functions can be customized freely. Thus, lecturers do not have to use the suggested functions if they do not want to.

FR6) Make *functional proposals* to improve the lecturer's strategy during execution

- ▣ Functional proposals are made as a post-processing of the lecture.
- Proposals of appropriate teaching methods, functions or settings are made during the ongoing lecture.

In [DG18], a *machine learning*-based methodology is presented, which allows lecturers to constantly improve their teaching. Therefore, it identifies teaching methods, instructional tools, and pedagogies that work best in the lecturer's classroom. This is done by analyzing data on students' results (more specifically, non-experimental on previous scores collected by the university) and making changes with data-driven insights, which is also called *data-driven classroom tuning*.


The method was evaluated in teaching undergraduate mathematics. By applying their algorithm in an ordinary differential equations class, the authors found that clickers (i.e., *Audience Response*) were more effective than traditional handwritten homework – however, online homework that provides immediate feedback was rated as even more effective. When evaluating the methods that were used in a calculus class, active teamwork was found to be more beneficial for students than individual work. The algorithm is integrated into an app, so it can be used without advanced methodological training.

Lecture creation

General informations > **Specific informations** > Function selection > Summary

i The data for the factors are used to suggest a function selection for the lecture. For factors marked with *, a value must be given in order to continue. Move the mouse over the **i** icon to get more informations for the factor. ✕


? *



Student count

250


? *



Beamer/Light conditions

Beamer with average light conditions


? *



Own preparation time for the system

30 min


? *



Free time during the lecture

25 min

? *



Time in semester

Begin of semester

+

Show optional factors

Back
Continue

(a) Input values for different influence factors.

+2 Answer questions

☒ Course questions

☒ Slide questions

☐ Lecture questions

+2 Request interests and personal goals. ⓘ

+1 Quizzes to test knowledge learned. ⓘ

Questions for evaluation of the lecture/course. ⓘ ⓘ

-1 Preparation questions for the lecture. ⓘ

-1 Request the (pre-)knowledge of the students. ⓘ

Example for request of personal informations

Lecture

Aus welchem Grund besuchen Sie die heutige Vorlesung?

Ich möchte das Thema der Vorlesung verstehen.

Ich möchte die Prüfung am Ende des Semesters erfolgreich bestehen.

Ich möchte mich für eine Masterarbeit in diesem Gebiet bewerben.

Ich würde mir gerne nachher für meine persönliche Tätigkeit als Softwareentwickler bewerben.

Beantworten

● ● ● ● ● ● ●

+1 Messages

☒ Metacognitive messages

☒ Cognitive messages

☐ Further links/materials

+1 Depending on the student's preference, strategic information on the preparation / follow-up of the event will be provided. ⓘ

+1 Students who have incorrectly answered a learning question will be notified when the subject matter covered is repeated. ⓘ

Further information will be provided for the current topic. ⓘ

Example of a metacognitive message

AMCS

Vorbereitung auf die nächste Vorlesung

Für Medieninformatiker wird empfohlen folgende Frage mit Antwort durchlesen: <http://nachklausurflow.com/questions/>

null.zibn.com

● ●

(b) The proposal of an appropriate functional scope.

Figure 3.10: A proposal-based help functionality to support lecturers in choosing an appropriate range of functions.

3.4.4 Current State of Adaptation

There could be identified three potential groups of adaptation that *digital learning environments* benefit from: *Customization*, *conditional sequences* and *runtime adaptation*. The first two groups can be summarized as *modeling adaptation*. In the following, the current state of technology for each group will be presented.

FR7) Support *customization* to create own sequences of adaptable functions

- ▣ A function selection is allowed.
- ▣ Functions can be adjusted by simple settings.
- The definition of custom sequences of functions is possible.
- Single functions can be adjusted by different parameters.

The selection of the range of functions is supported by several approaches. For example, in [Fli17], the lecturer can choose between *chatwall* (i.e., question & answer), *quiz* (i.e., audience response) and *panic buttons* (i.e., instant feedback). For each of the functions, hints are provided on how to use them. Furthermore, simple settings can be defined, e.g., the *chatwall* can be moderated. Similar function selections and settings are provided in [FWB13; Har16]. A more advanced approach that allows the definition of custom sequences as well as different parameters is presented by [Sch16] and will be described in the following paragraph.

FR8) Support *conditional sequences* to define different learning paths

- ▣ The manual execution of the defined conditions is possible.
- Conditions (or rules) can be defined that influence the subsequently chosen path within the sequence.

As motivated above, [Sch16] presents an approach that allows defining customized (conditional) teaching scenarios without predefined functions. Therefore, the author subdivides the creation of those in five phases:

- *Blueprint*: A blueprint of the quiz is created, i.e., the elements and behavior of the quiz are defined.
- *Quiz*: An instance of this blueprint is created, e.g., the formulations of questions or choices are added.
- *Interaction*: The instance is executed and students can participate.

- *Result*: The results can be retrieved and evaluated at runtime.
- *Analysis*: A post-processing of the quiz results is performed, e.g., by exporting the results.

Using a newly created generic model, scenarios are defined analogously to board games (with *gaming pieces* and *rules*). In this model, each *scenario* consists of *objects* with *attributes*, and *rules* with *conditions* and *actions*, as visualized in figure 3.11.

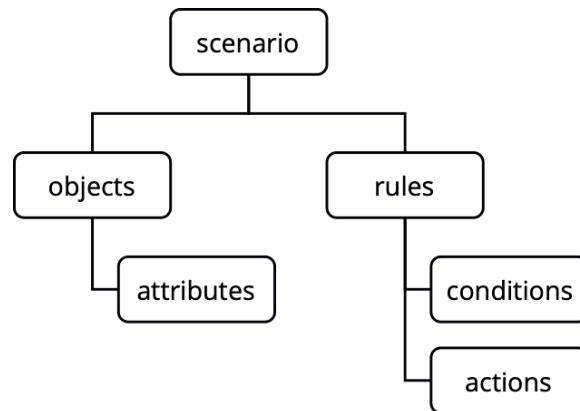


Figure 3.11: A generic model to define customized teaching scenarios that was presented by [Sch16]).

The model was implemented within MobileQuiz2²¹ and evaluated in several lectures. While the model is able to express a variety of different teaching scenarios, the modeling has been proven to be very difficult, so lecturers require the support of a didactic expert in order to create their individual teaching scenarios. Moreover, due to the generics of the model, even small adjustments (e.g., changing the question type) require extensive changes in the application model. Finally, the execution of the scenarios lacked performance, which is caused by the deep-nested objects created by the generic model.

FR9) Support *runtime adaptation* to adapt the scenario if the results indicate it

- The approach should allow for changes in the defined sequence during its execution, e.g., if the majority of students answered a question incorrectly, a *peer discussion* could be proposed to be integrated.

Although we strongly believe that the addition of runtime adaptation may be beneficial for *digital learning environments*, it was, to the best of our knowledge, not yet investigated.

²¹ <http://www.mobilequiz.org/core/index/index> – last successful access on October 8, 2021

3.4.5 Current State of Collaboration

Three groups of collaborative functions could be identified for *digital learning environments*: *Group formation*, *question & answer* and *group collaboration*. In the following, the current state of technology will be presented for each group.

FR10) Allow *group formations* in order to build groups of students

- ▣ The group size or count can be input.
- ▣ A random assignment of groups is done.
- A build schema (random vs. algorithmic) can be selected.
- Different roles within these groups can be defined.

[Riv19] presents a random *group creator* in which a group size can be input. However, all students have to be known, as the collaboration takes place offline – only the question is displayed within the online environment. Moreover, [SDG19] describes the *formation of groups* as well as the *assignment of questions to specific students or groups of students*. Students in the same working group can communicate with each other through messages or postings. However, more advanced collaboration within these groups, i.e., assigning different roles or voting for a common group answer, is not supported.

FR11) Add a *question & answer* option to allow students posting their questions

- ▣ Allow the posting of questions and answers.
- Allow the rating of questions.
- Sort questions and answers by their importance.
- Add favorites of questions and answers.

[Jir+15] describes a traditional *question & answer* functionality with a *voting* opportunity that allows for the selection of an emotion during the posting in order to track the lecture morale, as well as to send *private messages* to other students. Furthermore, [Rod+16] introduces the *creation of multiple-choice questions* as a student, which provides an addition to the common *question & answer* functionality. A more advanced question & answer approach is presented by [BP17] and adds both an *audience response* and *instant feedback* function to it. In addition to answering questions, students can create their own posts with different types (e.g., *question*, *answer*, *comment*, *too fast*, or *too slow*) and assign them to specific positions in the slides to highlight the cause of the problem or question, as visualized in figure 3.12. These posts can be answered and rated by other students, and discussions can be moved into private chats. There exists a variety of

similar approaches that combine the *question & answer* functionality with other functions [ABO10; Har16; FWB13; Fli17] – however, as they do not provide any function that was not yet presented, they will not be discussed further.

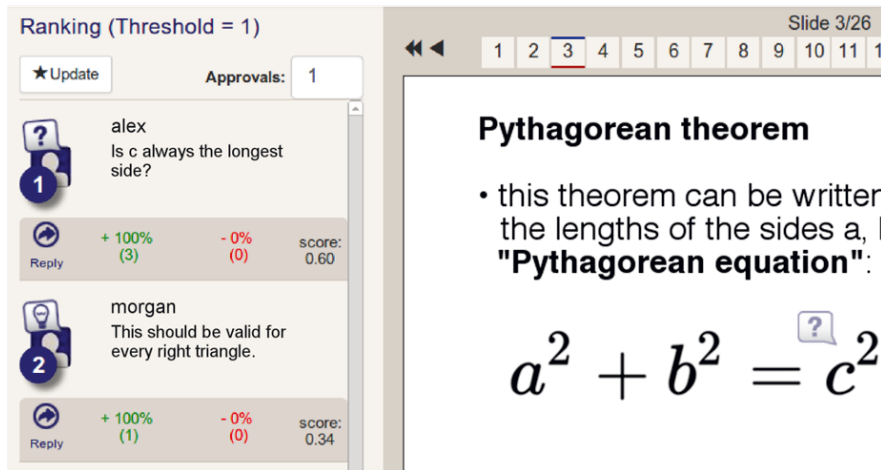


Figure 3.12: The *question & answer* functionality provided by Backstage2 (adjusted screenshot of an example from [BP17]).

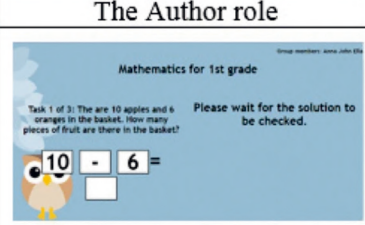
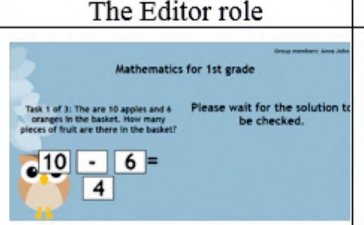
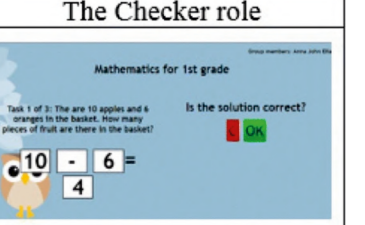
FR12) Enable *group collaboration* to let students interact in groups

- ▣ Individual students can receive help from other students.
- ▣ The interaction between two peers is supported.
- Students can work together in groups.
- Different opportunities based on a specific role within the group are provided.

[CA14] presents an approach of a *digital learning environment* that allows sending a *help request* to another student who has already answered the question. This student is then asked if he/she is willing to help, and if so, he/she moves to the other student in the room and collaborates with him/her offline. After the help procedure, the student who has searched for help can rate this collaboration. A similar approach is presented by [SMB13], in which *help requests* are integrated into a game-based *digital learning environment*. In comparison to the approach presented first, the help takes place by choosing a hint that is displayed to the student who has searched for help.

While the approaches presented above describe the collaboration between individual students, [MH16] describes the implementation of a scalable pedagogical method that refines the Pyramid collaborative learning pattern. The approach allows all students to post a question of interest. In the first level (on the bottom of the pyramid), students that are assigned to this pyramid rate and discuss the questions of others. Only the best-rated

half of the questions proceed to the next level of the pyramid, in which the procedure proceeds. The result of each pyramid is the potentially most interesting question or multiple questions of their students. Each pyramid is assigned to 16 to 20 students with 3 or 4 levels. Another approach is presented by [JB19] and describes collaboration and adaptivity as part of digital learning lessons, which is provided in its “collaborative operation mode.” It is visualized on an example of a *Math Widget* (see figure 3.13), in which collaboration takes place by assigning roles with different tasks to solve by different students: The *Author* translates the problem into a mathematical equation, the *Editor* solves the equation, and finally, the *Checker* verifies the entire solution.

The Author role	The Editor role	The Checker role
		

– Unfortunately, the image was only provided in low resolution.

Figure 3.13: A collaborative group activity with different roles defined for different tasks (adjusted image from [JB19]). The task is formulated as follows: “There are 10 apples and 6 oranges in the basket. How many pieces of fruit are in the basket?”. The *Author* translates the task incorrectly into an equation, which is then solved correctly by the *Editor*. It is expected that the *Checker* would now mark this solution as *incorrect*, as the translation of the task into an equation was not done correctly.

3.5 Summary

In table 3.1, the state of the literature for *lecturer support*, *adaptation* and *collaboration* in *digital learning environments* is summarized. It can be recognized that the related work of *lecturer support* and *collaboration* barely overlap – however, as *collaboration* is an essential component of several teaching strategies, it should be a crucial requirement in order to provide *support for the lecturer*. Furthermore, it is recognizable that both groups of related work, i.e., *lecturer support* and *collaboration*, can benefit from the *means of adaptation*, which strengthens our assumption that adaptation plays an essential role in order to solve these issues. Finally, it can be seen that there is only one paper in which all three groups of related work are integrated into a single approach [Kub+17]. However, this approach only partially supports the requirements of each group. As this thesis targets to tackle the overall problem of supporting lecturers in *digital learning*

environments by using the means of adaptation, the following research question has to be answered (cf. section 1.3):

How can different levels of adaptation support the lecturer in properly using digital learning environments?

Table 3.1: A summary of the related work for lecturer support, collaboration and adaptation in *digital learning environments* (□ = criteria not fulfilled, ▤ = criteria partly fulfilled, ■ = criteria fulfilled, ■ = criteria fulfilled, but not object of research).

Systems	Approach	Lecturer → all Students	Lecturer → specific Stud.	Students → Lecturer	Lecturer Support			Adaptation			Collaboration		
					Predefined Scenarios	Scenario Proposals	Functional Proposals	Custom Scenarios	Conditional Scenarios	Runtime Adaptation	Group Formation	Question & Answer	Group Collaboration
[Riv19; SDG19]	Group formation	▤	□	□	□	□	□	□	□	□	▤	□	▤
[Jir+15; Rod+16]	Question & Answer (Q&A)	□	□	□	□	□	□	□	□	□	□	■	□
[ABO10; BP17]	Q&A combined with ARS	■	□	□	□	□	□	□	□	□	□	■	□
[CA14; SMB13]	Help requests	■	▤	□	□	□	□	□	□	□	□	□	▤
[MH16]	Group question creation	▤	▤	▤	□	□	□	□	□	□	▤	▤	■
[JB19]	Group problem solving	■	▤	□	□	□	□	▤	▤	▤	■	▤	■
[Har16; Fli17; FWB13]	Function selection	■	□	▤	□	□	□	▤	□	▤	□	■	□
[Kub+17]	Propose suitable functions	■	■	■	□	■	□	▤	□	▤	□	■	□
[Sch16]	Customized conditional scenarios	■	■	■	■	□	□	■	■	▤	□	□	□
[Qui16b]	Predefined scenarios	■	□	■	■	□	□	▤	□	▤	□	□	□
[HMB19]	Needed components for scenarios	■	□	□	▤	□	□	▤	▤	▤	□	□	□
[DG18]	Identify teaching methods	■	□	□	□	□	▤	□	□	□	□	□	□
This Thesis	Adaptable collaborative learn. env.	■	■	■	■	■	■	■	■	■	■	■	■

Table 3.1 summarizes how related work addresses different functional requirements and identifies the research gap that is targeted to be solved by this thesis. Moreover, it highlights the functional requirements that have to be fulfilled in order to be able to verify the research theses *RT1* and *RT2* that were listed in section 1.3 (requirements that have to be fulfilled for *RT1* are marked in green and those for *RT2* in purple):

RT1) Modeling adaptation allows lecturers to create customized teaching scenarios that support their individual teaching strategies.

RT2) Runtime adaptation allows adjusting teaching scenarios on the fly in order to respond to real-time results.

Furthermore, the table highlights the importance of using the *means of adaptation* in order to provide both *lecturer support* and *collaboration*, which can be seen by the relation between the colored boxes: The fulfilled requirements for *adaptation* (orange box) seem to influence the fulfillment of the *lecturer support* (blue box) and *collaboration* (yellow box). This also emphasizes the importance of *RT3*, which was formulated in section 1.3 as follows:

RT3) The concept of roles provides a promising extension to integrate the means of adaptation in *digital learning environments*.

While up to now only functional requirements have been considered, in the following, we will summarize *non-functional requirements* that have to be considered in the concept and the implementation of our approach. Furthermore, they will help to decompose our research theses that were repeated above.

Overall, we distinguish the non-functional requirements into two groups. The first group targets to *improve the quality of the execution* and therefore defines the following requirements:

NFR1) Usability: As motivated by the results of MobileQuiz2 (cf. [Sch16]), the approach has to be *easy to understand* by any lecturer and yet be able to be *as expressive as possible*. As this is somehow contradictory, the underlying concept has to be *easy to learn*, and an *intuitive interface* has to be provided in order to create teaching scenarios with ease.

NFR2) Correctness / Validability: The approach has to ensure that *only valid scenarios can be created*. There should be no point in time where lecturers accidentally execute invalid scenarios. The scenario should be validated, and potential issues should be displayed as those.

The second group describes requirements to *ensure the quality of future development and evolution*:

NFR3) Extendability: The approach must be easily extendable to add functions to support novel teaching methods later on.

NFR4) Scalability: The implementation has to be scalable and thus allow starting different scenario instances independently. The performance of a scenario should not be affected by the number of other scenarios being executed simultaneously.

NFR5) Portability: The concept should be transferable to other use cases as well.

NFR6) Standards: The concept should rely on existing solutions (i.e., standards) if these exist and it is possible to integrate them.

NFR7) Maintainability: The system should be easy to maintain. For example, necessary certificates should be automatically renewed, or servers should be automatically activated or deactivated, if required.

Having both functional and non-functional requirements in mind, the three research theses listed above are further decomposed.

In order to verify *RT1* (i.e., modeling adaptation allows creating customized teaching scenarios), the following sub research theses are defined that have to be verified:

RT1.1) Lecturers want to use teaching scenarios in which students are actively involved more often and are willing to support these with technical tools.

RT1.2) By defining elements and parameters for interactive activities through a (meta-)model, a variety of scenarios can be expressed, but the model is still easy to understand and extendable.

RT1.3) The integration of standard formats to express interactive activities will ensure the quality of the approach.

RT1.4) A graphical user interface enables lecturers to express their individual teaching strategies by allowing them to model customized teaching scenarios.

RT1.5) Supporting lecturers in getting started and during modeling eases the understanding of the modeling process.

RT1.6) The approach can be used by both lecturers and students without limitations to similar approaches.

In *RT2*, the means of runtime adaptation in order to adjust teaching scenarios on the fly should be verified. Therefore, the following sub research theses were defined:

RT2.1) Lecturers want to change their teaching scenarios during the execution.

RT2.2) Changing teaching scenarios on the fly allows implementing teaching scenarios that rely on student-generated data.

RT2.3) Limiting the changing of teaching scenarios to additions will avoid errors during execution and still be expressive enough to make changes.

RT2.4) Functional proposals provide a suitable extension in order to respond to real-time results even if the lecturer is not aware of the necessity.

Finally, RT3 is about investigating the concept of roles for integrating the means of adaptation in *digital learning environments*. In order to verify it, the following sub research theses have to be checked:

RT3.1) The concept of roles provides a useful extension to the (meta-)model in order to model a variety of different teaching scenarios.

RT3.2) The concept of roles provides a useful extension at runtime in order to allow for changes (i.e., role transfers) within single functions.

RT3.3) The concept of roles improves the extendability of the approach, as runtime data that is not specified by the (meta-)model can be added.

In the following three chapters, we will describe the concept, implementation and evaluation of an *adaptable digital learning environment* that allows lecturers to express their individual teaching strategies by creating customized scenarios, including collaborative activities. Thereby, both functional and non-functional requirements will be considered. In order to answer the global research question of this thesis, we will specifically focus on verifying or disproving the research theses that were defined before.

4 An Adaptable Collaborative Learning Environment

In this chapter, the concept of an *adaptable collaborative learning environment* will be presented that is targeted to solve the problem definitions presented earlier. In the first section, a *global picture* is shown that gives an overview of all components involved as well as their general ideas. Afterward, *each component* is presented in more detail and its *individual concept* is described. Finally, the *didactic opportunities* are motivated before a *summary* concludes the main design decisions that have been made.

Note on References

References to persons involved in generating ideas or designing concepts will be provided in the respective sections or subsections.

4.1 Conceptual Idea

The conceptual idea was motivated by discussions with Dr. Thomas Kühn, Prof. Dr. Tenshi Hara and Dr. Iris Braun, and further developed with Ilja Shmelkin.

In the previous chapter, we have shown that *related work* lacks at supporting individual scenarios. Even if those are supported, the approach (i.e., *MobileQuiz2*) suffers from its generics, meaning that lecturers cannot create their scenarios without a technical support. Although *MobileQuiz2* is hard to understand by all lecturers, we strongly believe that using a model to allow lecturers to design their individual scenarios is an appropriate approach. However, as *usability* (cf. *NFR1*) is an essential requirement, the model itself has to be easy to understand to allow lecturers to use it intuitively. In our opinion, this is only possible if the model is less generic and provides predefined elements (to represent the functional scope summarized in table 3.1). Since this results in a reduced expressiveness, elements have to be customizable in order to still support

the creation of individual scenarios. Moreover, the model has to be *easy to extend* (cf. *NFR3*) to implement novel functions or further didactic scenarios. In addition to the model itself, an intuitive interface has to be provided that supports the creation of individual scenarios. As a model-based approach is chosen, a graphical editor presents a suitable option for modeling as well as defining customized attributes. Furthermore, the created model must be parsable and has to be checked for its correctness (cf. *NFR2*). Finally, it has to be understood by a runtime that unlocks the defined functional scope.

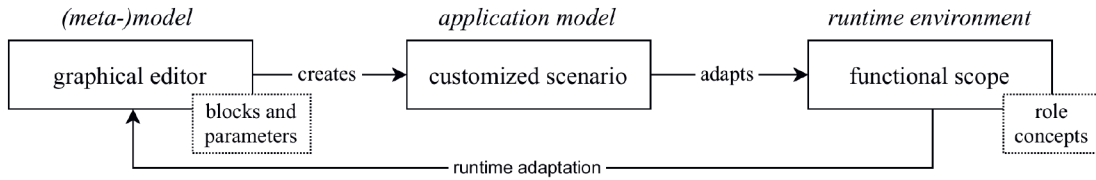


Figure 4.1: A summary of the conceptual idea for an adaptable collaborative learning environment (adjusted from [Kub19a]).

In summary, our concept consists of three main components, as visualized in figure 4.1, that are described in the following sections:

- A *(meta-)model* describes all available *blocks* (e.g., *structural* or *functional blocks*) with their individual *parameters* (i.e., *settings*).
- These blocks are provided in a *graphical editor*, which allows creating customized scenarios (later referred to as *application models*).
- These models are used to adapt the *functional scope* within a *runtime environment*, which allows lecturers to execute their individual teaching scenarios. This runtime is supported by *role concepts* to increase its flexibility. Moreover, *runtime adaptation* allows extending a scenario during execution, e.g., due to incoming results.

Note on the Terminology Used

Within the course of this thesis, the scenarios created by lecturers, which are built from the elements (i.e., *blocks*) defined by the (meta-)model, are referred to as **application models**. Within these *application models*, we distinguish between **structural blocks** (i.e., blocks to structure the scenario and model the lecture itself), **transition blocks** (i.e., transitions and forks), **functional blocks** (i.e., blocks that unlock an interactive functionality on the students' devices), as well as **blocks for visualization** (i.e., blocks to present different contents on the students' devices).

4.2 (Meta-)Model

The concept of the (meta-)model was developed together with Ilja Shmelkin during his master's thesis (cf. [Shm19]) – the results are published in [KSS19]. In the course of this thesis, the (meta-)model was refined and extended by further *functional*, *structural* and *transition* blocks, as well as different parameters.

The (meta-)model represents the main foundation of our concept that allows describing individual teaching scenarios by creating sequences of customizable functions. Therefore, it defines a variety of blocks, which are used to describe both the structure of the lecture and the interactive activities that are integrated into it. Each of these blocks defines different parameters that can be used to customize it to individual needs. The following groups of blocks are distinguished:

- *Structural blocks* allow modeling the general lecture, e.g., the start and end of it, the presentation given by the lecturer, or a traditional “offline” activity.
- *Transition blocks* are used to describe transitions between different blocks as well as their parallel, conditional or manual execution.
- *Functional blocks* represent unique interactive functionalities used to activate students in the ongoing lecture, e.g., by allowing them to answer questions or attend group tasks.
- Finally, *blocks for visualization* are added to present different media contents on the students' devices, e.g., an instruction of a task to solve.

In this way, we target to overcome the limitations (e.g., the *lack of comprehensibility*) while combining used-practice solutions with novel functionalities and settings. Even though this conceptual idea does not allow modeling arbitrary scenarios, it is easy to extend. This will be demonstrated by the addition of several blocks and parameters to implement seven well-known didactic scenarios: *Interactive Learning Questions*, *Peer Instruction*, *Jigsaw Classroom*, *Think-Pair-Share*, *Learning Stations*, *Peer Feedback* and *Learners-as-Designers* (cf. section 5.7). Before describing the currently available blocks in more detail, the following subsection will elaborate on choosing a suitable underlying (meta-)model that our (meta-)model will build on top.

4.2.1 Choosing an Appropriate Underlying (Meta-)Model

Within the master thesis of Ilja Shmelkin [Shm19], different (meta-)models that were considered to be potentially relevant were analyzed for their suitability to implement the concept of a (meta-)model, which is able to describe sequences of technology-enhanced teaching scenarios:

- *MobileQuiz2*, which was described in subsection 3.4.4,
- Autonomously Reconfigurable Workflows (in the following *Workflows*), and
- Σ -Automata (in the following *Automata*).

For comparison, different functional and non-functional requirements were defined:

- *Real-time capability* must be provided to allow application models to be adjusted even during execution, e.g., to add an additional question (cf. FR9),
- *Parameterizability* is required to describe different settings of an activity (cf. FR7 and FR8),
- *Simplicity* is a crucial requirement to enable the intuitiveness of the overall concept (cf. NFR1),
- *Validability* is required to enable checking the created application models (cf. NFR2), and
- *Extendibility* is necessary to add further blocks and parameters with ease (in order to support additional didactic scenarios) (cf. NFR3).

Each of the (meta-)models was evaluated by these criteria. While *real-time capability*, *parameterizability*, *validability* and *extendibility* could be rated for each approach, *simplicity* could only be rated for *MobileQuiz2*, as it was already evaluated by users (cf. subsection 3.4.4). Since *MobileQuiz2* lacks at fulfilling several requirements, its use was excluded. For the approach of *Workflows* and *Automata*, a user study was conducted in order to reason about the *simplicity* of these approaches when expressing different teaching scenarios. 11 participants attended this user study, 7 of whom had an IT background. The results of the comparison of different (meta-)models are summarized in table 4.1. For a more detailed description of why a particular approach fulfills a specific requirement, please refer to [KSS19].

Table 4.1: The fulfilled requirements of different approaches that can be used as an underlying (meta-)model, which is presented in [KSS19]. ✓ = Approach meets requirement, ✗ = Approach does not meet requirement.

Requirement	MobileQuiz2	Workflows	Automata
<i>Functional requirements</i>			
Real-time capability	✗	✓	✓
Parameterizability	✗	✓	✗
<i>Non-functional requirements</i>			
Simplicity	✗	✓	✓
Validatability	✓	✓	✓
Extendibility	✓	✓	✗

Although both approaches were rated to be intuitive, the participants decided unanimously for workflows when being asked to choose the approach they thought is more appropriate. Afterward, they were asked to use the chosen (meta-)model to complete several modeling tasks. Two user groups could be identified: One group modeled the task in a simple manner, while the other modeled more complex ideas. Moreover, it could be observed that the naming of elements and parameters will become a crucial aspect regarding the usability of the (meta-)model.

According to the results of the user study as well as the ratings of the requirements (cf. table 4.1), the approach of workflows was chosen as an appropriate (meta-)model to build on top. The general idea can be summarized as follows: The concept of our (meta-)model consists of *structural blocks* (including *StartBlock*, *LectureBlock* and *EndBlock(s)*), *transition blocks* (i.e., default transitions and different forks), *functional blocks* as well as *blocks for visualization*. Each of these blocks defines a variety of parameters that can be used to further customize it. This will allow creating individual workflows of customizable functions (cf. FR7), which represent personal teaching strategies. In the following four subsections, each group of blocks will be presented in more detail.

Note on the Parameters

Within the following four subsections, the function of parameters is only explained for selected examples. A list of all parameters and their description can be retrieved in the Appendix A.

4.2.2 Structural Blocks

In order to ease the understanding of the resulting workflows, our concept allows not only representing interactive activities (that are supported by technical tools) but also the lecture itself. Therefore, the following blocks are described by the (meta-)model:

- The *StartBlock* represents the start of the lecture. Only one *StartBlock* can exist.
- *EndBlocks* allow describing the end of the lecture. Due to conditional branching, multiple *EndBlocks* can exist – however, one is enough to describe the end.
- The *LectureBlock* describes a traditional part of the lecture, i.e., the presentation of a topic that is held by the lecturer.
- An *ActivityBlock* represents an “offline” activity that is executed in the lecture.
- Finally, *PauseBlocks* describe situations that require pausing, e.g., to let students think about a previously presented topic.

As a result of a preliminary user study, the *PauseBlock*, which was initially understood as pausing the functionality to hold the next part of the lecture, was semantically replaced by a *LectureBlock* and an *ActivityBlock* to describe these situations more intuitively. However, *PauseBlocks* can be used to describe actual breaks in the lecture. A summary of the structural blocks and their parameters is listed in table 4.2.

Table 4.2: An overview of the structural blocks and their parameters included in our concept. Parameters written in *italics* represent those set for the entire scenario. A description of all parameters can be retrieved in the Appendix A.

Structural Block	Description	Parameter(s)
StartBlock	Start of the lecture	<i>accessControl, advertise, anonymity, scenarioName, scenarioDate, pin</i>
EndBlock	End of the lecture	–
LectureBlock	Traditional part of the lecture	hasAudioVideoChat, audioVideoChatUrl, filter, timeout, autoFinishAfterTimeout, comment, topic
ActivityBlock	“Offline” activity in the lecture	hasAudioVideoChat, audioVideoChatUrl, filter, timeout, autoFinishAfterTimeout, comment, task
PauseBlock	Pause in the lecture	filter, timeout, autoFinishAfterTimeout, comment

As an example, the *LectureBlock* with its parameters is described: It is used to represent a traditional part of the lecture, e.g., a presentation given by the lecturer. The parameter

hasAudioVideoChat is used to decide whether the *LectureBlock* is held via an audio-video chat, as it is currently often done in the CoViD-19 pandemic. Using *audioVideoChatUrl*, the according URL can be set. If no URL is set, a new virtual room is created. The parameter *filter* allows limiting the presentation of this block to specific (groups of) students. For example, in a lecture with a hybrid schedule¹, the *LectureBlock* with the audio-video chat can be limited to students that work at home. Therefore, a question has to be asked, whether students participate from home or not, which can then be used to specify the *filter*. The parameter *timeout* allows defining a fixed duration for the block, after which the lecturer is asked to finish it. By setting *autoFinishAfterTimeout* to *true*, the block is finished automatically. Finally, *comment* is used to define a note for the graphical representation and *topic* to display information to the students.

Note on Similar Parameters

Contrary to the blocks presented later in subsection 4.2.4 and subsection 4.2.5, no inheritance is used at this point, as not all blocks have common parameters.

4.2.3 Transition Blocks

Transition blocks describe the transition between different blocks. Analogously to the domain of workflows, different kinds of transitions are provided²:

- A *DefaultTransition* is used to define a transition from one block to another.
- Moreover, different kinds of forks are included. An *AndFork* allows the parallel execution of an arbitrary number of blocks. *OrForks* are used to describe the conditional progression in the lecture, e.g., if the majority of students answered a previous question correctly, a specific block can be activated (cf. FR8). In addition, a *DecisionFork* is added to manually decide which block to proceed with.

In general, *transition blocks* include both blocks that are automatically (*DefaultTransition*, *AndFork*, *OrFork*) and manually (*DecisionFork*) forwarded. This means that lecturers have to decide for themselves which block to proceed with when activating the latter, while this happens automatically for the others. All *transition blocks* and their respective parameters are summarized in table 4.3.

¹ Half of the students are present in the classroom while the other half stays at home, thus ensuring distance in the classroom. This changes each day or week, depending on the federal state.

² During an early design phase, an *Iteration* was included. However, this was later partly replaced by the *DecisionFork*. Nevertheless, an automatically executed *Iteration* might be added in the future.

Table 4.3: An overview of the transition blocks and their parameters included in our concept. Parameters written in italics are later automatically defined in the workflow. A description of all parameters can be retrieved in the Appendix A.

Transition Block	Description	Parameter(s)
DefaultTransition	Transition from one block to another	<i>sourceID, destinationID</i>
AndFork	Parallel execution of blocks	<i>sourceID, destinationIDs</i>
OrFork	Conditional execution of blocks	<i>conditions, sourceID, destinationIDs, defaultDestinationID</i>
DecisionFork	Manual decision on the execution of blocks	<i>sourceID, destinationIDs, defaultDestinationID</i>

4.2.4 Functional Blocks

This group of blocks describes functions (i.e., interactive activities) that allow initiating interactions between the lecturer and the students, or even between students. Therefore, the representation of our concept for functions retrieved from current approaches (cf. section 3.2) are briefly summarized and novel components for the interaction between students are presented. All of the following blocks build on top of *FunctionBlock* and therefore inherit the parameters *comment, filter, timeout* and *autoFinishAfterTimeout*.

Interaction Initiated by the Lecturer

As defined in *FR1*, the interaction from the lecturer to all students has to be supported. Therefore, both qualitative and quantitative types of questions have to be added. In our concept, this is represented by the following function blocks:

- *LearningQuestions* allow students to check their gained knowledge and give feedback to the lecturer, which topics have not yet been understood. We defined several sub-types (retrieved from current systems): *SingleChoice, MultipleChoice, Freetext, Numerical, Order, Matching* and *Hotspot* learning questions. Another type that does not build on *LearningQuestion* but behaves similarly is the *GapTextQuestion*.
- *SurveyQuestions* allow students to provide an opinion on a certain topic. Therefore, they do not define a correct answer, which distinguishes them from *LearningQuestions*. The defined sub-types are *SingleChoice, MultipleChoice, Freetext, Numerical, FileUpload* and *Hotspot* survey questions.

Although there exist multiple standards that include the definition of different types of questions (e.g., the “IMS Question and Test Interoperability specification (QTI)”), none of them had been used, as the variety of parameters that were targeted to define could not be expressed. This especially holds for the definition of different feedback options, which is a critical requirement for *LearningQuestions*. Moreover, these standard formats strongly distinguish from the planned format of our (meta-)model and would not be consistent with it. Therefore, *NFR6* will not be fulfilled by our concept.

In order to provide full support for functional requirement *FR1*, the block *PresentMaterial* (cf. subsection 4.2.5) can be used to display textual messages to the students. This works similar to the concept of prompts that is presented by [Kap+14].

FR2 defines the limitation of questions or messages to specific (groups of) students. Therefore, our concept allows limiting the presentation of these blocks (i.e., *LearningQuestions*, *SurveyQuestions* and *PresentMaterial*) by defining different *filters*, e.g., a concrete given answer or the correct percentage for multiple prior *LearningQuestions*. These *filters* can be applied to all of the following functional blocks. The functional blocks and parameters to initiate interactions as a lecturer are summarized in table 4.4.

Note on Inheritance

It is important to note that for each abstract functional block (i.e., *LearningQuestions* and *SurveyQuestions*), a variety of concrete blocks exist. Each of them inherits the parameters of its corresponding abstract block, which will inherit the parameters of *FunctionBlock* (i.e., *comment*, *filter*, *timeout* and *autoFinishAfterTimeout*).

As an example, the *FreetextLearningQuestion* with its parameters is described: Since it inherits from *LearningQuestion*, which inherits from *FunctionBlock*, it has the parameters *comment*, *filter*, *timeout* and *autoFinishAfterTimeout* (cf. subsection 4.2.2). Moreover, it retrieves the parameters of *LearningQuestion*, e.g., *questionText* to formulate the question, *answerFeedback* to decide, whether students receive feedback on the correctness of their given answer, and *numberOfRepetitions* to give students multiple attempts to answer the question correctly (for all parameters, please refer to the Appendix A). Finally, the *FreetextLearningQuestion* defines its own parameters: *correctText* is used to specify the correct answer, *shortAnswer* can be set when only one word should be input, *caseSensitive* is set when the casing should be considered, and *characterMinimum* and *characterLimit* allow defining a minimum and a maximum number of characters to be input.

Table 4.4: An overview of the functional blocks and parameters to initiate interactions as a lecturer. Parameters written in *italics* represent references in the (meta-)model. A description of all parameters can be retrieved in the Appendix A.

Functional Block	Description	Parameter(s)
FunctionBlock → LearningQuestion		answerFeedback, showCorrectPercentage, allowAbstention, numberOfRepetitions, questionText, feedbackTexts, recordConfidence, displayType, displayPolicy, storagePolicy
SingleChoiceLearningQuestion	Select one correct answer	<i>choices</i>
MultipleChoiceLearningQuestion	Select multiple correct answers	<i>choices</i>
FreertextLearningQuestion	Enter the correct textual answer	correctText, shortAnswer, caseSensitive, textType, characterLimit, characterMinimum
NumericalLearningQuestion	Enter the correct numerical answer	correctMinNumber, correctMaxNumber, allowedMinNumber, allowedMaxNumber
OrderLearningQuestion	Bring choices in the correct order	correctOrderArray, <i>choices</i>
MatchingLearningQuestion	Match choices together	correctPairArray
HotspotLearningQuestion	Select the correct point on an image	image, imageURL, correctValue, correctRange
GapTextQuestion ³	Fill in the gaps	predefinedAnswers, completeText, gapPositions, misdirectionWords
FunctionBlock → SurveyQuestion		allowAbstention, questionText, voteCount, isAnswerChangeable, displayType, displayPolicy, storagePolicy, <i>peerBuilder</i>
SingleChoiceSurveyQuestion	Choose one answer	showAggregate, <i>choices</i>
MultipleChoiceSurveyQuestion	Choose one or multiple answers	showAggregate, <i>choices</i>
FreertextSurveyQuestion	Enter a textual answer	shortAnswer, textType, characterLimit, characterMinimum
NumericalSurveyQuestion	Enter a numerical answer	allowedMinNumber, allowedMaxNumber
FileUploadSurveyQuestion	Upload a file	–
HotspotSurveyQuestion	Select a point on an image	image, imageURL

³ *GapTextQuestion* does not inherit from *LearningQuestion* but provides a similar functionality.

Interaction Initiated by the Student(s)

As described in *FR3*, the interaction from the students to the lecturer is another function to be integrated. Therefore, both qualitative and quantitative feedback has to be supported. In our concept, this is represented by the following functional blocks:

- *ClosedFeedback* describes quantitative *Instant Feedback* and allows students to select from different feedback dimensions. These dimensions can be created individually.
- *OpenDiscussion* allows students to create their own questions or provide feedback. Depending on the set value for the parameter *visibleForAll*, the created feedback is either visible for anyone or just the lecturer.

If the feedback is visible for anyone (i.e., *visibleForAll* is set to *true*), the *OpenDiscussion* allows supporting the interaction between students. Depending on further parameters set (i.e., *allowVotingQuestions*, *allowAnswering*, *allowVotingAnswers* and *allowMarkCorrectAnswer*), the students can rate questions (which are sorted accordingly) or answer them in order to initiate discussions (cf. *FR11*).

In addition to supporting the interaction between all students, our concept includes functions that support the interaction between specific students, which is currently only rarely supported by existing approaches (cf. table 3.1). In the following, several innovative functions are presented that should be integrated into the targeted approach.

The concepts were developed together with Robert Peine during his practical courses and are partly published in an initial version in [KPB20].

Our concept supports two different types of interactions between specific students:

- *GroupInteractions* allow executing group tasks. Therefore, the most important part is the formation of groups using a *GroupBuilder* (cf. *FR10*), which does allow selecting a suitable *buildSchema* in order to form meaningful groups of students. The following build schemes were identified: *random*, *bestToWorst*, *similar*, *sameAnswer*, *differentAnswer*, *groupShuffle*, *groupMerge* (cf. [KPB20]). For example, in order to group students with different opinions on a previous *Learning- or SurveyQuestion*, the *buildSchema differentAnswer* can be used and referred to this question. Examples of the other *buildSchemes* can be retrieved in the Appendix B.1. Moreover, the *GroupBuilder* allows to either set a *groupSize* (i.e., the number of

students per group) or a *numberOfGroups* (i.e., the number of groups to be created). In order to allow the execution of group tasks, the *GroupBuilder* is followed by different group interactions (cf. FR12):

- A *GroupChat* allows group members to chat textually,
 - a *GroupAudioVideoChat* enables them to chat using audio and video,
 - *PresentGroupAnswers* allows prior given answers of the group members to be shown, and
 - *GroupVoting* can be used to select a unified group answer.
- In contrast, *PeerInteractions* describe interactions between peers, e.g., the execution of peer feedback. Similar to the *GroupInteractions*, a *PeerBuilder* will serve as the main component of these interactions⁴. Therefore, the following build schemes were identified: *random*, *bestToWorst*, *similar*, *sameAnswer*, *differentAnswer*, *sameGroup* and *differentGroup* (cf. Appendix B.2). Moreover, an arbitrary amount of *SurveyQuestions* has to be selected, on which the feedback should be collected. Finally, a *numberOfAssignments* can be set to control the number of feedbacks each student has to submit. The *PeerBuilder* is followed by different peer interactions:
 - *PresentPeerAnswers* is used to display the answers of a student to the questions that are referred in the *peerBuilder*.
 - *SurveyQuestions* can be reused and connected to the *PeerInteraction* by referring to the *PeerBuilder*. They allow enabling any kind of feedback, e.g., to rate the matched peer(s) using a *SingleChoiceSurveyQuestion* (with choices for grades such as 1, 2, 3, 4, 5 and 6).
 - *PresentPeerFeedback* can be used to display the given peer feedback on these *SurveyQuestions*, and
 - a *PeerChat* allows the exchange between the peer(s), i.e., between the feedback receiver and submitter(s).

Table 4.5 summarizes the functional blocks and parameters to initiate interactions as a student.

⁴ The *PeerBuilder* distinguishes from the *GroupBuilder*, as multiple assignments are possible.

Table 4.5: An overview of the functional blocks and parameters to initiate interactions as a student. Parameters written in *italics* represent references in the (meta-)model. A description of all parameters can be retrieved in the Appendix A.

Functional Block	Description	Parameter(s)
FunctionBlock → Feedback		<i>displayPolicy</i> , <i>storagePolicy</i>
ClosedFeedback	Provide feedback on defined dimensions	<i>feedbackText</i> , <i>showAggregate</i> , <i>cooldown</i> , <i>displayType</i> , <i>choices</i>
OpenDiscussion	Provide open-ended feedback	<i>allowVotingQuestions</i> , <i>allowAnswering</i> , <i>allowVotingAnswers</i> , <i>allowMarkCorrectAnswer</i> , <i>visibleForAll</i>
FunctionBlock → GroupInteraction		–
GroupBuilder	Form groups of students	<i>buildSchema</i> , <i>functionBlocks</i> , <i>groupSize</i> , <i>numberOfGroups</i>
GroupChat	Textual chat for group members	<i>task</i> , <i>groupBuilder</i>
GroupAudio-VideoChat	Chat using audio and video	<i>groupBuilder</i>
PresentGroup-Answers	Show group members previously given answers	<i>functionBlock</i> , <i>groupBuilder</i>
GroupVoting	Select a group answer	<i>functionBlock</i> , <i>groupBuilder</i>
FunctionBlock → PeerInteraction		–
PeerBuilder	Assign peer feedback tasks	<i>buildSchema</i> , <i>functionBlocks</i> , <i>questions</i> , <i>numberOfAssignments</i>
SurveyQuestions ⁵	Allow providing peer feedback	<i>peerBuilder</i> (for further parameters, see table 4.4)
PresentPeer-Answers	Display answers of students to give feedback for	<i>peerBuilder</i>
PresentPeer-Feedback	Display peer feedback to the student	<i>peerBuilder</i>
PeerChat	Allow the exchange between feedback receiver and submitter(s)	<i>peerBuilder</i> , <i>task</i>

4.2.5 Blocks for Visualization

This type of blocks describe the visualization of non-interactive content on the students' devices. Therefore, the following blocks are defined:

⁵ *SurveyQuestions* do not inherit from *PeerInteraction* but are functionally tightly coupled.

- *PresentMaterial* allows displaying media content on the students' devices, e.g., an image, a video, a text, or similar content. Therefore, the parameter *content* must be specified in the syntax of the markup language Markdown that allows expressing formatted text (including media contents) with a plain text format.
- *PresentCountdown* is used to display a countdown on the students' devices. Therefore, the parameter *timeout* has to be set, which defines the duration of this countdown.
- Finally, *PresentResult* allows displaying the result of one or multiple survey- or learning questions on the students' devices. This is done by defining those using the parameter *functionBlocks*.

Note on Inheritance

Similar to functional blocks, each of these blocks inherits the parameters from its parent (*VisualizationBlock*), i.e., *comment*, *filter*, *timeout* and *autoFinishAfterTimeout*.

In table 4.6, the blocks for visualization and their parameters are summarized.

Table 4.6: An overview of the blocks for visualization and their parameters that are included in our concept. Parameters written in *italics* are defined in the parent but are essential for the corresponding block to work and are therefore listed. A description of all parameters can be retrieved in the Appendix A.

Block for Visualization	Description	Parameter(s)
PresentMaterial	Present material on students' devices	content
PresentCountdown	Present a countdown on students' devices	<i>timeout</i> , <i>autoFinishAfterTimeout</i>
PresentResult	Present result(s) of previous question(s)	functionBlocks

Note on Adaptation at Runtime

The created workflows are fixed – however, it is planned to extend those at runtime (cf. FR9), which is either done by manually adding a sub-scenario or by automatically adding it when confirming a functional proposal (cf. FR6). Further details will be presented in section 4.6.

4.2.6 Preliminary Evaluation of the (Meta-)Model

Note on the Position of this Evaluation

As the results of this evaluation are crucial for developing the remaining concept, they are presented in this subsection instead of presenting them in chapter 6.

A preliminary version of the (meta-)model was evaluated at the end of the master thesis of Ilja Shmelkin (cf. [Shm19]) to reason about its understandability and intuitiveness. As a graphical editor was not present at this time (and also not part of this master's thesis), an approach had to be developed to simulate it. Therefore, paper-based representations of the *structural blocks*, *transition blocks*, *functional blocks* and *blocks for visualization* were created that were used as a construction kit to build customized teaching scenarios by simply placing those elements one after another, as shown in figure 4.2.

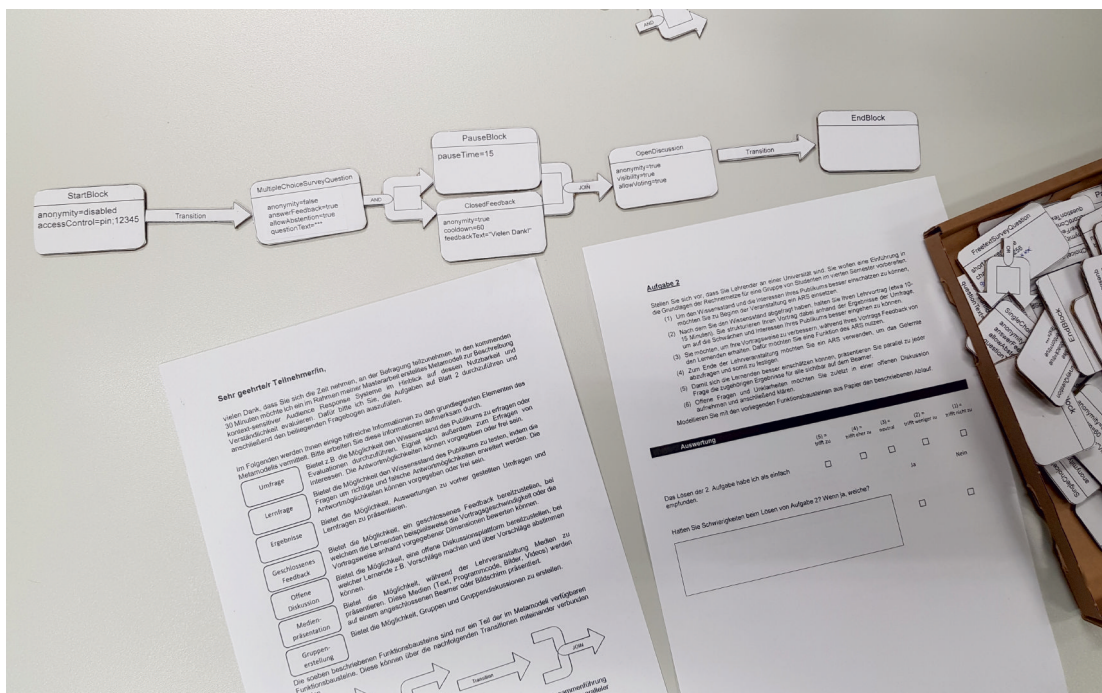


Figure 4.2: A paper-based construction kit to build customized teaching scenarios [Shm19].

While a graphical editor is able to support the user by providing descriptions, e.g., using tooltips, similar support had to be provided in this evaluation. Thus, in addition to a short introduction to the evaluation, a summary of all blocks and their underlying functionality was described on the first page, as can be seen in figure 4.2 on the left paper. The main part of the evaluation consisted of three tasks, where each was followed by a

question regarding the understanding as well as an open-ended feedback opportunity. In the initial task, the participants were asked to model three simple teaching scenarios: First, a *SingleChoiceLearningQuestion* should be presented to the students. Next, a *MultipleChoiceLearningQuestion* with a parallel *PresentResult* block should be modeled and finally, a *SingleChoiceLearningQuestion*, which is followed by a conditional decision (i.e., an *OrFork*) that either unlocks an *OpenDiscussion* or a *GroupInteraction*, had to be created. The goal of this task was to check whether participants understand the chosen concept of workflows or not. In the next task, the participants got descriptions of different problems occurring in a lecture, e.g., in order to improve the lecturing, students should be able to give feedback during the lecture, or at the end of the lecture, they should check their gained knowledge. This task was used to evaluate whether the participants were able to associate problems occurring in a lecture with the appropriate functional blocks. In the last task, a representation of the teaching method *Peer Instruction* was presented to the participants (cf. figure 4.3), who were asked to describe the functionality of the model as well as the meaning of the functional blocks and their connections.

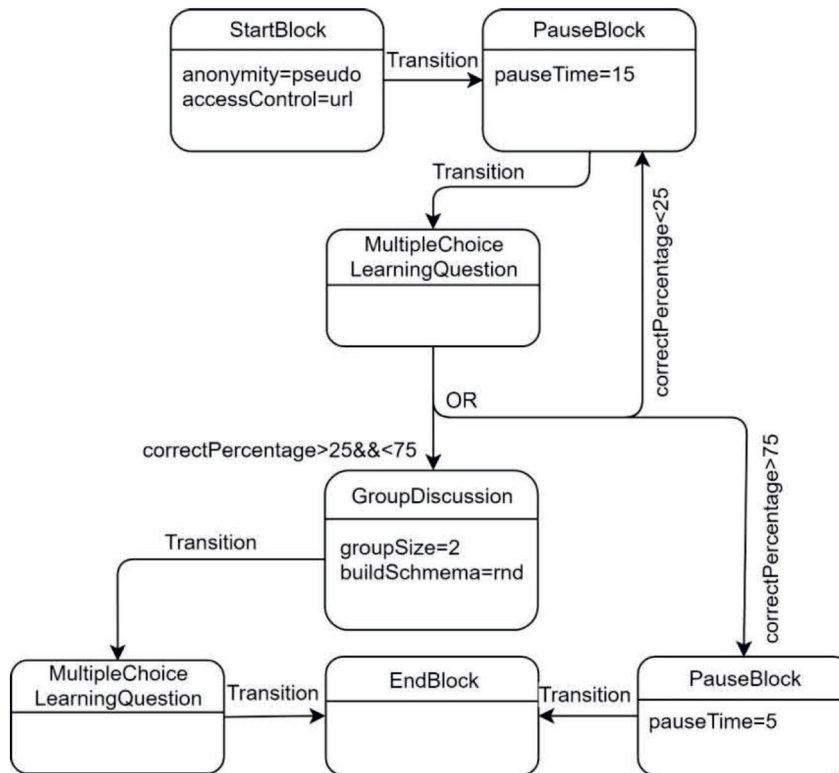


Figure 4.3: A representation of *Peer Instruction* that was used in the evaluation of the (meta-)model (cf. [Shm19]). The *PauseBlocks* were initially used to describe lecture phases (i.e., the *brief lecture* and the *conclusion of the topic*). The *conceptTest* is described by a *MultipleChoiceLearningQuestion*.

Moreover, they were asked to name the concept if they knew and could identify it. Within this task, the comprehensibility of the (meta-)model should be evaluated. Finally, the participants were presented with several questions regarding their person and prior experience, as well as their opinion regarding the (meta-)model.

In total, 20 participants attended the evaluation that was executed in individual sessions of 20 to 30 minutes. 11 participants had an IT background and 14 already used models such as UML. The results of the three tasks are summarized in figure 4.4: Task one could be solved with little or no trouble at all by every participant, resulting in 15 participants rated it as easy to solve and 5 as partly easy. Even the 9 participants without an IT background were able to model the described scenarios. However, several participants mentioned that the naming of the blocks is somehow confusing. This was later be adjusted in the concept (by adding a *LectureBlock* and an *ActivityBlock*). The second task was more difficult, as only the problem was given, but the blocks had to be searched alone. This is also visible in the ratings, resulting in only 6 participants rated the task as easy to solve, 13 as partly easy to solve and one who found the task neither easy nor difficult. Most of the confusion was caused by the correct use of *AndForks*, *OrForks* and *Joins*. The latter was often not used or used after both *AndFork* and *OrFork*. The concept of the graphical editor should consider these observations. In the last task, a model of *Peer Instruction* was shown to the participants and should be interpreted. 12 participants rated this task as easy, 6 as partly easy and one participant each as neutral and partly hard to understand.

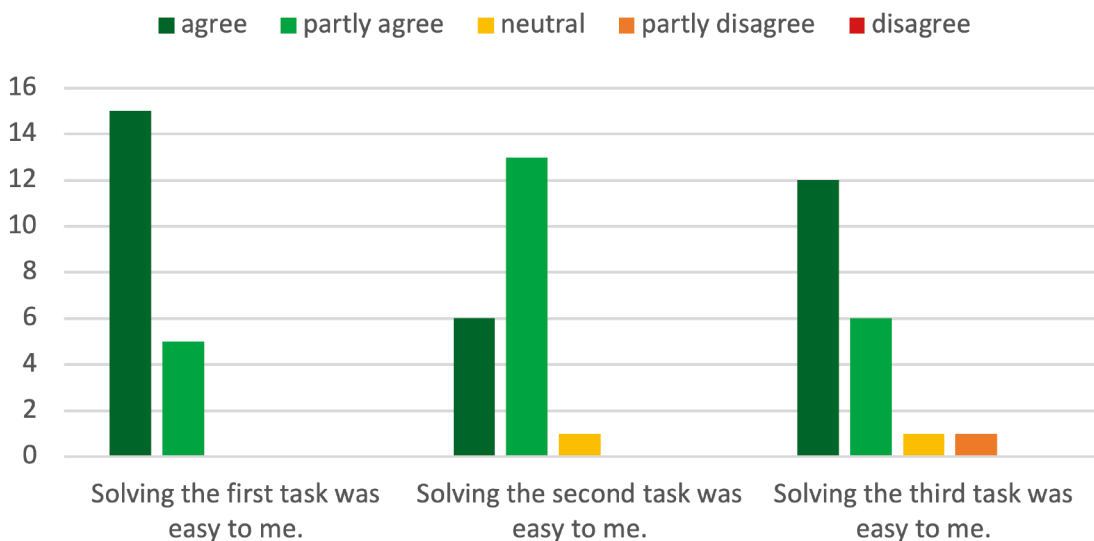


Figure 4.4: A summary of the evaluation results for the (meta-)model [KSS19].

The (meta-)model itself was rated by 12 out of 20 participants as easy and by the remaining 8 as partly easy to understand. Furthermore, 18 participants stated that they would like to use a learning environment based on the (meta-)model in the future.

Even though this confirms the requirement of intuitiveness, it is largely influenced by the naming conventions for the blocks as well as the parameters. Moreover, the concept of the graphical editor will have a substantial impact on the understandability of the overall concept – especially the modeling of parallel or conditional executions has to be investigated further. Further information on the evaluation can be retrieved in [Shm19].

4.3 Graphical Editor

The concept of the graphical editor was developed together with Lidia Roszko during her bachelor's thesis (cf. [Ros19]). In the course of this thesis, it was extended by further components (e.g., to support the lecturer in getting started or in modeling more complex scenarios), as will be described in the following subsections.

While the main foundation of our concept is described by the (meta-)model and was evaluated to be easy to understand (cf. subsection 4.2.6), the concept itself is largely influenced by the user interface, or more specifically, the editor that allows lecturers to create customized teaching scenarios. Therefore, in addition to the basic concept of a graphical editor, several supporting concepts will be introduced that were integrated to ease both the usability and the modeling process.

As the end-users of the graphical editor are lecturers with different modeling abilities who want to create their individual teaching scenarios intuitively, users were included in the entire development process. Therefore, an adapted version of the user-centered design process was chosen: After considering related work on establishing *usability*, an initial user study was conducted to retrieve users' perceptions on general design decisions for the editor (e.g., the position of menus). Based on these results, a first concept was created that was refined iteratively in several user studies. Finally, additional supporting components were investigated in order to cope with the complexity of the extensive functionality (i.e., the large number of blocks and parameters that were defined).

4.3.1 Basic Concepts for Usability

As a starting point, existing editors were investigated regarding concepts to improve their *usability* (cf. *NFR1*), which is a critical requirement when developing graphical user interfaces. The following six components were identified:

- *Instructions* help to improve the ease of learning and understanding of the functional scope,
- the *selection of the preferred workflow* allows avoiding the cognitive overload by hiding functions and elements intelligently,
- choosing appropriate *colors* improves the overall look and feel, and can help to increase the visibility of elements and generate accessibility,
- *comprehensible metaphors and placement of components* help to reduce the training time, as familiar concepts are recognized,
- *templates* allow a quick start in the editor by providing structures to build upon, and
- a *decision support system* helps to find appropriate elements or templates for a specific use case.

Note on the Sections' Structure

In this section, each of these components will be discussed in more detail.

4.3.2 General Design and Concept Decisions

In the first step, decisions had to be made regarding the structure and general functionality of the editor (i.e., identify appropriate *placements of the components* and *metaphors*). Therefore, a user study was conducted, in which participants had to fill out a survey that was structured as follows: After a short introduction and some questions regarding the modeling expertise and editors used so far, the participants had to rate 24 statements (visualized by examples) using a Likert scale (*strongly agree, agree, neutral, disagree, strongly disagree*). These statements were related to the placement of panels and menus, the preferred visualization of elements in the *element palette* (i.e., a selection menu for available elements), as well as to metaphors for adding and connecting elements. Additionally, a freetext question was added that allowed for qualitative feedback.

In this survey, a total of 34 participants was involved, 15 of them without modeling expertise and 11 with limited experience in this area. However, only 8 participants stated that they use graphical editors frequently. The most used editors were *Word* and *Visio* from Office 365⁶, *draw.io*⁷ and *dia*⁸. In the following, the Likert scales were converted into values ranging from 1 (*strongly disagree*) to 5 (*strongly agree*). The results larger or equal 3 are summarized as follows (the specific rating is added in quotes):

The *main menu* should be placed on the top (4.50) of the editor (followed by placing it on the left with a rating of 3.29).

The *element palette* is preferred on the left (4.32), followed by the top of the editor (3.88) and the *modeling canvas* itself (3.18).

The *properties panel* that allows defining parameters can either be placed on the right of the editor (4.15) or next to the element (3.32).

For the *selection of elements*, several visualizations were chosen to be intuitive: Tabs (4.12), accordion (3.56), item list (3.50), or a traditional menu (3.32).

The *options for inserting elements* were also rated positively, while the participants preferred drag-and-drop (4.24) over clicking on an element to insert it (3.47) as well as clicking on an element followed by the *modeling canvas* afterward (3.38).

Finally, *connecting elements* was preferred by using magnets on the elements⁹ (4.41) rather than a halo¹⁰ (3.09) and a global connect tool¹¹ (3.00).

Further information and visualizations on all 24 statements that were included in the user study can be retrieved in [Ros19].

As a result of the initial user study, three variants of a graphical editor were created and used as a starting point for another user study, in which the previously selected components should be further refined. One of these variants is displayed in figure 4.5. For ease of understanding, each component is marked by a number, which will be used for the description in the following. In the second user study, interviews with the participants were conducted that were asked to make decisions on the following components:

⁶ <https://www.microsoft.com/en/microsoft-365> – last successful access on October 8, 2021

⁷ <https://app.diagrams.net/> – last successful access on October 8, 2021

⁸ <http://dia-installer.de/> – last successful access on October 8, 2021

⁹ A number of magnets are placed on the border of the element that allow creating connections.

¹⁰ A small menu that is displayed next to the element and allows creating connections.

¹¹ By selecting this function, clicking on the following two elements creates a connection between them.

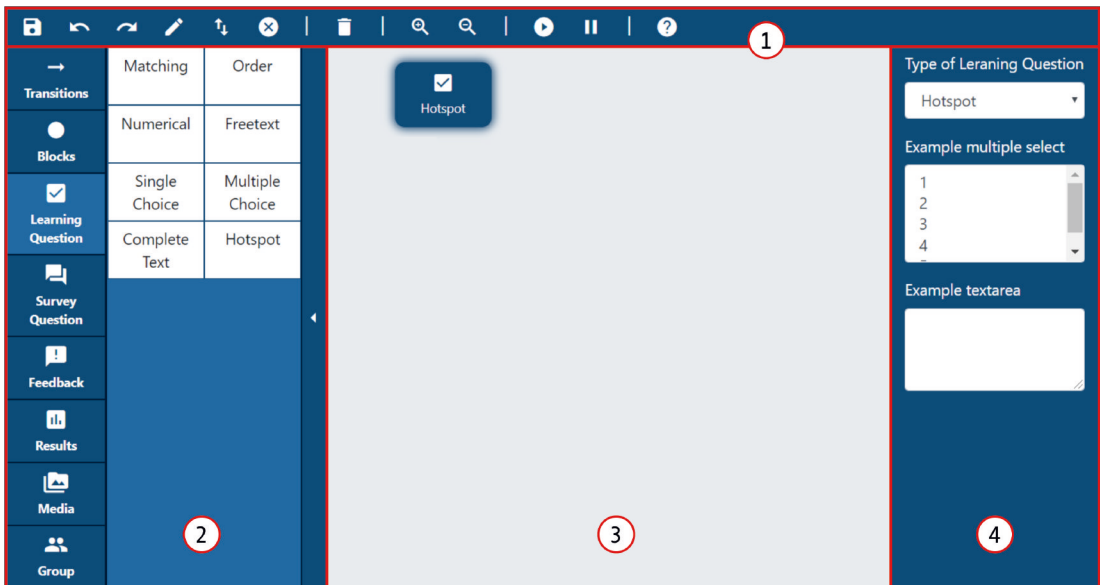


Figure 4.5: A preliminary concept of the graphical editor that was created as a result of the initial user study. The figure was extended by numbered markers for each component (adjusted from [Ros19]).

- The elements of the *main menu* ① and their order,
- the type and functionality of the *element palette* ② and the order of the groups,
- the way elements (i.e., *structural blocks*, *transition blocks*, *functional blocks*, or *blocks for visualization*) and their labels are presented on the *modeling canvas* ③,
- the way the connect option is presented,
- an initial template option,
- a help option,
- as well as different colored variants.

The user study was conducted with 10 participants. The answers and ideas were manually written down and can be summarized as follows:

Most participants agreed with the options provided in the *main menu* ① and were able to describe the underlying function of each icon, with the exception of the *import/export* as well as the *delete all* function. In order to improve the *main menu*, several proposals were made, e.g., adjust the position of *undo* and *redo* as well as the *save* option, or combine functionalities such as the *play* and *pause* option,

as only one of them can be active at one time. Moreover, several shortcuts were proposed that follow used practices of existing systems¹², i.e., **Ctrl** + **Z** (undo), **Ctrl** + **Y** (redo), **Ctrl** + **C** (copy), **Ctrl** + **V** (paste), **Ctrl** + **X** (cut), **Ctrl** + **S** (save), **Delete** (delete), **Ctrl** + **+** (zoom in), and **Ctrl** + **-** (zoom out). Furthermore, the addition of an input field for defining and displaying the scenario name was proposed.

The majority of participants agreed with the provided *element palette* ② and the option to distinguish between abstract and specific types but proposed that it should be closed at the beginning. They were unsure about displaying one or two specific types in a row at one time – this should be decided later on. The *properties panel* ④ should contain an option to change between the specific types, as shown in figure 4.5, and also an option to delete or copy the element. The order of the abstract elements should be taken into account the frequency of usage as well as the relationship between elements. Therefore, the order of *Results* (i.e., presenting a result on the students' devices) and *Media* (i.e., presenting a media content) was proposed to switch and *Feedback* was moved further down. This should be handled similarly for the specific types, e.g., for *LearningQuestions*, the sub-type *SingleChoice* should be followed by *MultipleChoice*, *Freetext* and *Numerical*.

Regarding the visualization of blocks, 9 out of 10 participants selected the option of an icon with the name of it on the bottom over the following options: name of the element on the right, the name inside the icon, and the name inside a rectangle. Furthermore, they decided not to use colors for different elements, as this might be confusing. For the visualization of forks, no preference could be recognized. Therefore, the type of the fork was written in the middle of the shape.

When hovering or clicking a specific block on the *modeling canvas* ③, the magnets (that are used to create connections) should become visible.

The participants agreed with the idea of an initial starting template that displays the *StartBlock* and maybe also an initial *Transition* (i.e., connection). Further templates should be able to be retrieved via an option in the *main menu*.

Regarding help options, the participants emphasized the importance of tooltips to explain elements as well as functionalities in the *main menu*, the *element palette*, as well as the *properties panel*. Moreover, they agreed with the idea of a help dialog in the *main menu*.

¹² The shortcuts for Windows and Linux are presented. On macOS, **Ctrl** has to be replaced by **Cmd**.

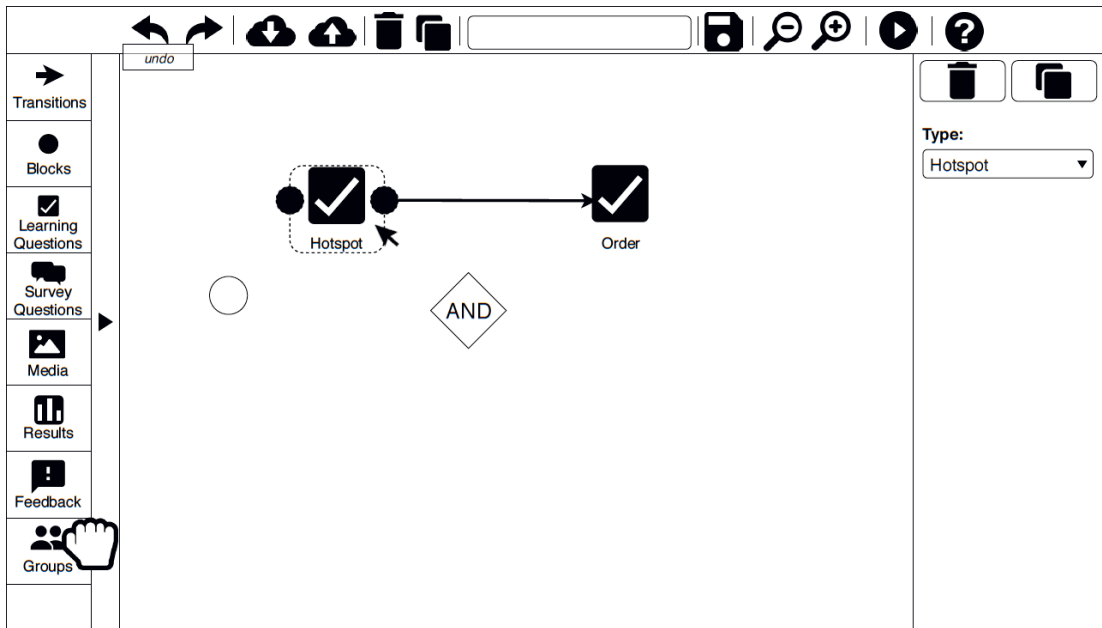


Figure 4.6: A refined concept of the graphical editor that was created after the second user study. For a better understanding of the included concepts, the colors have been omitted [Ros19].

The majority of participants selected the variant in dark blue that is visualized in figure 4.5 as the most appropriate one. However, they stated that further variants, e.g., a variant that has better contrast, should be available, too.

An adjusted version of the concept that visualizes the chosen metaphors is depicted in figure 4.6. For ease of understanding, colors were omitted. Further information on the results of the interviews can be retrieved in [Ros19].

While the concept was implemented prototypically after completing the second user study, further adjustments were made due to later extensions of the (meta-)model, as well as the feedback that was retrieved on three user studies conducted during the course of this thesis. In the following, the main decisions that refine the previously presented concept are summarized:

As retrieved from the results of the (meta-)model (cf. subsection 4.2.6), adding *Joins* was omitted in the concept of the graphical editor. Instead, users can simply draw multiple connections on a single element – the editor will insert the required *Join* in the background. This concept has been evaluated to be well-accepted in the final user study of [Ros19].

In a later user study, the importance of a *reset* function was highlighted to clear the entire *modeling canvas* (i.e., delete all elements). Moreover, a *center* function was proposed that allows to properly display the diagram, and zoom in or zoom out accordingly. Both of these functions had to be added to the *main menu* ①. Instead, the *play* button was removed, as the execution should later be independent of the modeling process.

Furthermore, the order of the *main menu* items was revised due to user feedback. On the left side of the scenario name input, all options were placed that are used to manipulate the display of the *modeling canvas*, i.e., *reset* and *center canvas*, *undo* and *redo*, as well as *zoom out* and *zoom in*. On the right, the remaining options are displayed, i.e., *load/import* and *save / export*, *copy* and *delete* elements, as well as the *help* and the *save dialog*.

Due to adjustments of the (meta-)model and the fact that these groups contain *blocks for visualization*, it was decided that *Media* and *Results* are inserted into the *Block* group. Furthermore, *Blocks* were moved to the first place and *Transitions* to the second. As *Transitions* also contain *Forks*, the group was renamed into *Transition & Forks*. Moreover, *Groups* were renamed into *Group Interactions* and another group was added, called *Peer Interactions*, which represents functions to execute scenarios with peer feedback, which was not part of the first version of the (meta-)model. Due to these adjustments, the number of groups was also reduced to 7, which suits much better for smaller devices.

Furthermore, the expand option of the *element palette* was investigated in more detail. Instead of providing a full-height button to expand or collapse the *element palette*, the sub-menu can be opened when a group is pressed and similarly also be closed if it is pressed again. Therefore, additional shortcuts should be added to select these groups and insert elements even easier.

The concept of the *properties panel* was further developed throughout the following user studies. The parameters of each element are distinguished between required and optional parameters, which allows avoiding the cognitive overload of a multitude of parameters at one time. Moreover, the buttons for deleting and duplicating an element were replaced by an input of the element's name, as both previous functions are present in the *main menu* and can also be called by shortcuts.

During implementation, it was recognized that full width or full height panels (i.e., *main menu*, *element palette* and *properties panel*) result in a lot of whitespace and reduce the *modeling canvas* even further. Therefore, these panels were adjusted to use only

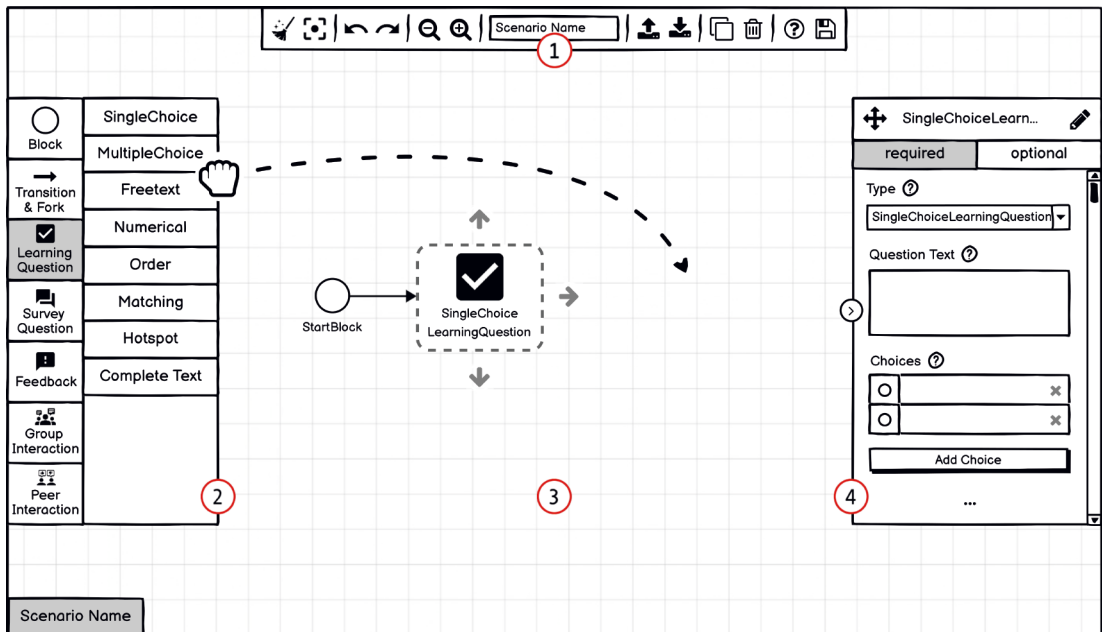


Figure 4.7: The final mockup of the graphical editor, in which the general design and concept decisions are visualized. For ease of understanding, the colors were omitted. Numbers are added for description purposes.

the necessary space. In this turn, the *properties panel* was adapted to be freely movable and collapsible if it limits the *modeling canvas* too much.

Moreover, a grid has to be added in the background in order to align elements more easily. Additionally, lines for properly aligning elements should be visible when inserting an element using drag-and-drop.

These adjustments resulted in the final concept that is visualized in figure 4.7. In the following, each component is described in more detail. The *main menu* (1) includes general modeling controls, which are presented in the following (from *left to right*). The *italic written* names represent the tooltip of a specific option:

- *Reset canvas* can be used to remove all model elements and connections in order to reset the scenario.
- *Center canvas* allows the centering and correct zooming of the modeled diagram.
- *Undo* will reset the last modeling step, while *redo* will repeat it if it is in the past.
- *Zoom out* can be used to manually zoom out the diagram and *zoom in* to manually zoom in.

- The *textual input* is used to define the scenario name.
- *Load/Import* allows loading a template from the system or from an external file (cf. subsection 4.3.3).
- *Save/Export* is used to save a template in the system or export it as an external file (cf. subsection 4.3.3).
- *Copy* is used to create a copy of the currently selected element, while *delete* will remove this element.
- *Help* provides several dialogs such as a summary of all elements and parameters, a list of shortcuts or the settings of the editor. Additional supporting dialogs are summarized in subsection 4.3.3.
- *Save* enables to create an instance of the created scenario that can be executed later.

The *element palette* ② displays all available *structural blocks*, *transition blocks*, *functional blocks* as well as *blocks for visualization*. While *functional blocks* are sorted by their abstract types (as a group), *structural blocks* and *blocks for visualization* are sorted into the group *Block*, and *transition blocks* into the group *Transition & Forks*. By clicking on a group (i.e., the parent type such as *Block* or *LearningQuestion*), the specific elements are listed that can be inserted using drag and drop. Additionally, it will be allowed to also add abstract elements using drag and drop in order to model more generic scenarios. In the following, the groups and their types are summarized (descriptions of all specific types can be retrieved in section 4.2):

- *Block* includes the *structural elements* as well as the *blocks for visualization*: *StartBlock*, *EndBlock*, *PauseBlock*, *LectureBlock*, *ActivityBlock*, *PresentMaterial*, *PresentResult* and *PresentCountdown*.
- In *Transition & Forks*, the *transition blocks* are provided: *(Default-)Transition*, *AndFork*, *OrFork* and *DecisionFork*.
- All remaining five groups represent abstract functional blocks (*LearningQuestion*, *SurveyQuestion*, *Feedback*, *GroupInteraction* and *PeerInteraction*) that list their specific functional blocks according to the order presented in section 4.2.

Each teaching scenario is created in the *modeling canvas* ③, which represents the main component of the editor and thus, takes the largest part of it. Within the *modeling canvas*, all elements except *transition blocks* (i.e., *structural block*, *functional block* or *block of visualization*) are depicted by a specific shape as well as a name that is initially set to the type of the element (but can be customized later). The elements can be moved freely onto the *modeling canvas*, which allows customizing the created scenarios even further.

When clicking on an element, the *properties panel* ④ opens, which enables to set values for different parameters to customize it. These parameters are sorted between *required* and *optional* parameters. If no required parameters exist, the tab for the *optional* parameters will be opened initially. Each parameter is described by a tooltip that can be opened by hovering over the *info circle* icon. The input of the tooltips is set specific to the type as well as the parameter itself, e.g., for a *questionText* (i.e., the formulation of the question), a textarea is an appropriate input, while for *correctText* (i.e., a concrete answer to a *FreetextLearningQuestion*), a text input is suitable. Moreover, the name of the element can be changed by clicking on the *pencil* icon. The *properties panel* can be moved to any position within the editor, or can also be collapsed, e.g., if only the structure of the scenario should be modeled without directly defining values for the parameters.

Overall, the general concept presents an appropriate starting point in order to create customized teaching scenarios with ease. However, in order to support this modeling process even further, several supporting concepts have been added that are described in the following subsection.

4.3.3 Supporting Concepts

Although both the (meta-)model and the graphical editor were designed through user-centered approaches, the underlying modeling approach does not necessarily make the developed concept directly understandable to every lecturer without providing support. Consequently, in the course of this thesis, several supporting concepts have been investigated that target to *support the lecturer within the initial phase* in the system, *the modeling*, as well as *the creation of complex scenarios*. In the following, for each of these groups, different supporting concepts are presented in further detail.

Support the Initial Phase in the System

In the final evaluation of the initial implementation of the graphical editor, it became apparent that lecturers who have not yet used the system find it difficult to get started with it (especially when having in mind that they have to handle it on their own). In order to tackle this problem, a variety of supporting concepts were investigated, which pursue the following goals:

- Presenting the underlying concept in a way that is understandable for all lecturers,
- supporting lecturers in getting started with the editor by describing the editor surface as well as the functioning of it, as well as
- supporting them to choose appropriate elements or templates to be integrated into their individual scenario.

The concepts for supporting the initial phase were developed together with Lidia Roszko and Robert Peine during a practical course as well as a master thesis. Several of these concepts were presented in [KRT20].

In order to *present the underlying concept intuitively*, the general approach (i.e., allow using technical tools in different application scenarios that are adapted to the teaching strategy of the lecturer), the functioning (i.e., after the graphical modeling, the scenario is saved and can be executed), as well as the realization of well-known didactic scenarios (e.g., *Peer Instruction*, *Jigsaw Classroom* or *Think-Pair-Share*) should be explained comprehensively on the landing page. This will allow lecturers to decide upfront whether the concept meets their requirements without having to manually trial and error. The functioning can be either explained by different screenshots or by a short demonstration video.

After lecturers decide to use the system, they have to be supported in *getting started with the editor*. Therefore, an *overlay* was investigated that is opened up when using the graphical editor for the first time. It consists of two steps: First, the main components of the graphical editor are presented as shown in figure 4.8, and secondly, an example model of *Peer Instruction* is presented, in which each step is added to the *modeling canvas* and its functionality is described to the user. In the future, it is planned to add further examples of well-known didactic scenarios.

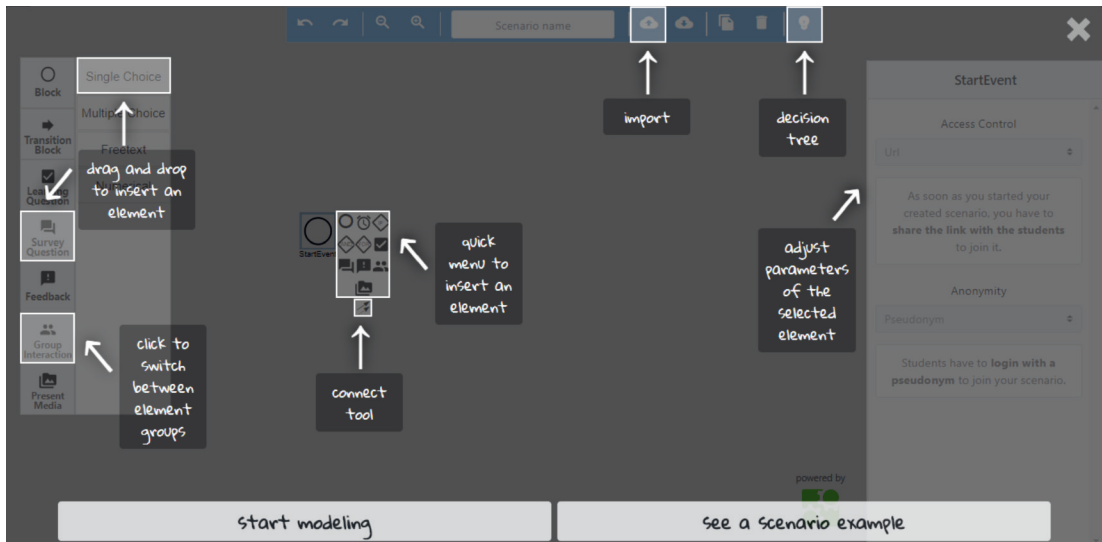


Figure 4.8: The first step of an *overlay* that is opened on the first usage of the editor [Ros19].

Afterward, lecturers should have a common understanding of the underlying concept as well as the components and functioning of the graphical editor. However, in order to *support lecturers in choosing suitable elements or templates*, the concept of a *decision tree* was investigated. The *decision tree* allows selecting elements based on their type of interaction. After answering between whom the interaction should take place, one to two further questions are asked, and finally, a list of appropriate elements is displayed. The idea of the *decision tree* is visualized in figure 4.9.

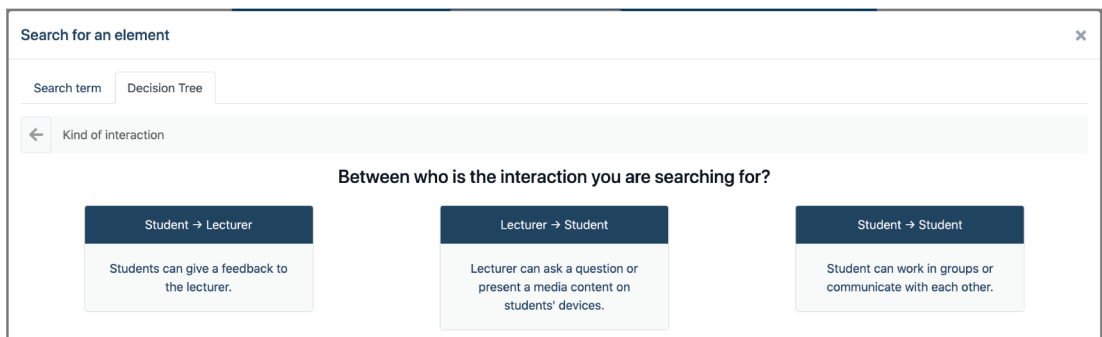


Figure 4.9: The initial concept of a *decision tree* that is able to select suitable elements based on the type of interaction [KRT20].

While the *decision tree* is suitable to select appropriate elements, it relies on a lot of manual effort, as a variety of questions have to be answered. Furthermore, it does not allow adding groups of elements, as it is required for group or peer interactions (e.g., a *GroupChat* must refer to a previous *GroupBuilder*).

Consequently, as an extension, a proposal-based support function was investigated by Robert Peine during his master thesis (cf. [Pei21]). This function is supposed to analyze the model that is currently being created, retrieve the intention of the user and make suitable suggestions to complete this model. The goal is not only to support new users in becoming familiar with the possibilities of modeling but also to provide inspiration for the use of didactic concepts. Furthermore, it might also reduce the manual effort when modeling complex scenarios.

In order to design this functionality, related work was investigated that supports users by providing context-sensitive proposals. Therefore, five papers from different areas were considered, which allowed visualizing the wide variety of possibilities to determine and present suggestions. In addition, different options to trigger the generation of suggestions as well as describe and sort them were presented (cf. [Pei21]).

Afterward, the concept was formulated: The function implements a *forward completion*, i.e., based on an element selected by the user, subsequent elements are suggested. The suggestions are made using a *knowledge base* created by the administrator of the system. The rules of this *knowledge base* are designed to be very expressive: For example, they allow defining *subpatterns* (i.e., a set of connected elements) that have to be found (in a specific search area) as well as their dependencies on each other. The ranking is calculated based on the matching between the *subpatterns* and the existing scenario (cf. [Pei21]).

The definition of the *knowledge base* by the administrator allows controlling the suggestions instead of having to rely on automatic reasoning. Thus, specific didactic methodologies can be suggested that are supported by the system. However, it is also possible to define the rules (in the *knowledge base*) according to the *used practices* of the users (an approach to retrieve those is described in section 4.3.3).

In order to properly integrate the calculation of the suggestions into the graphical editor, the advantages of both *reactive* (i.e., displaying suggestions after an explicit request) and *proactive* approaches (i.e., displaying suggestions automatically and continuously) were discussed. It was decided to implement a *reactive* approach, which is already a common concept in the editor (e.g., errors and warnings are only shown after pressing the according button to check it) that has several advantages such as the avoidance of user distraction and high computational effort (cf. [Pei21]).

Depending on the implementation of the graphical editor, this function has to be integrated into a suitable place so that it is not disruptive but still easy to find, as will be described in section 5.3.3.

Support the Modeling Task

Within the evaluation of the initial implementation of the graphical editor, it also became apparent that lecturers have to be supported during modeling. Therefore, *general concepts to support the modeling* as well as the concept of *reminder messages* are investigated that pursue the following goals:

- Ease the process to add and parameterize elements, as well as
- provide hints regarding modeling errors or available templates.

The concepts for supporting the modeling were developed together with Lidia Roszko and Niclas Zellerhoff during their practical courses.

In order to support lecturers during the modeling, the following utility functions have to be added:

- The addition of *snap lines* as well as a *grid* in the background to properly align elements,
- the *automatic connection* of elements following the left-to-right metaphor,
- the *centering and properly zooming* of the scenario,
- and the *automatic parameterization* based on previous elements.

Furthermore, the concept of *reminder messages* is investigated. *Reminder messages* represent hints that are displayed in specific situations, i.e., if one or multiple preconditions match. Based on the role concept, lecturers obtain different roles during the modeling task. According to their current role, certain reminder messages are displayed, e.g., a *Novice* receives different messages than an *Expert* who uses the editor regularly.

These messages are shown as toast messages directly in the editor, and pressing on them results in an appropriate action to be executed. For example, if lecturers model incorrect or incomplete teaching scenarios, a *reminder message* could provide the hint that the model contains errors and pressing on it would mark all invalid elements (cf. NFR2).

Support Complex Scenarios

While modeling simple teaching scenarios is possible without much effort and is supported by the previously presented concepts, the modeling of complex teaching scenarios turns out to be very time-consuming, as every element has to be modeled. Therefore, different supporting concepts are presented that intend to decrease redundancy:

- Enable saving and exporting of templates,
- sharing of templates,
- providing used-practice templates based on existing scenarios and templates,
- loading of templates,
- modifying and integrating templates, as well as
- propose suitable templates.

The concepts for supporting the modeling of complex scenarios were developed together with Sinthujan Thanabalasingam, Robert Peine and Chang Hong during their master theses.

A common functionality in modeling tools is the *support of saving and exporting templates* that can be reused later. This avoids redundancy when creating similar models multiple times and thus, has the potential to increase the speed of the creation of complex scenarios. When saving a template, our concept provides the options to set a name, a description as well as an arbitrary amount of categories. All of this information can later be used to retrieve this template more easily. A similar function is provided by exporting templates – however, those are exported to the file system. Therefore, also a name can be set that is later used as a file name. However, both the description and the categories can be omitted. The provided file types should include a graphic type (e.g., PNG or even better SVG), as well as an exchange format in which the scenario and its structure are stored. Moreover, for both dialogs, two options were investigated – an option to convert specific functional block types (e.g., *SingleChoiceLearningQuestion*) to abstract types (e.g., *LearningQuestion*), as well as the reset of all defined values for element parameters (e.g., a *questionText* would be reset). A mockup of the save dialog to store templates on the server is depicted in figure 4.10. While lecturers store their teaching scenarios as *private templates*, the administrator of the system will be able to create *public templates* that are visible to and can be used by everyone.

Save Template
Export Template to File System

Name
Peer Instruction

Description
Peer instruction is an evidence-based, interactive teaching method popularized by Harvard Professor Eric Mazur in the early 1990s. Originally used in many schools, including introductory undergraduate physics classes at Harvard University, peer instruction is used in various disciplines and institutions around the globe.

Preview

Save Options

Categories
Introduction x Recap x Test x

Function Blocks
☐ Reset all function block attributes
☐ Convert all function blocks to their abstract type

Save

Figure 4.10: The concept of a save dialog to store a template on the server that can later be filtered by its name, description or category [Tha21].

After creating and saving (or exporting) teaching scenarios as a template, lecturers are often willing to exchange their ideas. Therefore, the *sharing of templates* should be enabled. While the exchange of templates to known lecturers is possible by sending them the exported file, a *public template catalog* should be investigated, in which lecturers can share their teaching scenarios with the stARS community (i.e., user-generated templates). Those are later allowed to rate the templates in order to identify best practices. All of these provided templates can be loaded by any lecturer in his/her personal teaching scenario.

As not every lecturer will be aware of these templates or want to share his/her templates, another conceptual idea is to *extract used practice templates* from the multitude of existing

scenarios (or *private templates*) that was investigated by Chang Hong in her master thesis [Hon21]. This function should be available to the administrator of the system, e.g., in order to adjust the *public templates* to provide lecturers with (parts of) templates that other lecturers prefer to use. In addition to the appropriate comparison of two models (that might include circles) and the recognition of similar structures, a major challenge is the increasing number of scenarios, which requires more comparisons to be made and more templates to be generated than might be suitable. Thus, not only the number of scenarios to be compared has to be reduced (e.g., by defining a date range), but it must also be possible to filter templates in order to detect more popular ones. Further details on this concept can be retrieved in [Hon21].

After discussing different concepts to store or retrieve templates, they need to be loaded properly into the graphical editor. Therefore, a *load/import function* is investigated that either loads a template from existing private templates, public templates or the template store, or imports it from the file system. Afterward, two options will be provided: On one side, the *template can be loaded into the existing scenario*. On the other side, the entire *scenario can be replaced* by it. The suitability of these approaches depends on both the situation and the template itself. The conceptual idea of the *load/import* dialog is depicted in figure 4.11.

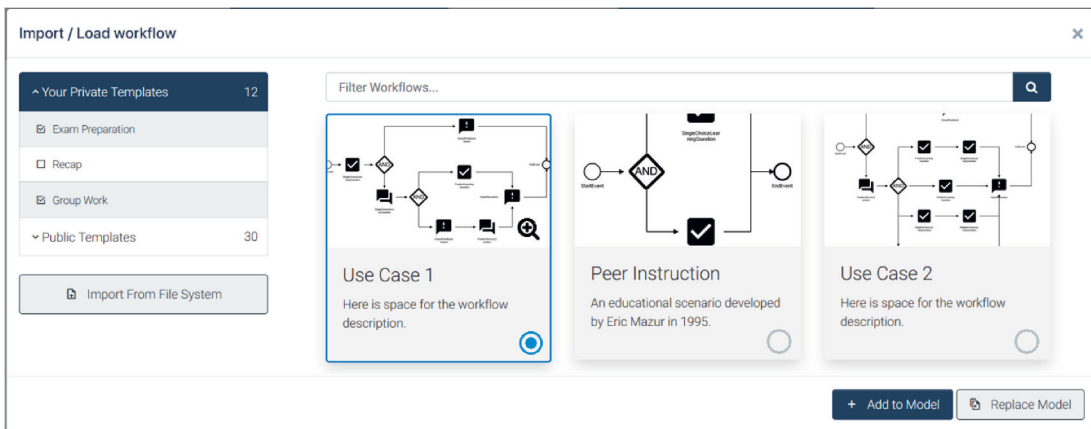


Figure 4.11: The concept of a load/import dialog that enables to either load templates into the existing scenario or replace them entirely [Tha21].

When choosing to *add the template to the existing scenario*, it should be visualized in a separate container. This will help to identify the loaded component but also ease the editing of it. Therefore, the concept proposes the idea of a separate *tab visualization* for every sub-container, in which it can be adjusted freely. However, these sub-containers can also be removed by *integrating* them into the existing scenario. Another idea is to

give lecturers the ability to *swap* the entire sub-container with another template. The general concept to interact with added templates is depicted in figure 4.12.

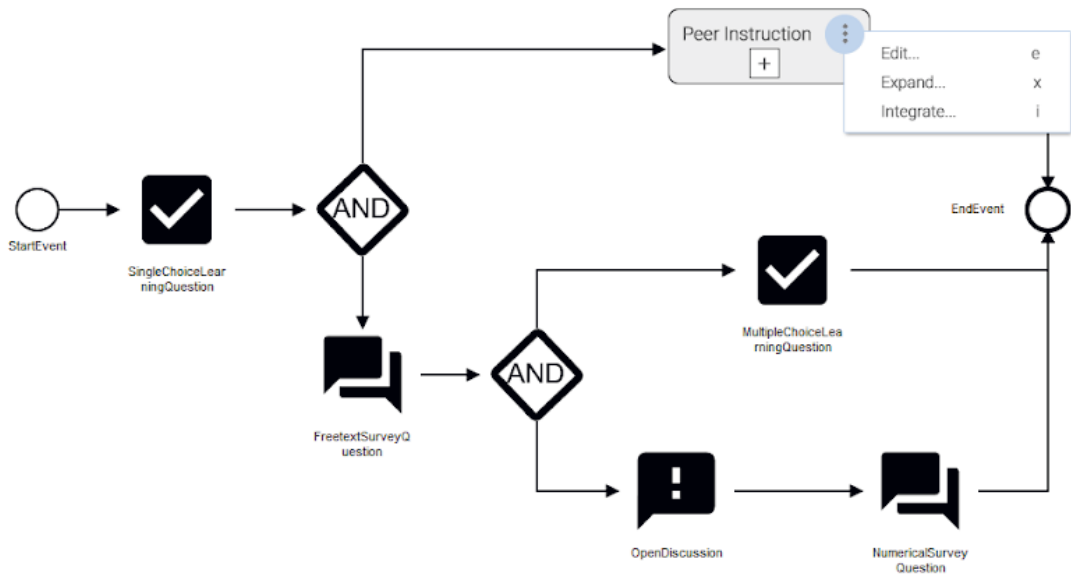


Figure 4.12: The concept of visualizing added templates as separate containers that can later be edited, integrated or swapped [Tha21].

As an extension for the visualization, an option should be provided to group multiple elements of the currently loaded scenario into one container. This will later help to identify elements that behave similarly, e.g., three parallel learning questions. Moreover, the modeling of complex scenarios can also be supported by the previously mentioned function to propose useful elements or templates. E.g., when modeling a scenario consisting of multiple question rounds, the system could propose different representations of a question round as retrieved and positively rated from the template catalog.

Overall, we strongly believe that the presented supporting concepts are able to ease the modeling process for the lecturers drastically. Using these concept extensions, lecturers should be able to model their individual scenarios without having any special prior knowledge in modeling, which is a key requirement of our approach (cf. *NFR1*).

Note on the Adaptation at Runtime

As motivated on the use case *Learners-as-Designers* in subsection 2.1.2, teaching scenarios are not always fixed workflows. Rather, they might evolve dynamically at runtime, which requires adapting those at runtime. The concept for adapting workflows at runtime will be described in section 4.6.

4.4 Runtime Environment

After describing the graphical editor that is able to model individual teaching scenarios (called *application models*), this section introduces a *runtime environment* that can interpret those and execute the customized functionality (cf. FR7 and FR8). Therefore, requirements for the *runtime environment* are discussed, and an appropriate infrastructure is presented. Finally, a concept for adaptation at runtime is presented.

First, the system has to enable *platform-independent access* in order to ensure that the broad audience can participate. This is especially important as our approach follows the BYOD¹³ paragon, and the student's used operating system should not be a barrier that excludes them from participating in the system. Therefore, a variety of approaches exist, with the best known probably being web-based applications that run independently of the operating system of the student devices in any web browser and sometimes even feel like native applications. They use interfaces such as REST to communicate with a backend or runtime-cloud, which offers the concrete functionality of the system, e.g., the authentication or the storage of answers to a question.

Next, we target to create a *standalone* solution that does not depend on another system (e.g., a *Learning Management System (LMS)*). This is motivated by the research of [Sch16], in which an earlier system called *MobileQuiz*¹⁴ was developed that was embedded in the LMS ILIAS¹⁵. While embedding has advantages in later use, as the solution is directly integrated with the LMS, it also creates a significant challenge: Other universities or chairs that are trying to use the system have to use the same LMS with the identical configuration, which is rarely the case. As a result, the next version of *MobileQuiz* was also implemented as a standalone solution [Sch16].

Furthermore, the system must be *scalable*, e.g., to handle large lectures such as those taking place in the Audimax of the TU Dresden with 892 seats (cf. NFR4). While the previously motivated separation between frontend and backend or runtime-cloud already ensures that no websites have to be generated on the server-side and thus enables efficient communication, it should furthermore be possible to improve the performance by adding further runtime-clouds. This requirement is closely related to the requirement of *independently running instances*, i.e., the

¹³ Bring your own device: Students use their own mobile devices to participate actively.

¹⁴ <https://github.com/dschoen/mobilequiz> – last successful access on October 8, 2021

¹⁵ <https://www.ilias.de/en/> – last successful access on October 8, 2021

execution of an instance (i.e., a teaching scenario) should not affect the execution of another instance, which can be realized by running them in different virtual containers (e.g., Docker-containers).

Finally, the system must allow for *changes at runtime* (cf. FR9), e.g., to spontaneously add a question to the currently running *application model*. This is especially important for scenarios in which no fixed workflow can be defined. Even though *DecisionForks* exist that allow deciding manually on the subsequent path, the progression of a lecture often depends on real-time results. For example, if a student holds a presentation, the lecturer may want to create questions (to all students) that were not yet answered.

Having these requirements in mind, we propose the concept of a fully scalable 3-layer architecture that consists of a *frontend*, a *backend* and a *runtime-cloud*, as depicted in figure 4.13. In the following subsections, each component is further described.

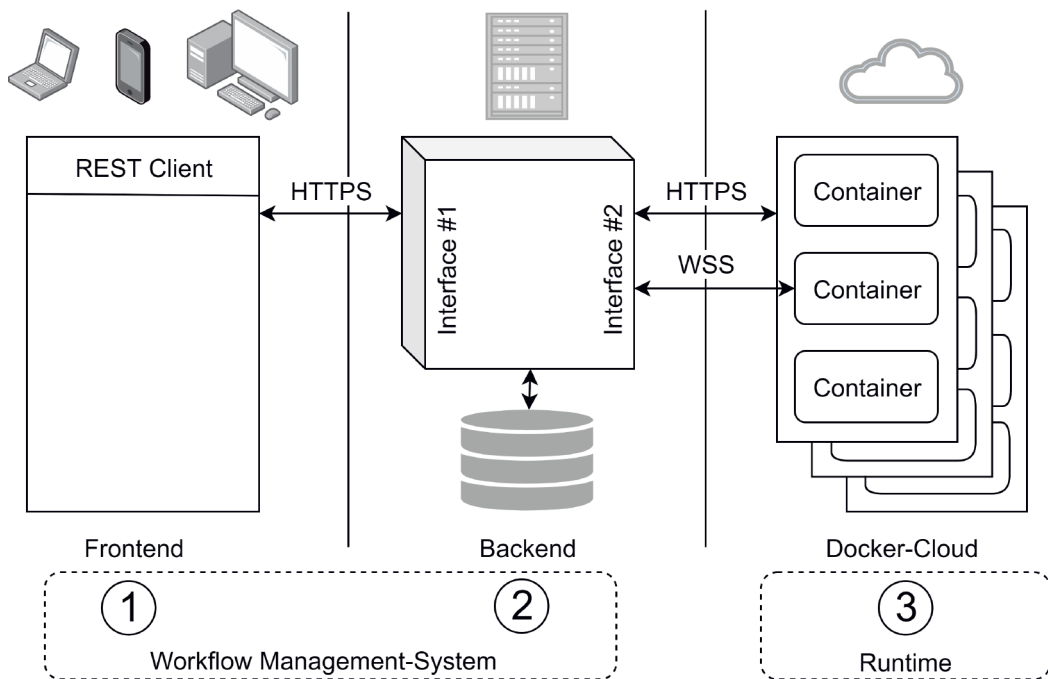


Figure 4.13: The concept of an infrastructure for an adaptable collaborative learning environment [KSS19] (the communication between the individual components was slightly adjusted in the implementation (cf. section 5.4)).

4.4.1 Frontend

The *frontend* provides the user interface for *students*, *lecturers* as well as the *administrator* of the system. As we target to create a web-based solution (to ensure platform-independent access), a common strategy is to separate the *frontend* from the *backend* or *runtime-cloud*. Therefore, the approach of *Single Page Applications (SPAs)* is used that allows to dynamically rewrite contents with new data instead of reloading the entire page. For instance, if the lecturer unlocks one or multiple questions, only the question view has to be refreshed, while the rest of the page remains unchanged. The same applies to the result view: If new answers were added, we only need to refresh the current diagram. This is also useful for realizing different views for different roles while relying on one shared codebase. Instead of implementing a view for every user role, views can be dynamically composed of a variety of components, while some of them are shared between different user roles (e.g., the authentication, global error messages, or even functions such as the presentation of results that should be visible for the lecturer, but also for the students, if the currently active block is of the type *PresentResult*). Using the concept of *lazy loading* (also called on-demand loading) furthermore allows loading and rendering only the required components instead of the entire website [Ish18]. This technique is also useful when having different user roles, as not every role has to load all components, e.g., the students do not need to load the component of the graphical editor. In the following, the main functions of the *frontend* are summarized:

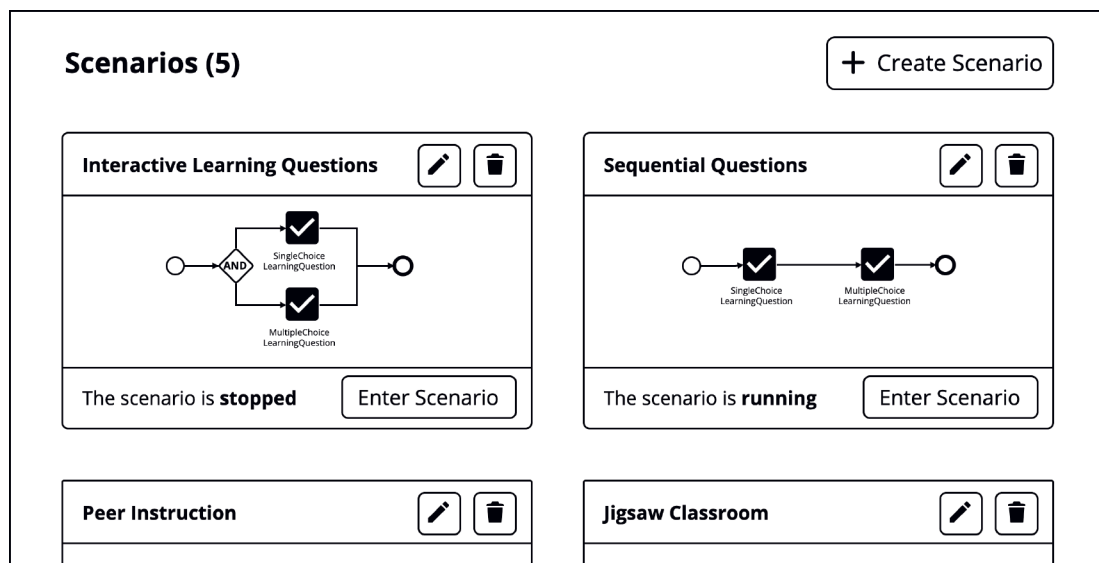


Figure 4.14: Mockup of the scenario overview with the abilities to create new scenarios and edit or delete existing ones.

- Authentication for different user roles based on a username and password, or even providing anonymous participation, if defined.
- The creation and modification of *application models* using the *graphical editor* as well as displaying an overview of existing *application models* with the ability to delete those (cf. figure 4.14).
- The import and export of *application models* in order to reuse or share them.
- The capability to start *application models* that are instantiated as dedicated containers in the *runtime-cloud*. Moreover, the ability to pause, resume and stop (i.e., reset) them.
- The management of the progression in the *application model*, i.e., finishing currently active blocks (cf. figure 4.15).

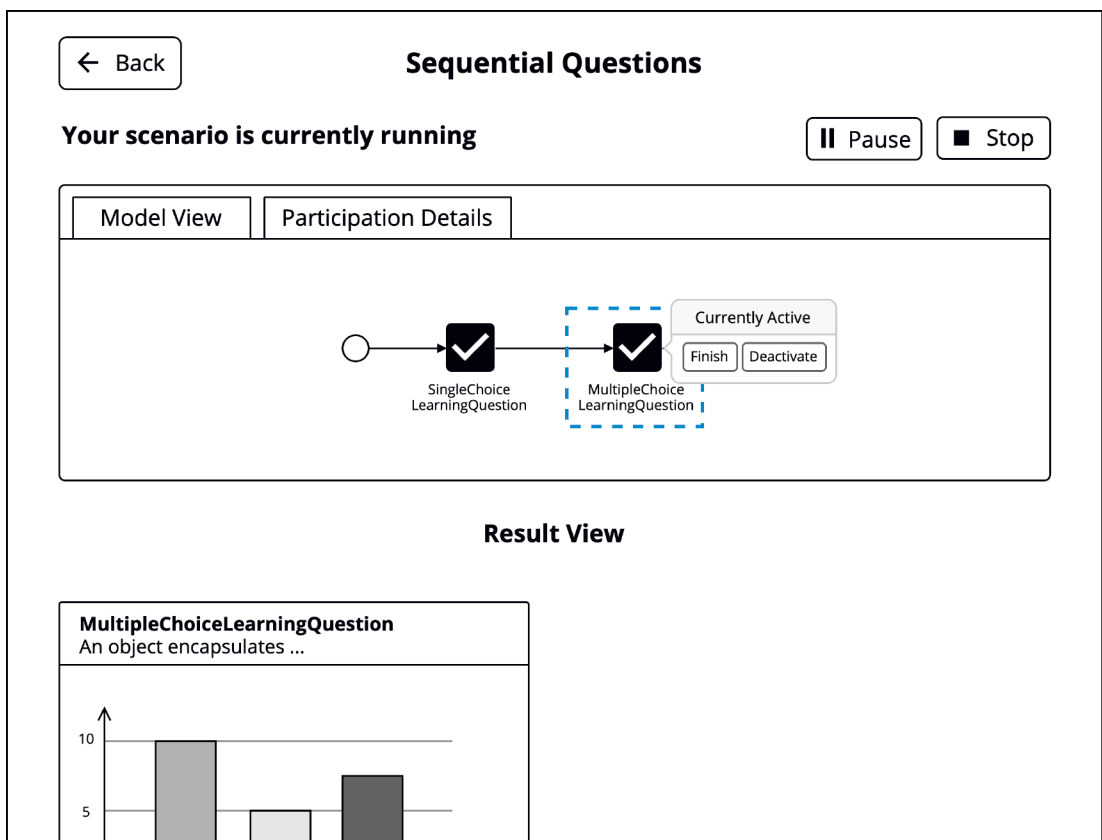


Figure 4.15: Mockup of the scenario management with the abilities to start, pause, resume and stop (i.e., reset) instances, as well as manage the workflow by finishing blocks.

The mockup shows a web interface titled "Sequential Questions" with a "Student View" subtitle. At the top left is a "Back" button with a left arrow. Below the title is a large rectangular box containing the question "An object encapsulates ...". Underneath the question are three radio button options: "Data", "Behavior", and "State". At the bottom of this box is an "Answer" button.

Figure 4.16: Mockup of the student view with the ability to interact with the currently active blocks. In this example, a *MultipleChoiceLearningQuestion* can be answered.

- The participation of students in the executed *application model*, e.g., allow answering active questions or interact in groups, as visualized in figure 4.16.
- The display of results to currently active blocks (cf. figure 4.15) as well as the usage of control options (e.g., the assignment of moderators in the *GroupVoting*).
- The export of results after finishing an *application model*.
- The administrator should have full access to manage both instances and users.

4.4.2 Backend

The *backend* serves as a gateway for different types of communication between the *runtime-cloud*, the *database*, and the *frontend*. When creating an *application model* through the graphical editor, it is stored on the *backend* server, from which it can be started by the lecturer. During the start, a container is created on one of the *runtime-cloud* servers that were configured by the administrator of the system. This container is initialized with the respective *application model*. As a variety of instances can run on one *runtime-cloud* server, the *backend* manages the information about which instances run on which *runtime-cloud*, how they can be contacted, and which users can access them. Moreover, it can move containers between different *runtime-cloud* servers without a noticeable interruption in

order to protect specific *runtime-cloud* servers against overload. Furthermore, the *backend* server is responsible for the authentication of the users and issues tokens that are used for the authentication against the individual instance in all subsequent interactions. Finally, WebSocket connections to each student are established to inform them about changes in the execution of *application models*, e.g., when starting or pausing them. The main functions of the *backend* are summarized as follows:

- Provide a scalable database.
- Authenticate users in the *backend* or on the *runtime-clouds*.
- Create instances of *application models* on one of the configured *runtime-cloud* servers.
- Manage instances of *application models* and, if required, move them to another *runtime-cloud* server.
- Manage the information about the instances, e.g., how to contact them, or the decision which users can access them.
- Hold WebSocket connections to all students to inform them when an instance is started, paused, resumed, or stopped (i.e., reset).

4.4.3 Runtime-Cloud

In contrast to the *backend*, the instances that run on the *runtime-cloud* servers contain the actual functionality, which was specified by the (meta-)model. They are able to understand the *application model* that was previously created and allow navigating through it. Students are presented with the currently active blocks and can interact with them, while the lecturer is able to retrieve the results in real-time and can use further control options, as described in the *frontend*. While the communication takes place largely via a REST API, WebSocket connections to each student and the lecturer are established to inform them about changes in the workflow, new results, or real-time communication, as it takes place, for instance, in the *GroupChat* between the group members. The main functions of the *runtime-cloud* are summarized in the following:

- Interpreting an *application model* and allow navigating through the blocks.
- Inform students about the currently active blocks and allow them to interact with the functionality.

- Provide temporary storage for the *runtime data* such as the *user data* or the *answer data*.
- Allow the lecturer to retrieve the results of the currently active blocks in real-time.
- Push data from the temporary storage back to the *backend* server.

Note on the Role Concept

While most of the functionality is directly defined by the (meta-)model, there exist several functions that cannot be expressed by the (meta-)model and thus, have to be included at runtime, e.g., the assignment of users to specific groups, or the addition of privileges when being assigned as a moderator for an *OpenDiscussion*. Therefore, the following section will elaborate on the *concept of roles* to be integrated into the *runtime* of our approach, which could make it much more flexible despite the statically activated blocks.

4.5 Integration of Role Concepts

The concept for integrating roles was developed together with Robert Peine during his time as a student research assistant and partly published in [KPB20].

The concept of roles has been proven to be a suitable paradigm when creating context-dependent and collaborative systems. For example, [Wut18] was able to show that it represents an intuitive abstraction to model *distributed systems* at design time and make them adaptive as runtime. Therefore, the example of smart service systems was used. Although the context is different from our context (i.e., *digital learning environments*), the general observations should also be true for this thesis, meaning that roles might provide a suitable extension to model collaborative scenarios (e.g., a group collaboration) and add means of adaptation at runtime. A more detailed elaboration will be done within the course of this section.

4.5.1 The Importance of Roles in Learning Environments

Integrating the concept of roles for developing an adaptable collaborative learning environment is motivated by the fact that roles are inherently defined in every *digital learning environment*. For example, common systems distinguish between the *lecturer*

who creates and unlocks interactive activities and *students* who are using those to participate more actively within the ongoing lecture. Of course, plenty of further user roles might exist, such as an *administrator* that manages the users or an *assistant* that supports the *lecturer* in creating content or unlocking activities.

In addition to these global user roles, there also exist roles that are specific to individual interactive activities. An example was presented in subsection 3.2.2 with *Tweedback's* Chatwall, in which a moderated mode exists. If this option is selected, the user of the role *moderator* is able to unlock posts, which will reduce potential spam. Another example can be found in group tasks, in which students find themselves in the role of a *group member*. However, further roles can exist, e.g., each student could receive another role when solving a task cooperatively, or a student could be assigned to be the *leader* (or *moderator*) of the group in order to organize the group task or select a unified answer for the group.

When looking from the modeling point of view, interactive activities themselves can also have different roles. For example, depending on a certain teaching context, a *multiple-choice question* could behave differently, i.e., the role of the question is changed. While *multiple-choice questions* are used to repeat the content on themselves when being placed in front of the lecture (i.e., the role of a *PreparationQuestion* is assigned), they are used to check students' gained knowledge and discuss it when being placed during the ongoing lecture (i.e., the role of a *LearningQuestion* is assigned). Another example would be an *OpenDiscussion*. While students should discuss these questions in the background of the lecture by themselves when the lecture is held (i.e., *BackgroundDiscussion*), it could also be used actively (i.e., *ClasswideDiscussion*).

In summary, the concept of roles seems to be an intuitive extension when developing *digital learning environments* and, more specifically, when integrating the means of adaptation and collaboration to them. In the following subsections, the addition of the role concept within the modeling as well as the runtime is discussed in further detail.

Note on the Implementation

The investigations described in the following subsections are used as a foundation for integrating role concepts during the implementation (cf. section 5.5).

4.5.2 Role-based Modeling of an Adaptable Collaborative Learning Environment

Role-based (Meta-)Model

Although the role concept has been proposed as an intuitive approach to model complex and dynamic domains, our (meta-)model is not directly founded on it. This is caused by various reasons that will be discussed in the following. The first reason(s) got obvious during early investigations on using the role model to express the variety of didactic scenarios (e.g., *Peer Instruction*): The idea was to represent each didactic scenario as a *compartment* that includes its different stages as different roles that are played by the specific functionality. Moreover, the functionality itself could include different roles to change its behavior, as visualized in the didactic scenario of *Peer Instruction* in figure 4.17.

However, different problems were observed: We quickly realized that mapping the variety of didactic scenarios with this approach is not only quite challenging due to the large majority of existing scenarios, it is moreover impossible because a didactic scenario is not a fixed scheme that always runs the same way. For example, for *Peer Instruction*, a majority of different kinds of sequences exist that distinguish even in their actual stages. While some lecturers use it to repeat and test contents that were prepared individually, others use it to check contents that were presented within the lecture. Moreover, the proceeding after the *peer discussion* is handled quite differently.

Even if it would be possible to create a role model that is able to express the majority of existing didactic scenarios, further issues would arise when lecturers want to customize it to their individual preferences. Therefore, they would have to be able to adapt the model, which would have led to similar problems as described in *MobileQuiz2*, which were caused by the model's size and incomprehensibility (cf. *NFR1*).

Consequently, for the modeling adaptation itself, another approach was selected (i.e., a model that uses ideas derived from workflows) that allows creating customized sequences of different functions. Each function can be further adapted by defining different parameters. This approach is targeted to be also understandable for lecturers without prior knowledge in modeling (cf. subsection 4.2.6), which would be a challenge when using the concept of roles for modeling individual scenarios. However, the chosen approach has to be evaluated separately later on.

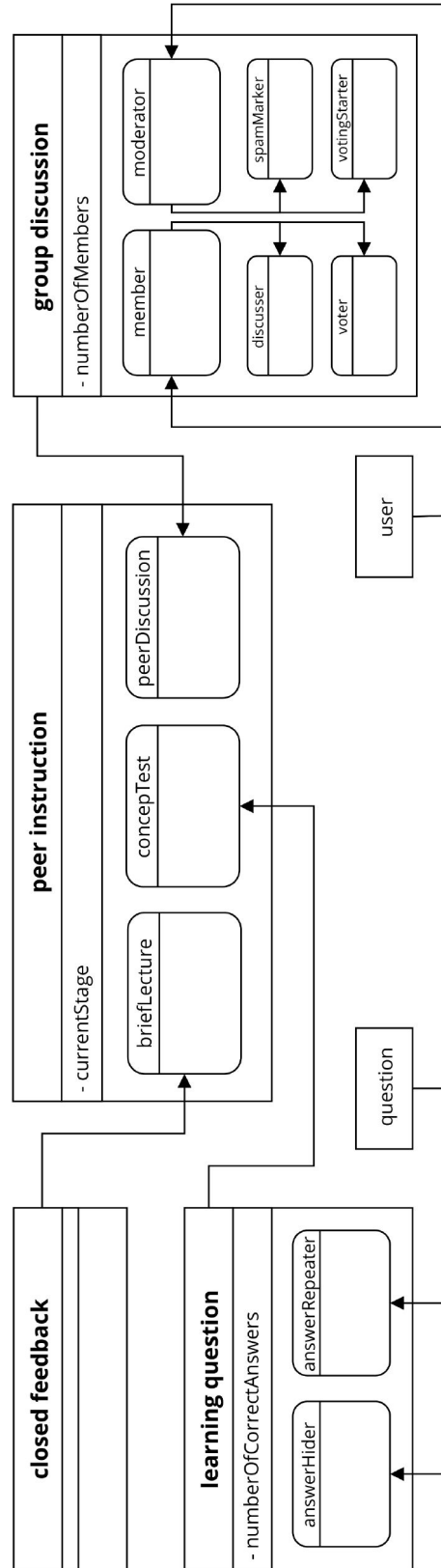


Figure 4.17: An exemplary role model for the didactic scenario of *Peer Instruction*.

Role-based Modeling to Decrease the Complexity of the Runtime

An advantage of role-based modeling can be found when expressing multiple classifications: Instead of modeling every sub-type separately, the characteristics could be represented as roles, which reduces the number of classes and, thus, the complexity of the runtime.

A similar approach was investigated for several components of our created (meta-)model. For example, the classes *LearningQuestion*, *SurveyQuestion* and the sub-types, such as *SingleChoiceLearningQuestion* or *SingleChoiceSurveyQuestion*, could be realized by roles, as is visualized in figure 4.18.

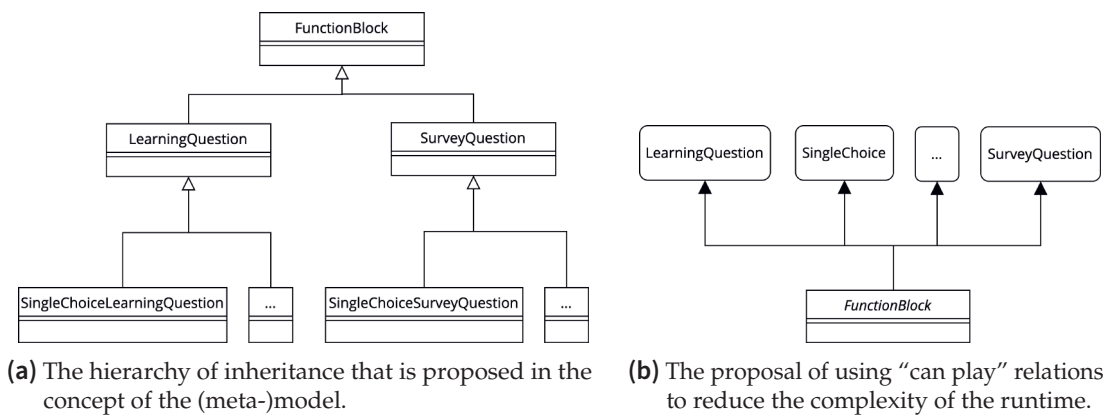


Figure 4.18: A visualization of possible advantages of role-based modeling when implementing our (meta-)model.

While this might be a suitable implementation at first, a critical problem becomes obvious when investigating the parameters that were proposed in the (meta-)model. For example, *SingleChoiceLearningQuestion* defines the reference *choices*, while *SingleChoiceSurveyQuestion* additionally has a parameter *showAggregate*. This is similar for other types, such as *FreetextLearningQuestion* that has two additional parameters (*correctText* and *caseSensitive*) that *FreetextSurveyQuestion* has not. As those parameters are specific for the sub-type, it would be necessary to add them (i.e., to the role *SingleChoice* or *Freetext*). However, this would result in problems as the previously defined parameters would not be reproducible anymore without having unused parameters.

For this reason, the role concept was not used for decreasing the complexity of the runtime, or more specifically, not within the modeling part of it.

Role-based Concepts as a Metaphor during Modeling

As motivated in the introduction of this section (i.e., the roles *PreparationQuestion* and *LearningQuestion*), as well as in subsection 4.3.3 when talking about the *reminder messages*, roles could provide a suitable metaphor that can be used to support the modeling task that is executed by lecturers. On the one hand, roles could be understood as presets (i.e., a predefined configuration of the parameters) to serve specific use cases. For example, when selecting the role *PreparationQuestion* for a *SingleChoiceLearningQuestion*, *answerFeedback* would be set to *true* in order to give students feedback whether they understood the concept or not. Furthermore, the parameter *numberOfRepetitions* could be set to 2, which allows students to answer this question two times to rethink the question after answering it incorrectly. A variety of further presets could be added for every *functional block* that allow serving specific situations. On the other hand, roles could be used to describe the current stage of the modeling process, i.e., lecturers could obtain different roles when advancing in the modeling process or when already using the system for quite a while. Depending on this stage, different *reminder messages* are displayed (cf. subsection 4.3.3).

Although the term *role* might be a suitable extension in this context, the *implementation* has to check whether this notion presents advantages or just serves as a suitable metaphor to describe these situations (cf. section 5.5).

4.5.3 Role-based Runtime for an Adaptable Collaborative Learning Environment

Although we decided not to integrate the concept of roles for most of the modeling tasks, in this section, we will show the importance of using roles to separate functions that are added at runtime from the actual model¹⁶. Therefore, after discussing general role terms (cf. subsection 2.2.1) in the context of our concept, different examples are presented in which the role concept presents a suitable extension for the runtime. This section ends by summarizing role features that are supported by different role types.

The implementation of a role-based runtime is described in section 5.5.

¹⁶ A similar but more advanced approach was presented and evaluated by [Sch20].

Fundamentals for Integrating Roles in the Concept

In our concept, lecturers have the ability to create customized workflows of interactive activities that can be further adapted by defining values for different parameters. Although the (meta-)model allows to precisely describe the functionality of these activities (i.e., *functional blocks*), there also exist functions that have to be added at runtime, e.g., the assignment of students to specific groups or the assignment of an assistant to the scenario that should retrieve the privileges of the lecturer. As a solution, we plan to integrate the role concept to describe these situations more intuitively.

An important component when integrating roles is the definition of a suitable *compartment*. In our concept of an adaptable collaborative learning environment, the *compartment* could either be defined by the scenario itself (i.e., the representation of the entire lecture) or by a specific *functional block* (i.e., an interactive activity) in a specific iteration. When a *functional block* gets active, the roles that were defined for this *compartment* can be played in order to add runtime-specific functionality that cannot be expressed by the (meta-)model itself.

The *natural type* (i.e., the *player*) of our role extension is always a *user* of the system. As described in the previous subsection, we decided against allowing roles to be played by other objects, such as *functional blocks*. Consequently, each role is assigned to either a *student* or a *lecturer* of the system.

Moreover, a variety of *role types* will be integrated. In the following, different use cases are discussed in further detail.

Possible Role Types to be Integrated into the Concept

In our concept, a *GroupBuilder* is added that allows forming groups based on a selected *buildSchema*, as well as either a defined *groupSize* or a *numberOfGroups*. However, the assignment of students to specific groups cannot be expressed directly by the (meta-)model and has to be added at runtime. Therefore, the role concept could be used twofold: First, students could be assigned a *GroupBuilderWaitingUserRole*, if they acknowledged joining the group task and are waiting for them to start. Second, students with an assigned *GroupBuilderWaitingUserRole* could be used to create groups of students, i.e., by assigning them a role *GroupMember*, in which its current group is defined, and abandon the *GroupBuilderWaitingUserRole*.

Another role type can be found in the *GroupVoting*, which allows selecting a unified answer as a group. However, if the group members cannot or do not want to agree on an answer, a *GroupVotingModerator* can be assigned to one of the group members that is then allowed to select the group's answer.

Another group of *functional blocks* that benefits from integrating role types is described by *Peer Interactions*, which enable to support peer feedback. In comparison to *Group Interactions*, a student can have multiple assignments for providing or receiving peer feedback – however, a student *A* that provides feedback to a student *B* does not automatically receive feedback from *B*. The role concept is again used twofold: First, to assign a role after acknowledging the participation in the peer task (*PeerBuilderWaitingUserRole*) and second, to either express the assignment as a *PeerFeedbackProvider* (i.e., the student has to provide peer feedback) or as a *PeerFeedbackReceiver* (i.e., the student receives peer feedback).

In addition to role types that are added to novel *functional blocks* introduced in this thesis, the concept of roles is also a suitable extension for traditional functions. For example, in the *OpenDiscussion*, an *OpenDiscussionModerator* could be assigned that is able to delete or unlock posts (similar to *Tweedback*'s moderated *Chatwall*) depending on the chosen configuration of this *functional block*.

Furthermore, roles can also be used to restrict access to specific users (cf. role feature <12> from table 2.1 that was presented in subsection 2.2.1). Therefore, the parameter *filter* can be implemented using a *FilterRole*. Only students that have been assigned this *FilterRole* are able to access the *functional block*.

While all previously described role types are assigned to the *compartment* of a *functional block*, it should also be possible to define role types for the whole scenario. An example of this is the role of a *Lecturer* that allows users to manage the workflow during the lecture, i.e., unlocking the interactive activities. There may exist multiple users that have been assigned this role, e.g., the lecturer itself as well as his/her assistant.

In conclusion, there is a large potential for integrating the concept of roles to describe functions added at runtime, which frequently occur in *digital learning environments*, e.g., when dynamically assigning students to groups or managing user roles for specific students. This does not only allow adding functionality that cannot be expressed by the (meta-)model (e.g., if the assignment of students to groups should be based on real-time results), but also increases the flexibility that exists when using individual functional blocks (e.g., the assignment of group moderators can be triggered, if a *GroupVoting* should proceed to its end).

Features of Roles that are Supported by these Role Types

In this section, different features of roles (cf. table 2.1 in subsection 2.2.1) are discussed that are used by the previously presented role types. Therefore, potential scenarios for selected role features are described.

<1> Roles have properties and behaviors.

An *OpenDiscussionModerator* is able to unlock posts that are stored in a *history*.

<2> Roles depend on relationships.

A *user* that can manage the workflow and evaluate the results has the role of a *Lecturer*.

<3> Objects may play different roles simultaneously.

If an *OpenDiscussion* and a *GroupBuilder* are active in parallel, a user can play the role of an *OpenDiscussionModerator* as well as of a *GroupBuilderWaitingUserRole*.

<4> Objects may play the same role (type) several times.

A user can play the role of a *GroupMember* for different group tasks that can even be active in parallel.

<5> Objects may acquire and abandon roles dynamically.

A user can be assigned to the role of a *GroupVotingModerator*, which is abandoned after finishing the *GroupVoting* block.

<6> The sequence of role acquisition and removal may be restricted.

A user can only be assigned to be a *GroupMember* if he/she was previously in a *GroupBuilderWaitingUserRole*.

<9> Roles can be transferred between objects.

A user that is assigned the role of an *OpenDiscussionModerator* could transfer this role to another user.

<10> The state of an object can be role-specific.

A user that is assigned the role of an *OpenDiscussionModerator* holds a history of unlocked posts.

<11> Features of an object can be role-specific.

A user that is assigned the role of a *Lecturer* can finish blocks of the workflows.

<12> Roles restrict access.

If a filter for a *functional block* is defined, only users that have assigned the *FilterRole* can access it.

<16> Relationships between roles can be constrained.

A user can only play the role of a *GroupVotingModerator* if he/she is previously in the role of a *GroupMember*.

<19> Roles depend on compartments.

Every of the above-listed role types depends on a *compartment*, e.g., the *GroupVotingModerator* depends on a *functional block* of the type *GroupVoting*.

<20> Compartments have properties and behaviors.

As *compartments* can be represented by *functional blocks*, they have both properties and behaviors.

<21> A role can be part of several compartments.

As *compartments* in our context are, for instance, defined by a specific *functional block* in a specific *iteration*, there exist different *compartments* for *functional blocks* of the same type that are active in parallel. Thus, the same role is part of several compartments. In addition, there also exist roles that are part of different types of *functional blocks* (i.e., *compartments*), e.g., the *FilterRole* can be defined for every *functional block*.

<25> Different compartments may share structure and behavior.

For example, both *SingleChoiceLearningQuestion* and *MultipleChoiceLearningQuestion* inherit from *LearningQuestion*.

<27> The number of roles occurring in a compartment can be constrained.

For instance, the number of *Lecturer* roles could be restricted to 2 if only the lecturer itself and one assistant should be able to manage the workflow.

By investigating example use cases, we could already motivate that our approach will support role features of all three natures of roles (i.e., behavioral, relational and context-dependent). However, this has to be further elaborated on in section 6.5.

4.6 Adapting Scenarios at Runtime

Representing a lecture and its activities as a static workflow is not suitable for every use case. Often teaching scenarios are very dynamic in their design: For example, the progress in a lecture may depend on answers given by students on a previous learning question. While the concept described so far makes it possible to model different paths that can either be selected automatically based on defined rules (*OrFork*) or manually (*DecisionFork*), the extension or modification of currently running (i.e., started) scenarios is not possible. This makes it impossible, for example, to add blocks (such as survey questions) spontaneously or to realize highly dynamic use cases, in which students' results are used to develop the scenario itself (and its contents).

In order to integrate this functionality in the concept, four requirements are defined:

1. The functionality should allow *adapting the scenario* in a suitable way, e.g., in order to add new blocks to the running scenario.
2. During the adaptation, the *validity* must be guaranteed, i.e., the scenario should never become invalid.
3. Moreover, the lecturer should *feel secure* when using this functionality, i.e., he/she should not be afraid that he/she might destroy the running scenario.
4. Finally, it should also be possible to *undo* a modification that has been made.

As both *validity* and *lecturers feeling secure* are crucial requirements when adapting the scenario, it should be avoided to modify the entire scenario in an unrestricted way. Instead, the functionality should be designed as minimalistic as possible and later be extended, if required. Thus, the general idea is to reduce the adaptation of scenarios to simple extensions (that are made next to a selected element). Not only does this ensure that the scenario is still valid (if the extension is valid as well), but it might also be rated as less critical by lecturers.

Moreover, this functionality is expected to be expressive enough to handle the demands of lecturers (cf. RT2.3). However, in order to present this functionality less prominent

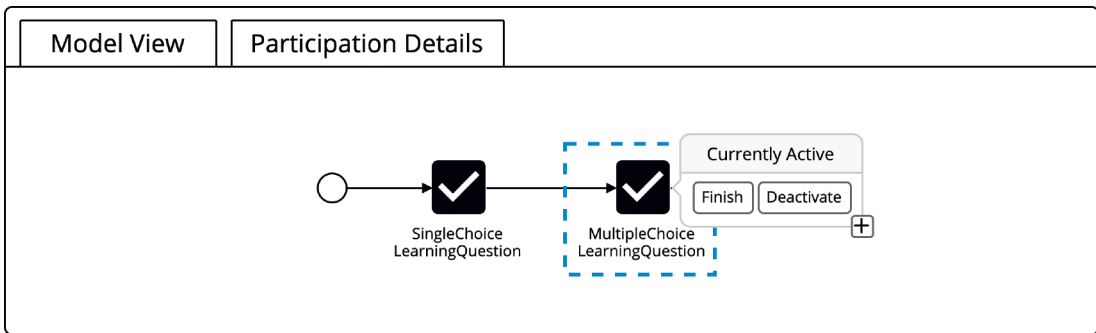


Figure 4.19: Mockup of the adjusted scenario management, in which the scenario can be extended next to a selected element when pressing the button with the plus icon.

for inexperienced users, only a small button with a plus icon is added on the bottom right of the block’s popover, as visualized in figure 4.19.

There exist three different options to extend the scenario:

- An *extension by a private template* allows adding groups of activities.
- An *extension by a public template* allows adding a general “best-practice” solution, such as a *one-minute paper* to assess the lecturer’s performance (cf. [Har96]).
- A *free extension* allows adding new (groups of) activities.

In order to increase the flexibility of extending scenarios by private or public templates, it should be possible to modify the parameters or even extend templates before adapting the scenario. Therefore, the selected template is loaded into the scenario editor, in which the extension is modeled. For the extension itself, a sub-container (as visualized in figure 4.12) is used that is automatically created next to the selected block and connected properly (while the remaining diagram is shifted accordingly) – similar to loading or importing a template in the scenario creation. Within this container, the extension is modeled, while the rest of the scenario remains unchangeable. However, parameters of blocks within the extension model can be referenced to blocks of the existing scenario.

When choosing to extend the scenario, a separate dialog is opened in which the entire model is previewed and its validity is checked. This step is important to ensure that lecturers feel even more secure when using the functionality. Moreover, a hint is displayed that the sub-container is integrated during extension. This is done in order to allow extending even the extensions that were made previously without having multiple sub-containers located into each other. The dialog is visualized in figure 4.20.

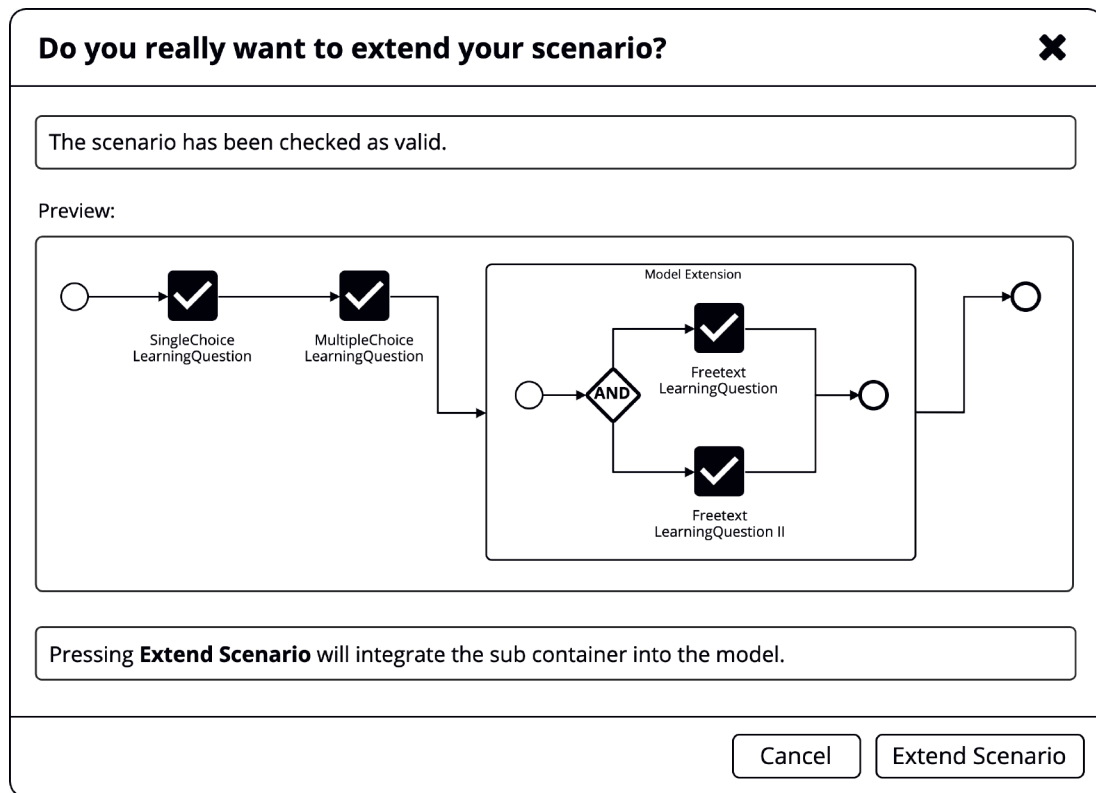


Figure 4.20: Mockup of the extension preview, which checks the model and gives hints.

After the extension of the scenario is finished, a button is displayed within the scenario management that allows undo it. This will ensure that an extension can be undone even after it is finished. Although our concept does not include an option to modify (i.e., adjust parameters or delete) blocks, these functions were not integrated intentionally, as they might overwhelm lecturers when both managing their scenarios and teaching in parallel. Moreover, those functions can be implemented indirectly: As all blocks can be activated and deactivated manually, deletion is not required, as those blocks can be skipped. Furthermore, the adjustment of existing blocks can be implemented by adding copies of them to the scenario extension and extending the parameters in those.

Note on the Evaluation

As this functionality was designed to be as minimalistic and simple as possible, the evaluation has to check whether it is expressive enough for lecturers to perform adaptations. Moreover, the concept of proposing scenario extensions was omitted (for the moment), as it is strongly influenced by the quality of proposals. However, those have to be investigated in the future, e.g., by analyzing existing scenarios.

4.7 Didactic Opportunities

This section explores the opportunities that are offered by our concept. Therefore, the general approach of using workflows as a representation of a technology-enhanced lecture will be discussed before looking at the addition of novel functional blocks and parameters (i.e., settings) and presenting references to suitable use cases.

Note on the Implementation of Didactic Scenarios

The implementation of the didactic scenarios that were introduced in subsection 2.1.2 will be presented in section 5.7.

While creating teaching scenarios as workflows might be a time-consuming task, it has several advantages: First, it helps to structure the entire lecture, as, in addition to interactive activities, the lecture itself can be described. Moreover, it helps to implement complex scenarios that include decisions to be made, which can be represented by either an *OrFork* (an automatic decision on the progression is made that depends on defined rules) or a *DecisionFork* (a manual decision on the progression is made). Using the parameters *timeout* and *autoFinishAfterTimeout* even allows creating scenarios that run on themselves. Furthermore, the creation of lectures as workflows allows re-using those “plans” quite easily: It is possible to load own templates, share an exported scenario with a colleague (by sending him/her the file), or load used-practice templates that are provided by the system. These used-practices can be extracted automatically by considering the similarity in the list of existing scenarios.

In addition, the concept includes novel functional blocks that allow executing *group-* and *peer interactions* within the online environment. This helps to implement a variety of well-known teaching scenarios (e.g., *Peer Instruction*, *Jigsaw Classroom*, *Think-Pair-Share*, or *Peer Feedback*) even in remote settings such as live-stream lectures. Moreover, it can also be beneficial for traditional settings that suffer from their size, which makes the grouping of students into meaningful groups or the interaction within these groups nearly impossible. As a solution, our concept includes a variety of build schemes for both the *GroupBuilder* and the *PeerBuilder*, e.g., to let students with different opinions on a specific task discuss in *Peer Instruction* (cf. figure 4.3).

Furthermore, the variety of parameters allows customizing the functionality of specific blocks even further or adjusting the display to specific groups of students. This enables lecturers to create and execute their individual teaching scenarios that are supported by interactive activities (i.e., technical tools).

Finally, a concept for runtime adaptation was added that allows changing running scenarios by extending them with further blocks. This allows taking real-time results into account when progressing the lecture, which is specifically important for student-centered approaches (as the students' input cannot be predicted in advance).

4.8 Summary

In this chapter, the concept of an adaptable collaborative learning environment was presented. Therefore, the following contents were presented:

In section 4.1, the conceptual idea of an adaptable collaborative learning environment was presented that consists of the following components: A *(meta-)model*, a *graphical editor*, a *runtime environment* (that is supported by *role concepts*), as well as a concept for *runtime adaptation*.

Section 4.2 presented the *(meta-)model* in further detail, from choosing the underlying idea of *workflows* as a suitable representation of lecture scenarios that are supported by technical tools over the available *structural blocks*, *transition blocks*, *functional blocks*, as well as *blocks for visualization* to a *preliminary evaluation*, in which the understandability and intuitiveness of the *(meta-)model* were proven.

In section 4.3, the concept of the *graphical editor* was presented by describing *related work*, *general design and concept decisions*, and different *supporting concepts* to ease the *initial phase* in the system, the *modeling task* and the modeling of *complex scenarios*.

In the next section, the *runtime environment* was presented by elaborating on different requirements before presenting each component of a suitable infrastructure in detail: The *frontend*, the *backend*, as well as the *runtime-clouds*.

Section 4.5 discussed the importance of integrating *role concepts* to provide more flexibility to the concept that the *(meta-)model* cannot express.

Afterward, section 4.6 motivated a concept for *adaptation at runtime*, allowing the extension of teaching scenarios by further blocks or templates.

Finally, the *didactic opportunities* of our approach were presented in section 4.7 by elaborating on novel functions and settings that have been added to the concept.

In the following chapter, the prototypical implementation of the approach is described that will later be used to discuss the research theses presented in section 3.5.

5 stARS – The scenario-tailored Audience Response System

This chapter describes the implementation of the previously presented concept of an *adaptable collaborative learning environment* in a prototype called *scenario-tailored Audience Response System (stARS)*. Therefore, the first section gives an overview using a global picture that includes all components with their respective technologies. Afterward, the implementation of each component will be presented in more detail before exemplary didactic scenarios are discussed. Finally, the chapter ends with a summary, in which the use of the prototype is visualized, and each component is highlighted accordingly.

Note on References

References to persons involved in implementing the concepts that were presented in the previous chapter will be provided in the respective sections or subsections.

5.1 Global Picture on the stARS Components

This section gives an overview of all components that are integrated into stARS (cf. figure 5.1). The combination of these components creates a prototype¹ that realizes the concept of an *adaptable collaborative learning environment*.

- The *(meta-)model* that specifies all *blocks and parameters* is implemented with the modeling tool Sirius², whose model description is created using Ecore³ that is part of the Eclipse Modeling Framework (EMF)⁴. An Acceleo⁵ script is used to convert the *(meta-)model* into a JSON representation.

¹ <https://stars-project.com/> – last successful access on October 8, 2021

² <https://www.eclipse.org/sirius/> – last successful access on October 8, 2021

³ <https://wiki.eclipse.org/Ecore> – last successful access on October 8, 2021

⁴ <https://www.eclipse.org/modeling/emf/> – last successful access on October 8, 2021

⁵ <https://www.eclipse.org/acceleo/> – last successful access on October 8, 2021

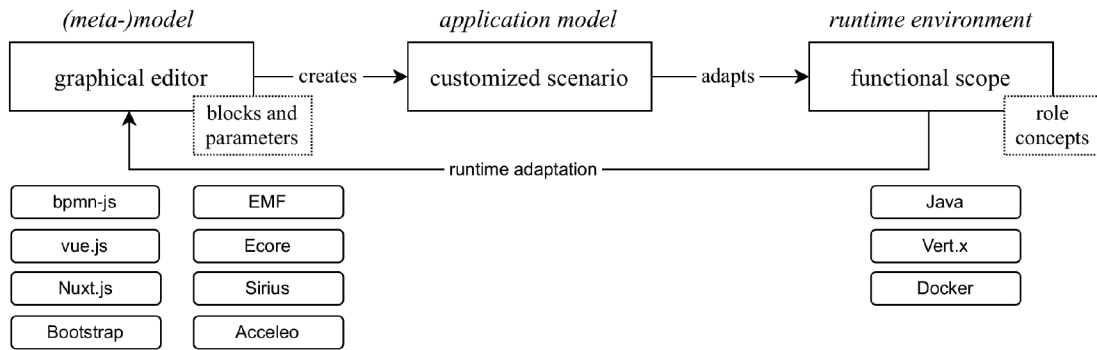


Figure 5.1: A global picture of the main components involved in the implementation of stARS.

- The *graphical editor* builds on top of bpmn-js⁶ and is integrated into a Single Page Application (SPA) that is developed in vue.js⁷ with the support of Nuxt.js⁸ and Bootstrap⁹ as a web framework. It reads the JSON representation and allows creating customized *application models* using the provided *blocks and parameters*.
- The *runtime environment* is implemented in Java using the application framework Vert.x¹⁰ as well as Docker¹¹. It is able to understand the *application models* and create instances with an adapted functional scope. An implementation of the role concept that is customized to our approach is integrated to add dynamic functionalities, which cannot be expressed by the *(meta-)model* itself. Moreover, the concept of *runtime adaptation* is implemented in order to extend running scenarios.

5.2 (Meta-)Model

The (meta-)model was realized by Ilja Shmelkin in the course of his master thesis (cf. [Shm19]). Later, it was extended by several *structural blocks*, *transition blocks* and *functional blocks*, as well as a variety of parameters in order to express further didactic scenarios.

While the previous chapter selected the concept of workflows as an intuitive representation for lectures that are supported by technical tools and specified both the elements

⁶ <https://bpmn.io/toolkit/bpmn-js/> – last successful access on October 8, 2021

⁷ <https://vuejs.org/> – last successful access on October 8, 2021

⁸ <https://nuxtjs.org/> – last successful access on October 8, 2021

⁹ <https://getbootstrap.com/> – last successful access on October 8, 2021

¹⁰ <https://vertx.io/> – last successful access on October 8, 2021

¹¹ <https://docker.com/> – last successful access on October 8, 2021

and parameters of the (meta-)model, this section will describe its implementation. Therefore, in the first step, the technology for creating the (meta-)model is selected. Afterward, the structure of it, as well as the implementation of the specified blocks and parameters, are described. In order to make the (meta-)model accessible to later applications, the implementation of a transformation script that converts it into a JSON representation is presented. The section ends with a discussion of the possibilities of validating *application models*.

5.2.1 Choosing a Technology to Implement the (Meta-)Model

In subsection 2.2.2, the relation between *domain-specific languages (DSLs)* / *domain-specific modeling languages (DSMLs)* and (meta-)models was briefly discussed, concluding that the *abstract syntax* (i.e., the domain concepts and rules) of a DSL / DSML is usually specified by a (meta-)model. The (meta-)model itself is a model that describes the concepts of the language, the relationships between them, as well as the structuring rules that constrain the model elements and their combinations to respect the domain rules [Gob19]. Thus, when realizing a (meta-)model, tools should be examined that allow creating DSLs / DSMLs. In the following, we will select an appropriate technology to create a DSL / DSML for the use-case of an *adaptable collaborative learning environment*.

In general, there exist three groups of modern *languages workbenches* that can be used to describe a DSL / DSML [Erd+15; Shm19]:

- *Textual language workbenches* represent the standard in DSLs and are most commonly used. A prominent example is Xtext¹².
- *Graphical language workbenches* provide the options to create both a DSML and a graphical editor, which is important to provide an intuitive starting point for users, who are not familiar with text-based programming languages.
- *Projectional language workbenches* can be understood as a superclass to textual and graphical languages and thus can represent anything that those languages offer and even more. They build on top of specially adjusted data models that project contents of a file into a user-readable form. Projections can be of any nature, i.e., also graphical or textual. However, these language workbenches suffer from their high complexity and are consequently only used if a hybrid solution is demanded.

¹² <https://www.eclipse.org/Xtext/> – last successful access on October 8, 2021

As workflows, which are commonly represented as graphs with vertices and edges, were selected as a proper representation to be used for the implementation of our (meta-)model, we focused on *graphical language workbenches*. This ensures that a graphical editor can be implemented on top of it, as it will be described in the upcoming section. In order to select an appropriate approach, we use the comparison of [Gra16] that is displayed in table 5.1.

Table 5.1: A summary of *graphical language workbenches* that was retrieved from [Gra16] with its original assessments (– = none, ✓ = poor, ✓✓ = good, ✓✓✓ = excellent).

Tools	Scope	Frame- work	Abstr. syntax	Concr. syntax	Syntax distinct.	Editing	Models	Auto- mation	Usa- bility	Meth. basis
Diagen	OS (GPL)	Eclipse	Ecore/ UML	DiaMeta Design	No	✓✓	✓✓	✓✓	✓✓✓	–
Eugenia	OS (EPL)	Eclipse	Ecore	EOL	Yes	✓✓	✓✓✓	✓✓✓	✓✓✓	–
GMF	OS (EPL)	Eclipse	Ecore	Draw2D	Yes	✓✓✓	✓✓✓	✓✓	✓✓	–
Graphiti	OS (BSD)	Eclipse	Ecore/ Java	Draw2D	Yes	✓✓✓	✓✓✓	✓✓✓	✓✓	–
MetaEdit+	Com	Own	GOP- PRR	Internal API	Yes	✓✓✓	✓✓	✓✓✓	✓✓✓	✓
Obeo Designer	Com	Eclipse	Ecore	Odesign	Yes	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓
Sirius	OS	Eclipse	Ecore	Odesign	Yes	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓
Tiger	OS (GPL)	Eclipse	AGG	Shape- Fig	No	✓	✓✓	✓	✓✓✓	–

As summarized in table 5.1, all investigated tools for creating a graphical representation of a DSML build on top of the Eclipse framework, except from MetaEdit+. However, as MetaEdit+ only provides a commercial approach, it was excluded from the selection. In order to describe (meta-)models, the Eclipse Modeling Framework (EMF) provides a (meta-)metamodel called Ecore¹³, which is used by the most listed tools (cf. table 5.1). Due to the fact that it builds the reference implementation of the *Essential Meta-Object Facility (EMOF)* of the *Object Management Group (OMG)* and thus has the same origin as UML, it is quite intuitive to learn for users who are already familiar with UML [Shm19].

Although several tools exist that provide a promising functional scope (cf. table 5.1), we have chosen the modeling tool Sirius, as it was still actively developed during the course of this thesis¹⁴. Moreover, compared to other approaches, Sirius benefits from its

¹³ Ecore as a (meta-)metamodel on layer M₃ is self-describing, i.e., it provides its own (meta-)model [Hoo14].

¹⁴ Since October 2020, even a web-based version of Sirius exists (cf. <https://www.eclipse.org/sirius/sirius-web.html> – last successful access on October 8, 2021).

active community, a detailed documentation, as well as from its easy to learn workflow [Doc21]. Sirius distinguishes the following concepts [Doc21]:

- A *viewpoint* is a logical set of representation specifications and representation extension specifications.
- A *representation* describes the structure, the appearance as well as the behavior of the modelers.
- A *mapping* describes how a semantic element (i.e., a *block* in our context) is represented as an instance (e.g., a container).
- Each *mapping* can be associated with one or multiple *styles* that describe the appearance of the concrete instance.
- *Tools* are used to interact with the elements of the *application model*, e.g., to create, edit, delete or connect them.

Even though Sirius supports expressive means to describe the creation of graphical modeling tools (i.e., editors), for the moment, we will not use the majority of the presented concepts. This is caused by the fact that our concept intends to integrate the editor in a web-based approach, rather than just allowing lecturers to model *application models* in an Eclipse internal editor. At the time of creating the (meta-)model (Mid 2019), Sirius-Web was not presented yet. In addition, all other efforts to automatically generate a web-based graphical editor from the model description (e.g., [PCG17; WGF17]) had not gained acceptance. For this reason, we decided to only concentrate on the description of the (meta-)model and develop a graphical editor afterward (cf. section 5.3). However, there is an opportunity to convert the project, whose implementation is described in the next subsection, to Sirius-Web and make use of the concepts described above. This is one of the reasons we build on a graphical modeling tool, anyway (cf. *NFR3*). Even though this could have several advantages, such as the opportunity to create *application models* collaboratively, it could also be restricted, as it is not especially targeted to the domain of modeling workflows. Nevertheless, this has to be evaluated in future work.

5.2.2 Implementing the (Meta-)Model

The *model description* in Sirius is created using Ecore that is part of the Eclipse Modeling Framework (EMF). For the modeling process itself, Eclipse (in the version of June 2019) has been used, which was to the time of creation the most recent version. In order to

install the requirements necessary, the most intuitive way is to select the *Eclipse Modeling Tools* version that preinstalls several extensions used. Only Sirius, as well as Acceleo (that is used for transforming the (meta-)model into a JSON representation (cf. the following subsection)) have to be installed manually to get started.

During the implementation, a structure was chosen that generally distinguishes between several *structural blocks*, *transition blocks*, *functional blocks* and *blocks for visualization*¹⁵, as described in the previous chapter. Moreover, the element *SubProcess* was added that we will use in the graphical editor as a graphical notation to group elements. A simplified version of this (meta-)model showing the first and second hierarchical levels of inheritance can be seen in figure 5.2. As the *structural blocks* do not share any parameters except for the ID, no parent class was created for them.

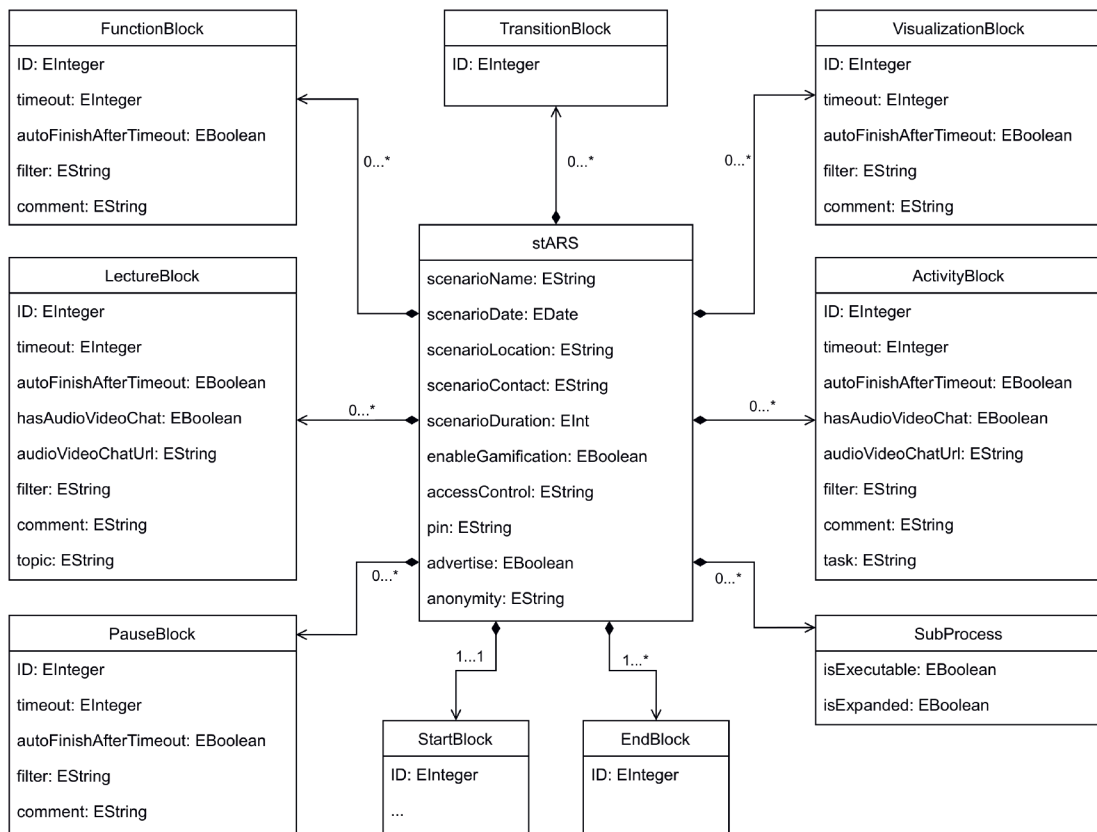


Figure 5.2: A simplified version of the (meta-)model that shows the first and second hierarchical levels of inheritance. The duplicate parameters of *stARS* and *StartBlock* are not listed twice (adjusted from [KSS19]).

¹⁵ In the initial implementation, *blocks for visualization* were part of the *functional blocks*. However, we decided to exclude them, as they differentiate strongly from *functional blocks* by not presenting an interactively used functionality.

All blocks of the (meta-)model are represented as *EClasses* with *EAttributes* and, if necessary, *EAnnotations* [Shm19; KSS19]:

- An *EClass* describes a container for attributes and has properties (e.g., *Name*, *isAbstract*, *isInterface*, *isContainment*, or *ESuperTypes*) that specify them in more detail. *EClasses* can inherit from other *EClasses*, include them, or implement them. The relationship between *EClasses* is denoted by connection elements.
- *EAttributes* describe typed elements within an *EClass* and are used to specify their parameters (i.e., the settings of the individual blocks as described in section 4.2). *EAttributes* have different properties, such as a *Name*, a *Type* (e.g., *EString*, *EBoolean*, *EInteger*), an optional *Default Value* as well as a *Cardinality*.
- *EAnnotations* represent textual notes (i.e., annotations) that can be used by third-party software to add additional functionality to the *Ecore* model. Examples are the description of graphical properties for *EClasses*, or the definition of constraints in the *Object Constraint Language* (OCL) (cf. subsection 5.2.4).

The blocks and parameters conceptualized in section 4.2 could be created in a straightforward manner. During implementation, the properties *Default Value*, as well as *Cardinality*, were of particular importance. Not only could it be used to define appropriate default values, for instance, for integer or boolean values, but it was also possible to mark parameters as required or optional by specifying the *Cardinality*. Due to the number of parameters present in the (meta-)model, this distinction is crucial for the implementation of the graphical editor (cf. section 5.3).

The result of the implementation is an expressive (meta-)model that contains a variety of *EClasses*, *EAttributes*, as well as several *EAnnotations* in order to be able to support a large number of teaching scenarios. To illustrate the size of the (meta-)model, it is shown in figure 5.3 – however, of course, it is not readable anymore¹⁶. In figure 5.3, an excerpt from the (meta-)model shows the functional blocks *LearningQuestion*, *Order-LearningQuestion* and *FreertextLearningQuestion*. We will explain it on the example of *FreertextLearningQuestion*: This block inherits from *LearningQuestion* (that inherits from *FunctionBlock*). Thus, *FreertextLearningQuestion* has both the parameters of the parent elements, as well as the parameters it defines on its own, i.e., *correctText*, *shortAnswer*, *caseSensitive*, *characterLimit* and *characterMinimum*.

¹⁶ For a complete and up-to-date version of the (meta-)model, please refer to <https://stars-project.com/metamodel.pdf> (last successful access on October 8, 2021).

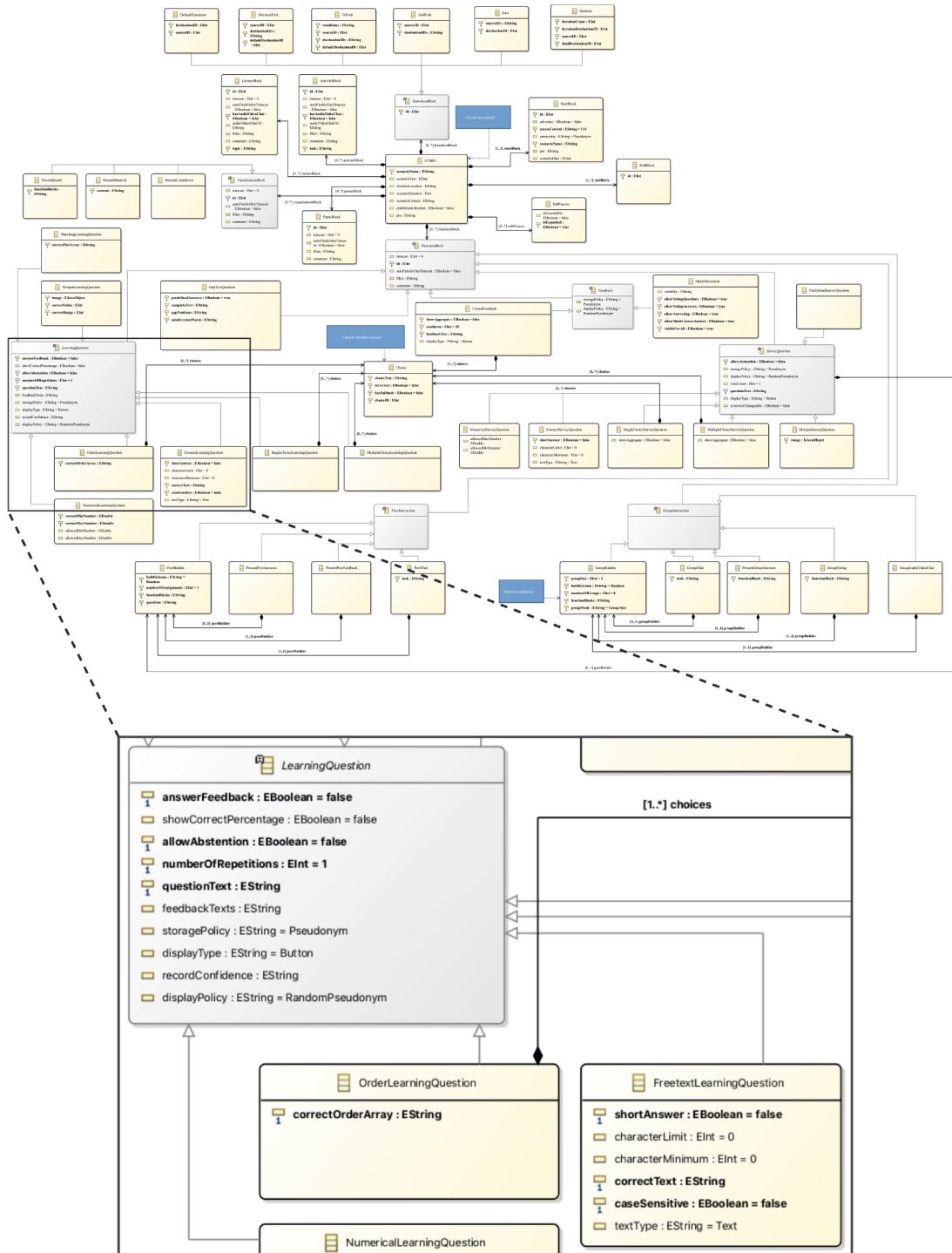


Figure 5.3: An overview of the (meta-)model to visualize its size (details are omitted intentionally), as well as an excerpt. For a better readable and up-to-date version of the (meta-)model, please refer to <https://stars-project.com/metamodel.pdf> (last successful access on October 8, 2021).

5.2.3 (Meta-)Model-Transformation to JSON

In order to use the (meta-)model in different applications (e.g., the graphical editor, or the backend or runtime-cloud), a parsable structure is highly recommended. This is done by transforming the (meta-)model that was described in EMF into a specific data format using Model2Text transformations, which describe structural transformations of a model representation, e.g., Ecore to JSON, by using tools for code generation. In Eclipse, the extension Acceleo¹⁷ is commonly used. The input is defined by a (meta-)model (not necessarily described in Ecore) and an Acceleo transformation template. Running these transformations produces one or more files of a different structure as an output. Normally, both a starter and transformer class are automatically generated in order to run transformations even without using the Eclipse IDE.

Although the (meta-)model is already described in a notation (XMI/XML) that can be parsed by different frameworks, for the ease of usage and readability, we target to convert it into JSON. Especially in the domain of web-based applications, JSON represents the most practical data integration [Mar17]. Thus, in the following, a template is created that describes this transformation. Therefore, it uses the Java package of the (meta-)model (*stARSpacage*) as a root element and traverses the full XMI tree. Each node of this tree represents an *EClass* (i.e., a *block* of the (meta-)model) and is analyzed regarding its parameters and attributes that specify it further. If a relevant element is found, it will be added to a JSON structure. Elements that are currently not relevant for further processing are omitted¹⁸ from the output [Shm19].

The template is shown in Listing 5.1 and will be summarized shortly: The first two lines describe the encoding of the template and the (meta-)model, which describes the rules that are used to traverse the input (in this case, Ecore). In line 4, both the name of the template as well as the root element of the transformation are specified. Next, in line 6, a new file is opened using the name of the root element with the desired file ending *.json*. In line 8 to 10, the meta information of the root element is extracted before line 11 to 39 describes a loop that iterates over all *EClasses* of the root element. In line 12, the name of the currently iterated *EClass* is added to the output as text (similar functions are present in the entire loop and are not described in the following). Line 13 checks whether the class inherits from others and lists the parent classes if available. This is done analogously for the elements *abstract*, *interface*, *parameter* and *references*, which are checked for their existence before being written to the output. Using *before*, *separator*

¹⁷ <https://www.eclipse.org/acceleo/> – last successful access on October 8, 2021

¹⁸ Consequently, it is a lossy transformation.

and *after* allows setting *delimiters* between the elements. The lines 40 and 41 close the output file and finish the template [Shm19].

```

1 [comment encoding = UTF-8 /]
2 [module generateJSON('http://www.eclipse.org/emf/2002/Ecore')]
3
4 [template public generateElement(stARSPackage : EPackage)]
5 [comment @main/]
6   [file (stARSPackage.name.concat('.json'), false, 'UTF-8')]
7     {
8       "name" : "[stARSPackage.name/]",
9       "nsPrefix" : "[stARSPackage.nsPrefix/]",
10      "nsURI" : "[stARSPackage.nsURI/]",
11      "classes" : [for (class : EClass | stARSPackage.eAllContents(EClass)) before
        ('{\n') separator(',\n') after ('\n\t\n')]
12        "[class.name/]":{
13          [if (class.eAllSuperTypes)->notEmpty()]
14            "superTypes" : [for (c : EClassifier | class.eAllSuperTypes)
              before ('[') separator(', ') after (',,')]"[c.name/]"[/for]
15          [/if]
16          "abstract" : [class.abstract/],
17          "interface" : [class.interface/][if (class.eAttributes->select(name
        <> 'id')->notEmpty())],
18          "parameter" : [for (a : EAttribute | class.eAttributes->select(name
        <> 'id')) before ('{\n') separator(',\n') after ('\n\t\t\t')]
19          "[a.name/]": {
20            [if (a.defaultValueLiteral)->notEmpty()]
21              "defaultValue" : "[a.defaultValueLiteral/]",
22            [/if]
23            "parameterType" : "[a.eType.name/]",
24            "lowerBound" : [a.lowerBound/],
25            "upperBound" : [a.upperBound/]
26          }[/for]
27        [/if]
28        [if (class.eReferences->notEmpty())],
29        "references" : [for (r : EReference | class.eReferences) before ('{\n
        ') separator(',\n') after ('\n\t\t\t\n')]
30        "[r.name/]": {
31          "isContainment" : [r.containment/],
32          "lowerBound" : [r.lowerBound/],
33          "upperBound" : [r.upperBound/]
34        }[/for]
35      [else]
36      [/if]
37    }
38  [/for]
39 [/file]
40 [/template]

```

Listing 5.1: An Aceleo transformation template that converts the (meta-)model into JSON.

In order to get an understanding of the output, an excerpt of it is shown in Listing 5.2 that describes the *LearningQuestion* and the *FreetextLearningQuestion* (the parameters were reduced as they are repetitive).

```

1 {
2   "name" : "stARS",
3   "nsPrefix" : "stARS",
4   "nsURI" : "stARS",
5   "classes" : {
6     ...
7     "LearningQuestion":{
8       "superTypes" : ["FunctionBlock"],
9       "abstract" : true,
10      "interface" : false,
11      "parameter" : {
12        "answerFeedback": {
13          "defaultValue": "false",
14          "parameterType" : "EBoolean",
15          "lowerBound" : 1,
16          "upperBound" : 1
17        },
18        ...
19      }
20    },
21    ...
22    "FreetextLearningQuestion":{
23      "superTypes" : ["FunctionBlock", "LearningQuestion"],
24      "abstract" : false,
25      "interface" : false,
26      "parameter" : {
27        "shortAnswer": {
28          "defaultValue": "false",
29          "parameterType" : "EBoolean",
30          "lowerBound" : 1,
31          "upperBound" : 1
32        },
33        ...
34      }
35    },
36    ...
37  }
38 }

```

Listing 5.2: An excerpt of the JSON output that was generated by the Acceleo template.

5.2.4 Define Constraints in the (Meta-)Model

While models based on Ecore are validated after their creation by the *Eclipse Modeling Framework (EMF)*, our (meta-)model defined a variety of *EAttributes* that describe the parameters of the *blocks*, which require further validation mechanisms to be added.

Therefore, the *Object Constraint Language* (OCL) can be used that allows formulating invariants in terms of restrictions on the validity of object models [OMG06]. An invariant represents a constraint that has to be valid for the entire lifetime of an object [ITW13].

In the (meta-)model, OCL can be used for different purposes. In the following, several examples are presented: First, it can ensure that each *block* has a unique identification number (ID). Next, it can describe restrictions between parameters, e.g., in the *GroupBuilder*, either a *groupSize* or a *numberOfGroups* can be specified, but not both at the same time. The reason for this can be found in the implementation: When setting a *groupSize*, the number of groups is calculated automatically, while when setting a *numberOfGroups*, the size of groups is calculated. Finally, OCL can describe constraints that specific parameters have to be defined, e.g., the *GroupChat* or the *PresentGroupAnswers* blocks have to refer to a *GroupBuilder* in order to be valid.

Note on Integrating Constraints

During the time of implementing the (meta-)model, only exemplary constraints were defined. For example, a *MutexGroupBuilder* rule was created that compares the parameters *groupSize* and *numberOfGroups* and throws an error if both are set, or a *CheckForDoubleID* rule that compares the IDs of every element and throws an error if two elements have the same ID. However, a complete list of constraints has to be evaluated during implementation and integrated into either the *graphical editor* or the *runtime-cloud* itself [Shm19].

5.3 Graphical Editor

The graphical editor was implemented by Lidia Roszko in the course of her Bachelor thesis (cf. [Ros19]). Later, it was integrated into the infrastructure (in this case, into a *single-page application* (SPA)), further refined and extended by several supporting functions.

As described in the previous section, at the time of creating the (meta-)model (i.e., Mid 2019), there was no suitable option to provide a web-based graphical editor directly through the graphical language workbench. However, a web-based editor was chosen as it can directly be integrated into the solution of a web-based learning environment and “may allow a much richer graphical representation of the DSML” [WGF17].

Consequently, a graphical editor had to be created that takes the (meta-)model as an input and provides all necessary options to create elements, connect them, specify parameters, or delete them. Although this solution lacks at automatically updating the editor if a new element is created, the available elements and parameters can still be loaded using the JSON representation (cf. subsection 5.2.3). Moreover, another advantage is the option to customize and integrate the editor into the already existing frontend application, which could be problematic in approaches that are delivered with the language workbench (e.g., Sirius Web that was presented in Mid 2020). In addition, another advantage that is closely related to the opportunity to customize the editor is the possibility to select a graphical editor that already contains concepts of the domain of workflows.

In the following, the implementation of the *graphical editor* is presented. Therefore, first, a suitable library for creating a graphical editor is chosen based on a list of previously defined requirements. Next, the general structure and the functions of the editor are described. Finally, the implementation of the concepts to support the lecturers during the usage of the graphical editor (cf. subsection 4.3.3) is described and its suitability is discussed.

5.3.1 Choosing a Suitable Library for a Graphical Editor

In subsection 4.2.6, the concept of workflows to represent teaching scenarios was proven to be a suitable method to describe the integration of technical tools in lectures. For graphical modeling in web-based systems, a variety of libraries exist [Ed-20] that will be presented in the following.

In order to rank libraries, in the first step, requirements had to be defined that ensure the customizability as well as integration into the already existing infrastructure that provides a *Single-Page Application (SPA)*. Therefore, the following must-have requirements and optional requirements were defined as summarized in table 5.2.

The graphical editor should be provided as a *JavaScript library* to integrate it into the existing infrastructure. Moreover, an *open source* solution is targeted to provide the option to offer the solution as *open source* itself, which would allow other institutions to use it as well. Next, a crucial requirement is the *addition of custom elements* in order to represent the (meta-)model, while importing the (meta-)model would be a goal to achieve. Furthermore, *ongoing support*, which allows maintaining the solution more easily, as well as *good documentation* to get started with ease, are demanded.

Table 5.2: The requirements for selecting a library to create diagrams [Ros19].

Must-have requirements	Optional requirements
JavaScript library	Drag-and-drop support
Open source	Main menu
Add custom elements	Inspector for elements
Ongoing support	Magnets for connections
Good documentation	Accessible by keyboard
Serializability	Snaplines
	Undo/Redo support

Finally, the last must-have requirement is *serializability*, as the created models have to be parsable to create custom *application models*. For faster development, it is useful if the libraries already support components that were evaluated as suitable during the user studies of the concept (cf. subsection 4.3.2). Thus, the support of *drag-and-drop*, a *main menu*, an *inspector for elements*, *magnets for connections*, *accessibility by keyboard*, *snaplines* as well as *undo/redo* has to be investigated.

The following graphical editors were selected from [Ed-20] for comparison: JointJS or Rappid¹⁹, mxGraph²⁰, GoJS²¹, jsUML2²², bpmn-js²³, Draw2D²⁴ and Mindfusion²⁵. Three of them (bpmn-js, Draw2D and Mindfusion) are specifically targeted to workflows. As shown in table 5.3, both mxGraph and bpmn-js support (almost) each requirement. In bpmn-js, *magnets for connections* are not supported – instead, a similar concept (i.e., a *connect tool* in a *halo*) is used. Therefore, both libraries were tested. While mxGraph supports all requirements, bpmn-js was evaluated to be the more suitable solution. This is caused by the fact that mxGraph relies on an old code-base and its user interface does not look modern anymore. Instead, bpmn-js is a continuously developed library that is especially targeted to support the modeling of workflows and provides an active community²⁶. Consequently, it was chosen as the foundation of the stARS editor.

¹⁹ Rappid is a commercial version of JointJS (cf. <https://www.jointjs.com/> – last successful access on October 8, 2021)

²⁰ <https://github.com/jgraph/mxgraph> – last successful access on October 8, 2021

²¹ <https://gojs.net/latest/index.html> – last successful access on October 8, 2021

²² <http://www.uco.es/users/in1rosaj/tools/jsUML2/editor/index.html> – last successful access on October 8, 2021

²³ <https://github.com/bpmn-io/bpmn-js> – last successful access on October 8, 2021

²⁴ <http://www.draw2d.org/draw2d/> – last successful access on October 8, 2021

²⁵ <https://www.mindfusion.eu/javascript-diagram.html> – last successful access on October 8, 2021

²⁶ <https://forum.bpmn.io/> – last successful access on October 8, 2021

Table 5.3: A comparison of requirements supported by libraries for graphical editors
(□ = not fulfilled, ■ = fulfilled) [Ros19].

		Joint- JS	Rapid	mx- Graph	GoJS	jsUML	bpmn- js	Draw- 2D	Mind- fusion
Must-have requirements	JavaScript library	■	■	■	■	■	■	■	■
	Open source	■	□	■	□	■	■	■	□
	Add custom elements	■	■	■	■	□	■	■	■
	Ongoing support	■	■	□ ²⁷	■	□	■	■	■
	Good documentation	■	■	■	■	■	■	■	■
	Serializable	■	■	■	■	■	■	■	■
Optional requirements	Drag-and-drop	■	■	■	■	■	■	■	■
	Main menu	□	■	■	□	□	■	□	■
	Inspector	□	■	■	□	□	■	□	□
	Magnets	■	■	■	□	□	□	■	■
	Keyboard accessible	□	■	■	■	□	■	□	■
	Snaplines	□	■	■	■	□	■	■	■
	Undo/Redo	□	■	■	■	□	■	□	■

A crucial requirement to model workflows with stARS elements in bpmn-js is the extension of the (meta-)model that can be understood. Therefore, the attribute *moddleExtensions* can be set during the instantiation of the bpmn-js modeler and a converted version of the (meta-)model is added for the key *stars*. Afterward, the editor is able to understand elements such as a *stars:singleChoiceLearningQuestion*.

5.3.2 Structure and Functions of the Graphical Editor

As described in section 4.3, the graphical editor consists of different components, i.e., the *main menu*, the *element palette*, the *modeling canvas*, as well as the *properties panel*. In this subsection, the implementation of each component is briefly summarized. A global picture of the final implementation of the editor that includes all four main components is displayed in figure 5.4 and visualizes the ongoing modeling process of a scenario with multiple rounds of interactive learning questions.

²⁷ On November 11, 2020, the development on mxGraph was stopped, resulting in the fact that this repository is effectively end of life (cf. <https://github.com/jgraph/mxgraph>). However, there are still efforts to keep the project alive (cf. <https://github.com/jsGraph/mxgraph>).

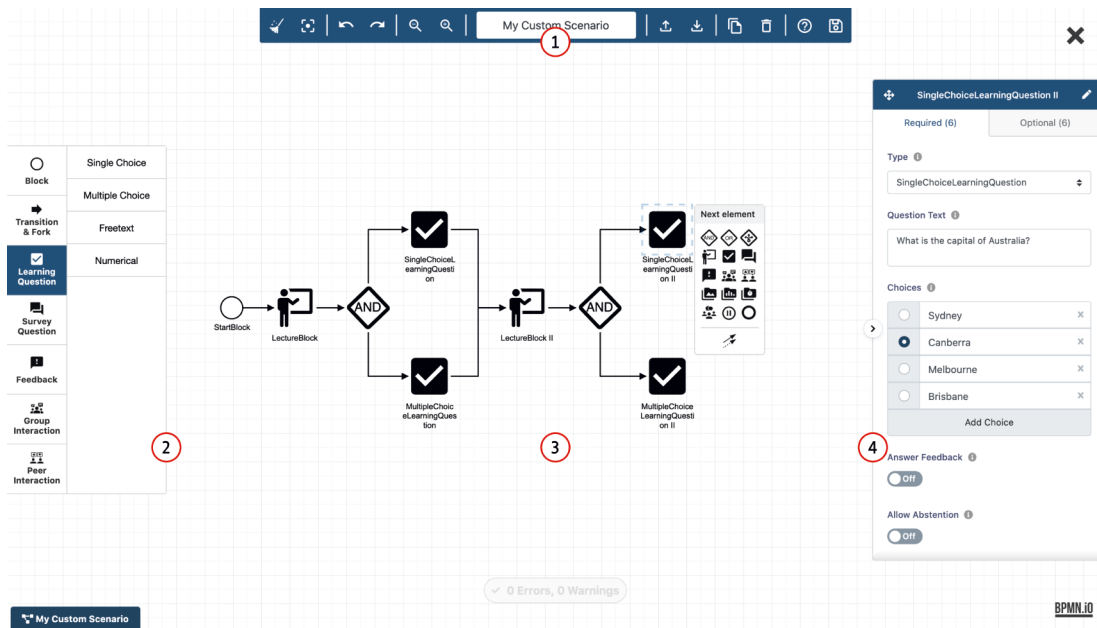


Figure 5.4: The final version of the graphical editor of stARS that consists of the *main menu* ①, the *element palette* ②, the *modeling canvas* ③ and the *properties panel* ④, which allows creating customized application models.

① Main Menu

The *main menu* was implemented as a custom component because bpmn-js does not include a menu on the top of the editor. Instead, functions such as loading or saving a diagram are provided on the bottom left corner. However, by overflowing the container of the editor, the menu could be added intuitively on the top of the editor and adjusted freely to match the requirements that were defined in the concept (cf. subsection 4.3.2). For realizing the desired icons²⁸, a custom icon component was added that does not only allow including icons of Font Awesome²⁹ but also custom icons by loading them manually as SVG files. Moreover, tooltips were added that are shown instantly when hovering an item. The final *main menu* of the graphical editor is depicted in figure 5.5.

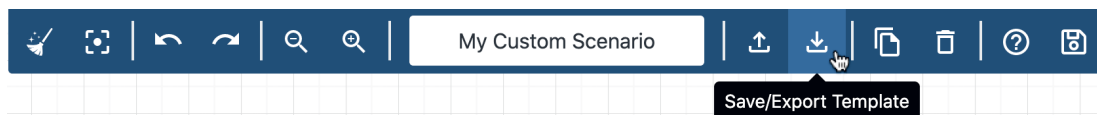


Figure 5.5: The *main menu* of the graphical editor of stARS.

²⁸ In the user studies, many suggestions were made to improve the icons.

²⁹ <https://fontawesome.com/> – last successful access on October 8, 2021

② Element Palette

While the *element palette* of bpmn-js is limited to one single level (see figure 5.6a) and consequently does not distinguish between abstract and specific elements, it had to be adjusted. Therefore, bpmn-js allows creating a *CustomPaletteProvider* and loading it using the *additionalModules* attribute when instantiating the editor. As this provider is mainly used to define the elements and their presentation in the *element palette*, a lot of customization had to be done: Thus, the elements of the (meta-)model were loaded and placed according to their type (i.e., abstract or specific) to the desired position. Moreover, an *openMenu* function was added to the abstract elements that is called when clicking it and opens or closes the sub menu for the current block. Instead, when using drag-and-drop, these abstract elements can be inserted anyway. For specific elements, both click and drag-and-drop functions are identical and insert this element as a new shape. In order to adjust the style of the *element palette*, unused elements were hidden (i.e., bpmn-js does include BPMN elements that are not used by stARS) and the palette was centered vertically.

A comparison of the *element palette* from bpmn-js and stARS is shown in figure 5.6. As can be seen in figure 5.6b, not all specific types that were defined in the (meta-)model are present in the *element palette*. This is caused by the prototypical implementation of stARS and the prioritization of novel features, rather than implementing a variety of question types that go beyond traditional question types such as *single-choice*, *multiple-choice*, *freetext* or *numerical*.

③ Modeling Canvas

The *modeling canvas* is the main component of the graphical editor and is used to create the workflow that represents the teaching scenario. Therefore, each element is visualized by an icon as well as a text. The icon for *functional blocks* is retrieved from the parent element (cf. figure 5.6b), which reduces the complexity and visualizes elements with similar behavior. Instead, the icons of the *transition blocks* and the *blocks for visualization* are specific for each element. In bpmn-js, the display of elements is controlled by a renderer. Similar to the *element palette*, it is possible to add a *CustomRenderer* that provides SVG representations of each new element. As the stARS elements inherit from *bpmn:ServiceTask*, its label is automatically displayed under the element if no incoming connection is placed at this location. Keeping in mind that the text can be controlled by setting the label of it, a *postExecute* function on the event *shape.create* was

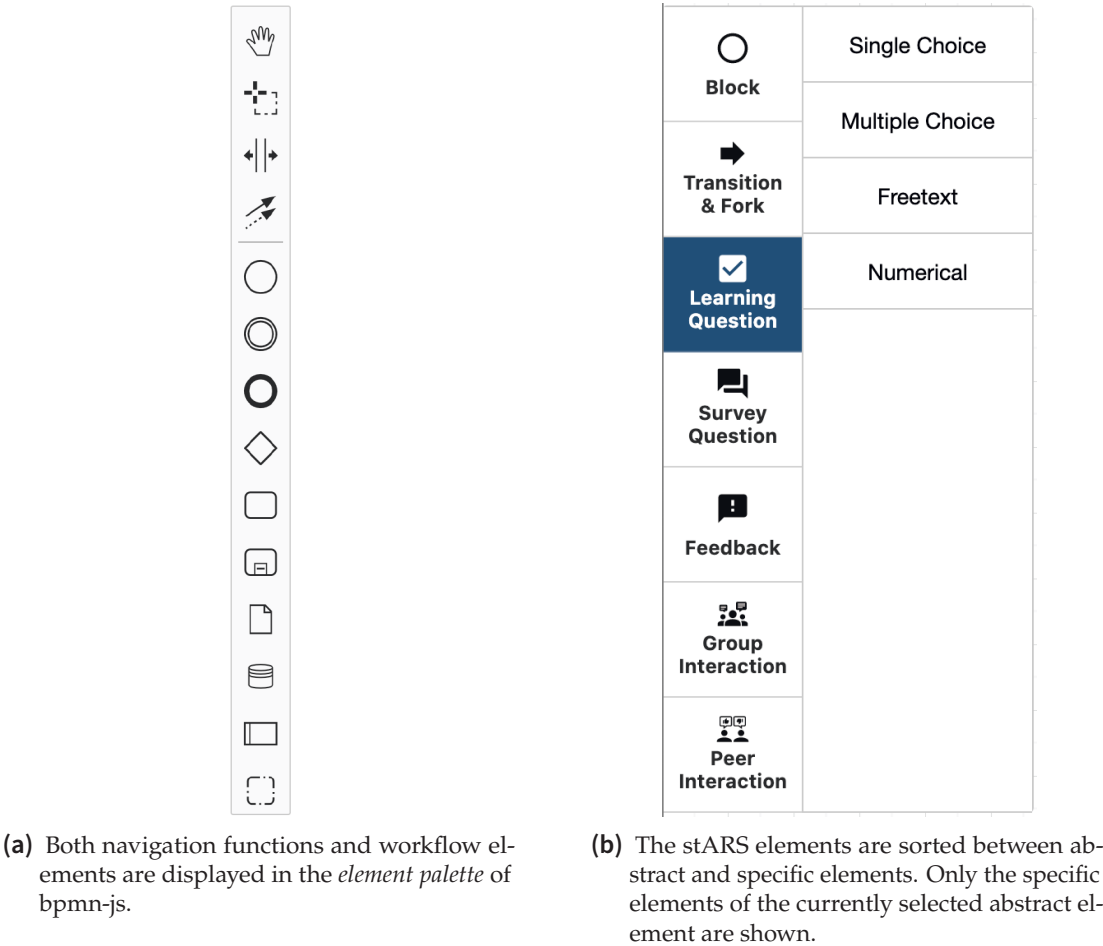


Figure 5.6: A comparison of the *element palette* of bpmn-js and stARS.

added that sets the label to the type of the element. If a second element with the same name is inserted, it gets the name of the type with a corresponding roman number to distinguish it (e.g., *SingleChoiceLearningQuestion II*). However, the text can be edited, either by double-clicking it or using the *properties panel*.

As described in the last subsection, bpmn-js does not support magnets to add connections between elements. However, a similar approach is presented by the *context pad* that is displayed next to the selected element and allows not only connecting elements but also adding the next element. While this is not always obvious in bpmn-js and also further settings are added, a *CustomContextPad* combined with CSS was used in order to match the design of the previously presented components. A comparison between the *context pads* of the traditional bpmn-js and stARS is displayed in figure 5.7.

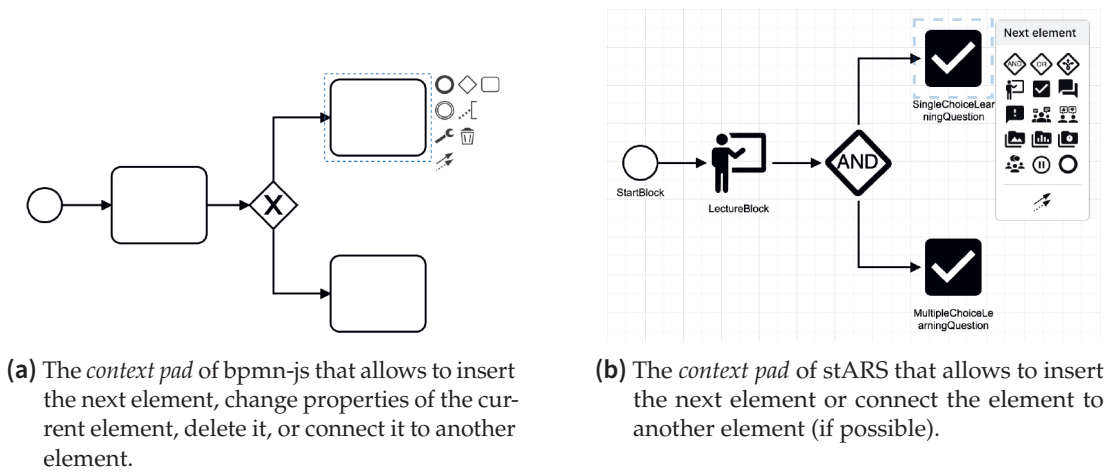


Figure 5.7: A comparison of the *context pad* of bpmn-js and stARS.

④ Properties Panel

Although there exists an official extension for a *properties panel* from bpmn-js³⁰, it was decided to create a custom component for realizing it. This is caused by several reasons: The *properties panel* provided by bpmn-js is very difficult to customize, for example, when different parameter types should be entered using different types of input fields, events have to be triggered when a parameter is set. Furthermore, parameters that depend on each other could also lead to problems, as those require custom input fields mandatory. Finally, the idea to hide or move the *properties panel* would not be possible without great effort.

Similar to the *main menu*, a component was created that overflows the editor component. However, since this panel is always shown when an element is selected, it has been implemented in such a way that it can be freely moved as well as collapsed if it disturbs the modeling process. Furthermore, it allows changing the name of the current element and setting values for the parameters that are classified between required and optional ones. For different types of parameters, different input fields are used, such as number inputs for numeric values or switch buttons for boolean values. Moreover, also custom inputs are supported, e.g., the input of choices as displayed in figure 5.8b. A comparison between the extension of bpmn-js as well as the *properties panel* implemented in stARS is visualized in figure 5.8.

³⁰ <https://github.com/bpmn-io/bpmn-js-properties-panel> – last successful access on October 8, 2021

(a) bpmn-js properties panel for 'Process_1':

- General**
 - Id:** Process_1 (with a clear button 'x')
 - This maps to the process definition key.
 - Name:** (empty text field)
 - Version Tag:** (empty text field)
 - ☐ Executable
- Documentation**
 - Element Documentation:** (empty text field)

(b) stARS properties panel for 'SingleChoiceLearningQuestion':

- SingleChoiceLearningQuestion** (with a plus icon and an edit icon)
- Required (6)** / **Optional (6)** (tabs)
- Type:** SingleChoiceLearningQuestion (dropdown)
- Question Text:** What is the capital of Australia? (text field)
- Choices:**
 - ☐ Sydney (with a clear button 'x')
 - ☒ Canberra (with a clear button 'x')
 - ☐ Brisbane (with a clear button 'x')
 - ☐ Melbourne (with a clear button 'x')
 - Add Choice** (button)
- Answer Feedback:** ☒ On
- Allow Abstention:** ☐ Off (with a tooltip: "The number of attempts to answer a question")
- Number Of Repetitions:** 2 (text field)

(a) The *properties panel* extension of bpmn-js that does only support simple inputs.

(b) The highly flexible *properties panel* of stARS that can be moved, collapsed and provides different kinds of inputs for different parameters (sorted between required and optional).

Figure 5.8: A comparison of the *properties panel* of the extension of bpmn-js and stARS.

5.3.3 Supporting Concepts

A variety of the supporting concepts proposed in subsection 4.3.3 were implemented in stARS. These include components to *support the lecturer in the initial phase* in the system, *the modeling*, as well as *the creation of complex scenarios*. In the following, a brief summary of the implementation of each group is given.

Support the Initial Phase in the System

The concepts for supporting the initial phase were implemented by Lidia Roszko and Robert Peine during a practical course as well as a master thesis.

In order to support the initial phase, on the landing page of the stARS prototype, the three steps that are necessary to support a custom scenario are described and visualized, namely, (1) define a customized workflow of interactive activities, (2) save it and (3) run it. In addition, both the student view, as well as the result view of the lecturer, are visualized. This should help to get a general understanding of the approach.

Next, after logging in as a lecturer and opening the editor for the first time, an overlay is displayed that includes two steps: First, the main functions of the editor are labeled (similar to the concept that is visualized in figure 4.8), and second, an example scenario is shown, which is described step by step.

In order to support lecturers in choosing appropriate elements, a *decision tree* was implemented. However, as mentioned in section 4.3.3, this dialog requires a lot of manual effort and was thus conceptually replaced by a *suggestion function* that proposes useful functions based on the currently existing scenario as well as a set of defined rules. Even though this concept was implemented, as visualized in figure 5.9, the definition of rules for proposing suitable extensions is a future topic of research (cf. [Pei21]).

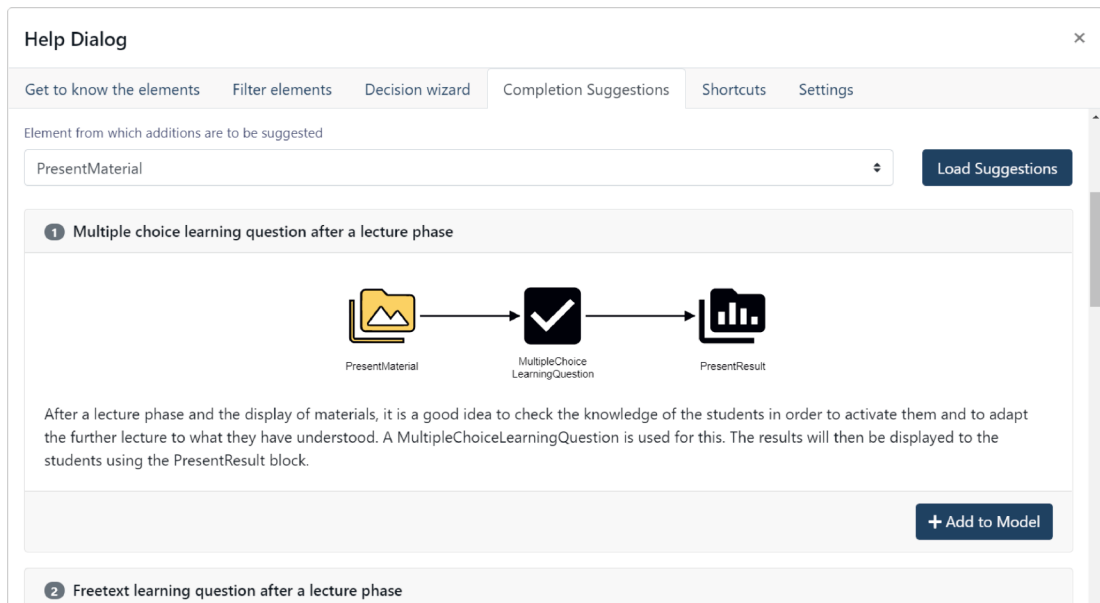
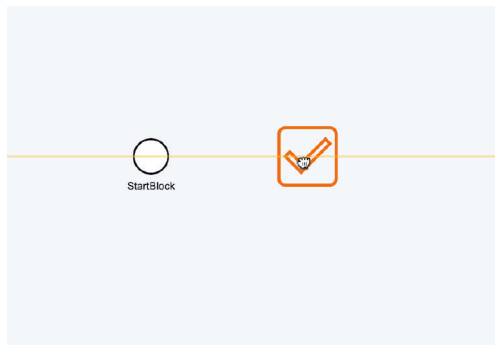


Figure 5.9: The implementation of a suggestion function for suitable next blocks [Pei21].

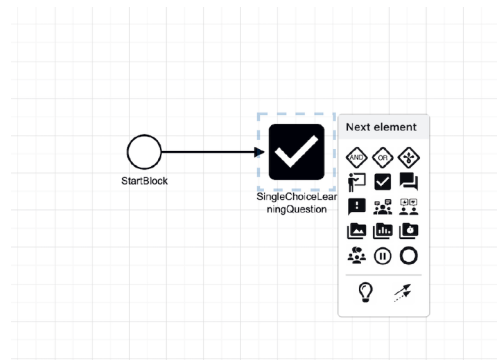
Support the Modeling Task

The concepts for supporting the modeling were partly implemented by Lidia Roszko and Niclas Zellerhoff during their practical courses.

In order to ease the modeling, the suggestions made in section 4.3.3 were implemented, i.e., *snap lines* as well as a *grid*, *automatic connection* of elements, the *centering* and *properly zooming*, and the *automatic parametrization*. In figure 5.10, some of those functions are visualized.



(a) When inserting a block, *snap lines* are shown.



(b) After inserting the block, it is *automatically connected* to the previous block and a *grid* is shown.

Figure 5.10: The implementation of *snap lines*, a *grid* as well as the *automatic connection* of blocks.

The concept of *reminder messages* was prototypically implemented as well. However, it was decided to initially disable these messages, as some lecturers rated them as disturbing during a preliminary user study.

Instead, a function was added, in which the validity of the model can be checked easily, as depicted in figure 5.11. This function is initially disabled when creating a scenario, as the initial state that is loaded (i.e., a single *StartBlock*) is invalid and thus, would confuse lecturers. However, it can be activated at any time and is also activated during the editing of existing scenarios by default. The model checking differentiates between warnings and errors. While models with warnings can be executed, models with errors would result in runtime errors and thus, have to be fixed before starting them.

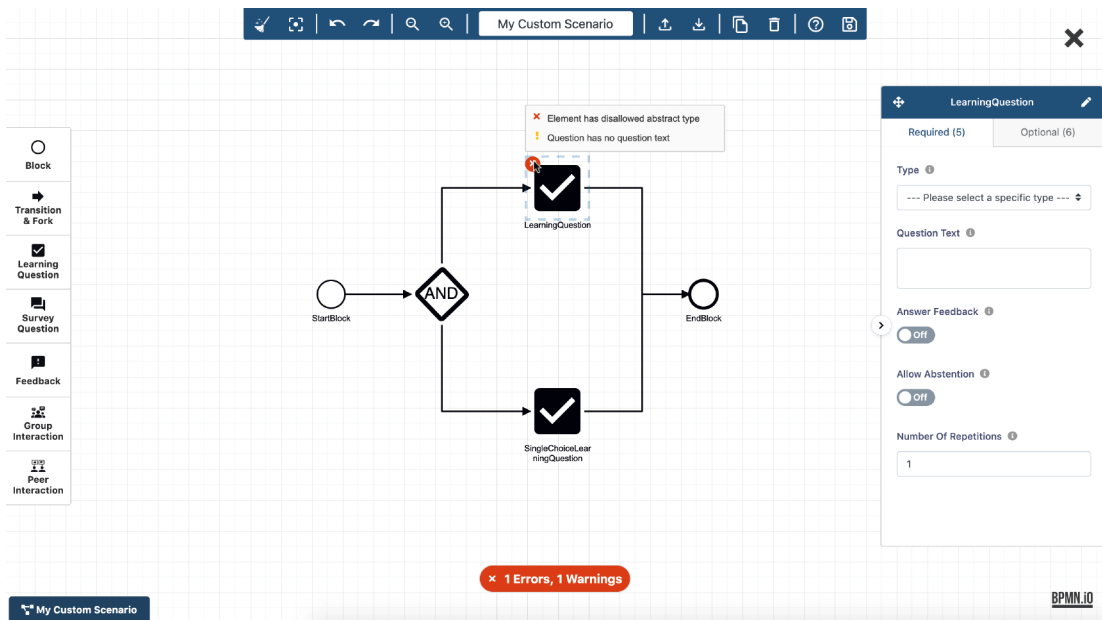


Figure 5.11: An invalid scenario that is validated by the model checking functionality.

Support Complex Scenarios

The concepts for supporting the modeling of complex scenarios were implemented by Sinthujan Thanabalasingam and Chang Hong during their master theses.

A crucial function when trying to model complex scenarios is the re-use of existing parts of other scenarios (i.e., templates). Therefore, [Tha21] implemented expressive means to save or export templates (the implemented dialog is similar to the concept visualized in figure 4.10), load or import them into the existing scenario, or exchange the entire model with it. As a common use case is to build new scenarios from templates, the implementation of adding templates to existing scenarios is visualized in figure 5.12. The template will be integrated as a sub-process, which can be reduced (i.e., minimized) or integrated (i.e., the container, as well as the StartBlock and the EndBlock(s) of the sub-process, are removed and connected properly to the existing scenario). Furthermore, it can either be edited in the combined view or using the tab view on the bottom left, in which free editing of the added template is possible (cf. [Tha21]).

The sharing of templates that was proposed in section 4.3.3 could so far not be implemented. However, this should be a future topic of research.

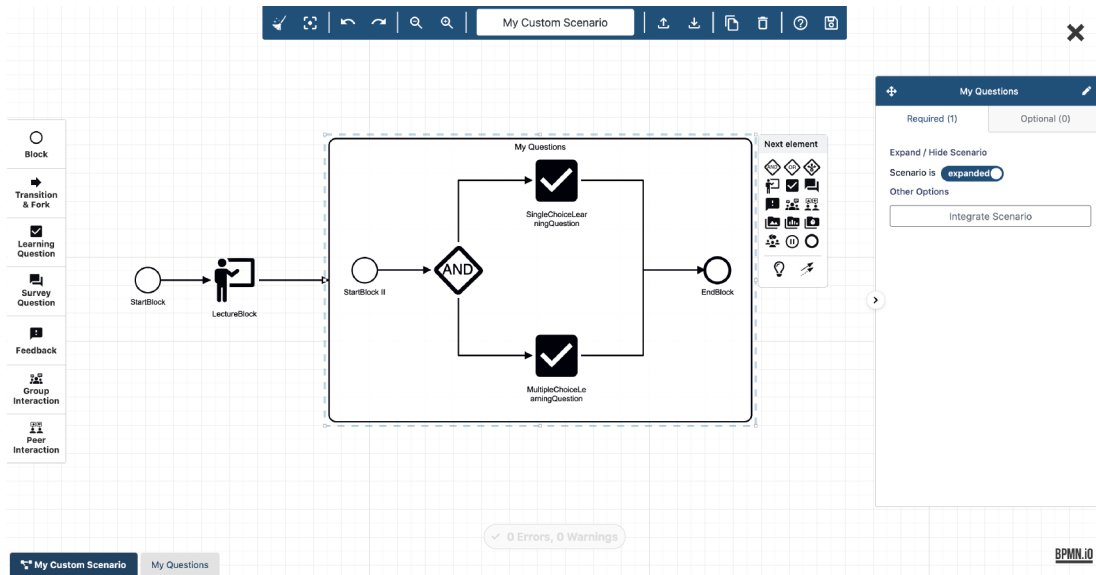


Figure 5.12: A scenario in which a template was added to the existing model.

Instead, the function to extract *used practices* of existing scenarios was implemented prototypically in [Hon21]. As shown in figure 5.13, different representations of a template can be extracted by this functionality. Moreover, in addition to the template's *size*, information on the number of lecturers that modeled this template (*popularity*) as well as the total number of occurrences among all scenarios (*frequency*) are exported.

Templates (Count: 34)

+ Create Scenario

Sort Options ▾ Size of Templates ▾ << < 1 2 3 4 ... > >> Preferred Elements ▾ templates

Template		
#1	#2	#3
Popularity: 3 Frequency: 5 Size: 4		
Export		
Template		
#1	#2	
Popularity: 2 Frequency: 4 Size: 4		
Export		

Figure 5.13: The result of the template generation, in which different representations of templates as well as popularity, frequency and size are displayed [Hon21].

Although this functionality already presents expressive means to detect reoccurring structures, a problem got obvious when investigating the execution time. The number of comparisons necessary and, thus, the total time of computation heavily increases with the number of scenarios available. This issue has to be investigated in future research, as exporting templates from thousands of scenarios would currently last several days. Moreover, this large number of generated templates is another challenge to tackle, as suitable templates have to be found and exported (cf. [Hon21]).

5.4 Runtime Environment

The architecture of the runtime environment was originally proposed by Ilja Shmelkin in the course of his master thesis (cf. [Shm19]) and later implemented with slight adjustments in collaboration with Robert Peine during his time as a student research assistant.

Although the graphical editor is the main component of stARS, it is useless without a runtime environment that is able to interpret the teaching scenarios created with it. Thus, as motivated in section 4.4, a *runtime-cloud* had to be implemented that understands those and provides the desired functionality. Moreover, this *runtime-cloud* had to be connected to a *backend* that manages the scenarios and runs them on these cloud servers. Furthermore, it has to enable access to a *frontend* that displays the user interface for the administrators, the lecturers as well as the students. In the following, each component of the implemented infrastructure (cf. figure 5.14) is presented shortly – the main functionalities of each component will be repeated.

5.4.1 Backend

The backend is used to perform general tasks such as managing users, their authentication, as well as saving, starting, pausing, resuming and stopping (i.e., resetting) scenarios that were created using the graphical editor of the frontend (cf. section 5.3). Therefore, both a Java application³¹ and a Mongo database are used, each running in a separate Docker container, which simplifies the maintainability (cf. *NFR7*). The Java application processes requests from the frontend and communicates with the runtime-cloud if required. For example, if the request to start a scenario is received by the backend, it is

³¹ The Vert.x framework is used in order to develop a web application with RESTful service more efficiently.

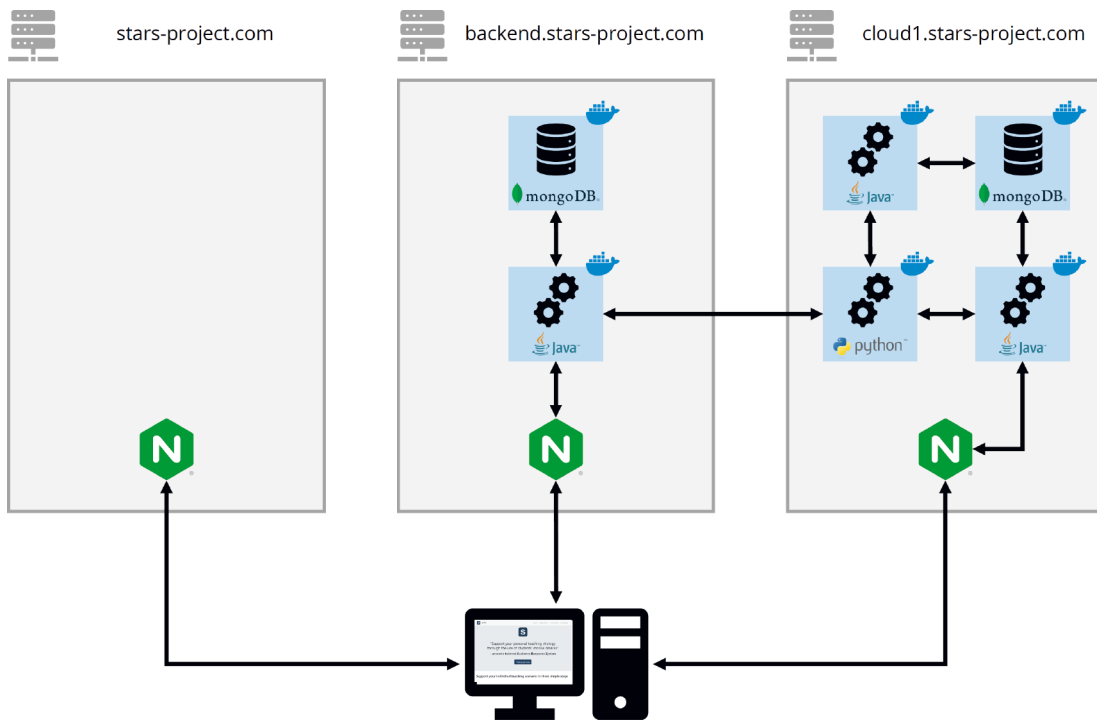


Figure 5.14: The infrastructure of the prototype called scenario-tailored Audience Response System (stARS) [Pei21].

communicated to a cloud server that handles it and the data necessary to connect to the started scenario (i.e., the URL of the cloud server and the exposed port) is stored in the Mongo database. Moreover, the backend is used for the initial authentication of users, i.e., they receive a JWT that is used to authenticate on one of the cloud servers. Furthermore, even the cloud servers available are managed by the backend. The backend of the stARS prototype is accessible via <https://backend.stars-project.com/>.

5.4.2 Runtime-Cloud

The runtime-cloud is used to perform the execution of scenarios. This is done by starting a Java application³² that allows executing scenarios in its own Docker container on one of the cloud servers specified by the backend. For the creation of these containers, a Python utility script is used. After a scenario is started, both the lecturer and the students can communicate with it using the frontend, which communicates with the runtime-cloud using the URL of the cloud server and the exposed port (the information is retrieved from the backend). Depending on the *accessControl* of the scenario, the

³² The runtime-cloud uses the Vert.x framework as well.

communication requires authentication at the cloud server. The data that is retrieved during the execution of scenarios (e.g., the blocks and their transitions, as well as the states such as the answers to currently active questions) is stored in an instance of a Mongo database on the cloud server. The stARS prototype currently uses one cloud server that can be accessed via <https://cloud1.stars-project.com/>. However, it can be extended by further cloud servers easily.

5.4.3 Frontend

As shown in figure 5.14, the server <https://stars-project.com/> is used to retrieve the data that is needed to generate the frontend (i.e., the user interface) in the browser. This frontend communicates with both the backend and cloud servers, as described above. It is realized as a Single Page Application (SPA)³³ using Vue.js³⁴ (cf. [Mac18]). This framework was selected because it generally has a high rate of satisfaction [BGR20] and provides low barriers in order to get started compared to analog frameworks [Tea21]. However, it could be replaced by any other frontend framework that is capable of creating SPAs (i.e., allow rewriting contents dynamically), which is a crucial requirement in such a dynamically changing service (i.e., *digital learning environment*). Vue.js was further extended by Nuxt.js (cf. [Kok20]) – a meta-framework (also called app-level-framework) that offers further functions, such as an improved scaffolding that provides a uniform structure of the project. In order to make the frontend available for different devices, the framework Bootstrap³⁵ is used – at the time of writing, the world’s most popular framework for building responsive, mobile-first websites. However, this is mainly important for the student view as building responsive graphical editors is a separate topic of research (e.g., [PNP17; VM17]). Thus, when choosing a library for realizing the graphical editor, the focus was on implementing the concept that was described in section 4.3³⁶. Therefore, the library bpmn-js was selected, which provides an editor for modeling BPMN models that can be customized extensively (cf. subsection 5.3.1). A variety of further libraries were integrated, such as FontAwesome to provide icons or Chart.js to create diagrams for the evaluation during the lecture.

In order to visualize the general functionality, the following paragraphs will briefly visualize the implementation of the frontend.

³³ A SPA allows rewriting contents dynamically with new data instead of reloading the complete page.

³⁴ The implementation used Vue.js 2.6.11.

³⁵ The implementation used Bootstrap 4.4.1.

³⁶ Nevertheless, providing accessibility for lecturers should be a future topic of interest.

After logging in as a lecturer, new scenarios can be created, which are then displayed in a list view³⁷ (cf. figure 5.15). Therefore, the name of the scenario, a preview of the model as well as the scenario date and the information whether the scenario is running or not is displayed, as proposed in section 4.4. Furthermore, if at least two scenarios exist, the “Sort Options” are displayed, which can be used to either sort the scenarios by their name or date in ascending or descending order. Moreover, pagination options exist as soon as more than 20 scenarios exist.

The screenshot shows the stARS Dashboard interface. At the top, there is a navigation bar with the stARS logo, a user profile icon labeled 'tkubica', and links for 'Legal Notice', 'Accessibility', 'Changelog', and 'Logout'. Below the navigation bar, the main heading is 'Dashboard'. Underneath, it says 'Scenarios (Count: 5)' and there is a '+ Create Scenario' button. A 'Sort Options' dropdown menu is visible. The dashboard displays four scenario cards: 'Interactive Learning Questions', 'Sequential Questions', 'Peer Instruction', and 'Jigsaw Classroom'. Each card shows a workflow diagram, the scenario date and time, and its current status (e.g., 'stopped'). There is an 'Enter Scenario' button on each card. The 'Interactive Learning Questions' card shows a decision diamond leading to two paths, one with a checkmark and one with a cross. The 'Sequential Questions' card shows a linear flow from 'StartBlock' through 'SingleChoiceLearningQuestion' and 'MultipleChoiceLearningQuestion' to 'EndBlock'. The 'Peer Instruction' and 'Jigsaw Classroom' cards show empty boxes for their respective workflows.

Figure 5.15: The scenario overview of a lecturer with the options to create a new scenario as well as edit or delete existing scenarios.

When clicking on “Enter Scenario,” the scenario view is opened, as visualized in figure 5.16. Within this view, the scenario can be started, paused, resumed, or stopped (i.e., reset), depending on the currently active state. In addition, the workflow can be managed using either the model or list view. Moreover, the participation details can be opened, in which both a QR code and a link to join the scenario are presented. Furthermore, the real-time results of the currently active blocks are displayed using either a bar chart, a pie chart, or a text. In addition, a summary of the currently active blocks is presented as well.

³⁷ The student view is very similar to this view, except for the buttons to create, edit and delete scenarios. Instead, it is possible to join (and leave) scenarios.

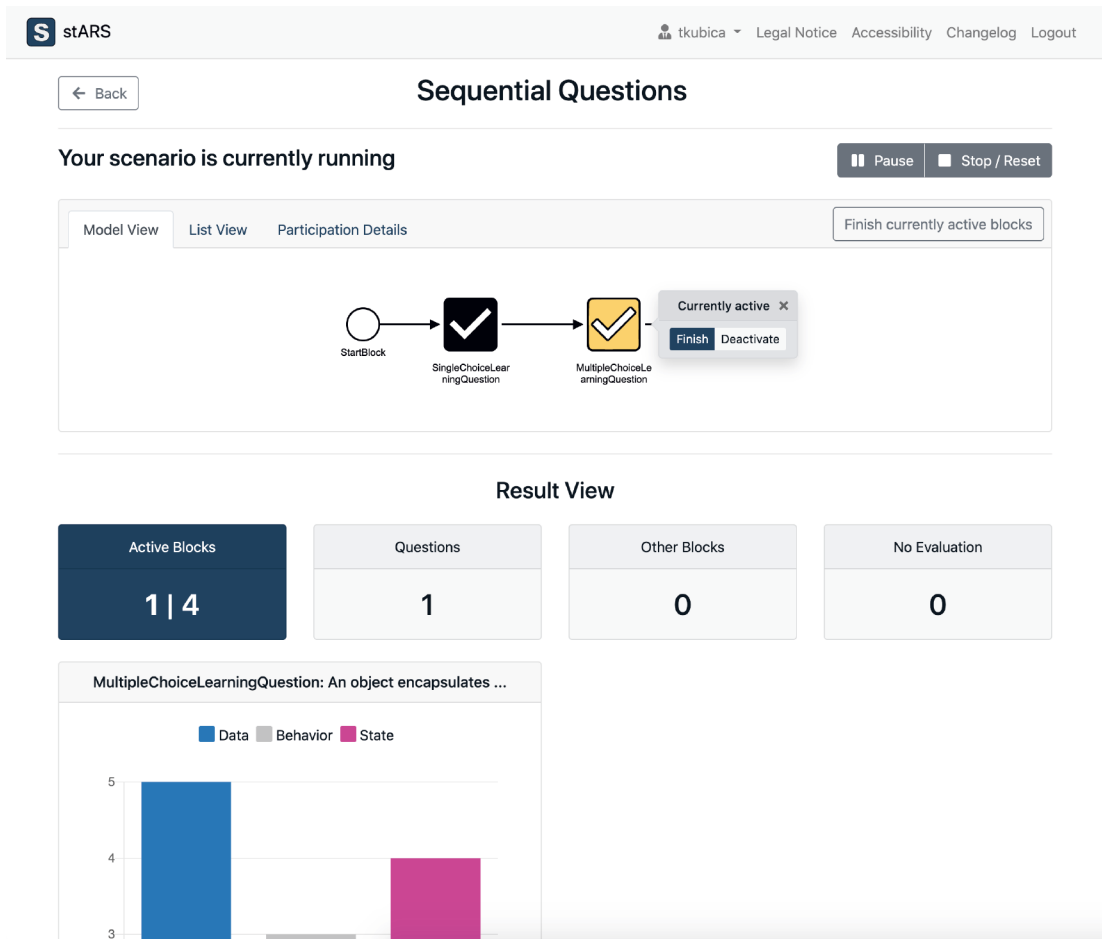


Figure 5.16: The scenario view with the options to start, pause, resume or stop the scenario, as well as managing the workflow and showing the real-time results of the currently active blocks.

In the student view, which is visualized in figure 5.17, the currently active blocks are displayed and can be used, e.g., in the example, a *MultipleChoiceLearningQuestion* can be answered. Depending on the defined parameters, the question behaves differently. For example, if the parameter *answerFeedback* is set to *true*, the correct answer is displayed after answering.

Another information that is displayed on the student view is a randomly generated pseudonym, which is used for activities that require an identity to be used, such as an *OpenDiscussion* or a *GroupChat*.

The screenshot shows the stARS web interface. At the top, there's a navigation bar with the stARS logo, a user icon labeled 'tkubicastudent', and links for 'Legal Notice', 'Accessibility', 'Changelog', and 'Logout'. Below this, a 'Back' button is on the left, and the title 'Sequential Questions' is centered. Under the title, 'Student View' is shown on the left and a user pseudonym 'FluffyWalrus38' on the right. The main content area has a light gray background. It starts with a question 'An object encapsulates ...'. Below the question are three radio button options: 'Data', 'Behavior', and 'State'. At the bottom of this section is a dark gray button labeled 'Answer'.

Figure 5.17: The student view, in which the currently active blocks can be used. Moreover, a random pseudonym that was generated for the scenario is displayed.

5.5 Integration of Role Concepts

The role concepts were implemented together with Robert Peine during his time as a student research assistant. The results of this section were partly published in [KPB20].

In this section, the implementation of the concept of roles in the stARS runtime is described. Therefore, in the first subsection, an overview of general implementation details as well as the hierarchy of inheritance is presented. Next, concrete implementation details are given for each use case that was proposed in subsection 4.5.3. Finally, an outlook on the tradeoffs of using the concept of roles in *digital learning environments* is given that will be further elaborated on in section 6.5.

5.5.1 Implementing the Role Concept in stARS

In stARS, an implementation of the role concept that supports selected role features (cf. section 4.5) was integrated. Therefore, as a foundation, the class *Role* was added that implements the *Comparable* interface, which is used to sort lists or arrays of custom

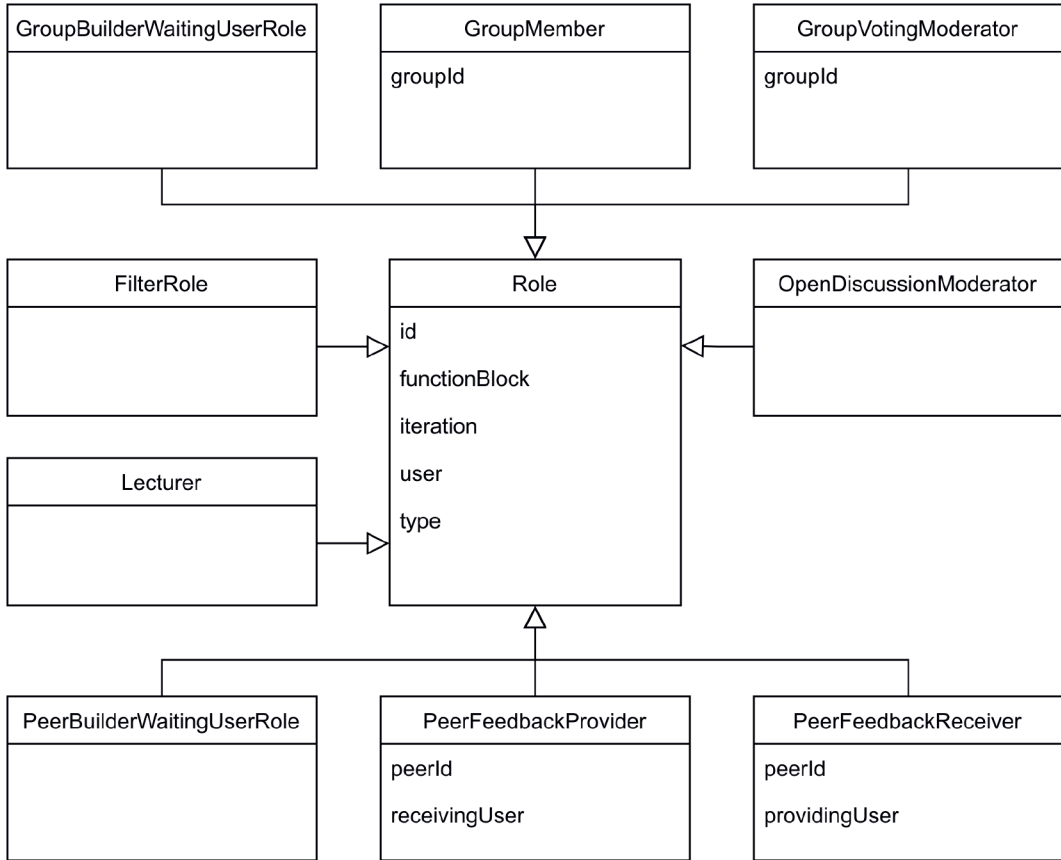


Figure 5.18: The role hierarchy that is implemented in stARS. Users are assigned with the specific role types that inherit from *Role*.

objects, i.e., the role assignments. If a role is played by a user, an instance of a specific role type (e.g., *FilterRole*) that extends the *Role* class is created that defines the underlying functionality. Each instance of a *Role* describes a *plays* relation and consists of a *compartment*, a *player*, as well as a *role type*. As motivated in subsection 4.5.3, the *compartment* is described by the current block (i.e., *functionBlock* in stARS) and the current iteration. If no current block is set, the *compartment* will be the scenario itself. The *player* is the user that holds the role. Finally, the *type* is denoted by the type of the specific role. A hierarchy of inheritance that lists all role types that were implemented in stARS is displayed in figure 5.18.

In comparison to similar approaches such as LyRT [Tai18] or SCROLL [Leu17], our approach does not include a look-up table (or similar functionality) that decides, depending on the order of *plays* relations, which function has to be called. As a consequence, the current role implementation of stARS does not allow playing the same role multiple

times in the same *compartment* (i.e., the current block and iteration, or the scenario). This design decision was made intentionally, as there is no meaningful use case in stARS that required it. This is also caused by the decision that the *compartment* is defined by a specific *functional block*. Even if students are *GroupMembers* or *GroupVotingModerators* of multiple group tasks/votings, our role implementation would still work as the roles are played in different *compartments*. In order to use the role implementation also in the frontend and keep track of the currently played roles, a route was added that provides a list of privileges (i.e., roles). Changes, when a role is acquired or abandoned, are triggered by a WebSocket.

In the following, implementation details on several use cases are provided that integrate the concept of roles into stARS.

5.5.2 Different Use Cases that Implement the Role Concept

As visualized in figure 5.18, a variety of use cases were implemented that integrate the role concept. The *GroupBuilderWaitingUserRole* just represents a simple role assignment that is created when acknowledging the participation in a group task, i.e., the current user, the *functionBlock* as well as the current iteration are stored in the instance of the role. When the *GroupBuilder* finishes, the users that are part of the group formation are retrieved from the instances of *GroupBuilderWaitingUserRoles* with the same *functionBlock* and iteration. Each user is assigned to the role of a *GroupMember* that holds the *groupID* the user is assigned to, as well as the common properties of the *Role* (i.e., *functionBlock*, iteration, user). Afterward, the *GroupBuilderWaitingUserRole* will be abandoned because the user is not waiting anymore.

If a compartment of the type *GroupVoting* is active, the role of a *GroupVotingModerator* might be relevant. This role is automatically assigned to one of the *GroupMembers*, if no common group answer can be found. By holding the role of a *GroupVotingModerator*, the *GroupMember* is allowed to input his/her group's answer. The role is abandoned after the *GroupVoting* is finished.

Similar to the *GroupBuilderWaitingUserRole*, a *PeerBuilderWaitingUserRole* exists that is assigned to all users that joined the peer task. After finishing the *PeerBuilder*, the assignments are created, i.e., depending on the parameters set, each user with the *PeerBuilderWaitingUserRole* is at least once assigned as a *PeerFeedbackProvider* and a *PeerFeedbackReceiver*. After all *PeerFeedbackProviders* and *PeerFeedbackReceivers* are assigned, the *PeerBuilderWaitingUserRoles* are abandoned automatically.

Another role that depends on the currently active compartment is the *OpenDiscussion-Moderator*. This role can be dynamically assigned to one or multiple selected users. However, it is automatically assigned if the user already holds the role of a *Lecturer*. The *Lecturer* is a global role that influences the entire scenario. If a user holds this role, he/she is allowed to manage the current workflow and retrieve the real-time results. As a result of a later lecture experiment (cf. subsection 6.2.2), this role was replaced by the two distinct roles *Controller* and *Evaluator*, which allows functionalities to be specified in greater detail.

Finally, a *FilterRole* exists that is assigned to all users that match the requirement of the filter when the block is activated. If a parameter *filter* exists on a block, only users holding the *FilterRole* can retrieve it. The *FilterRole* is automatically abandoned as soon as the block finishes.

5.5.3 An Outlook on the Tradeoffs of Using the Role Concept

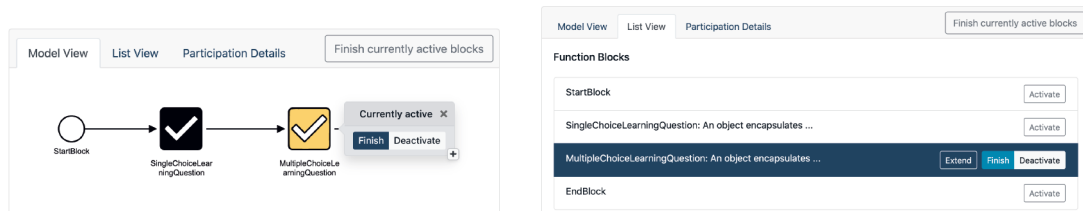
While using the concept of roles for implementation might feel quite differently, this could be caused by the habit of object orientation. The developer has to rethink how to implement a specific function in a role-based manner. What was particularly noticeable, however, was that after implementing the first use case, other use cases could be implemented analogously without further problems. Moreover, getting started seems even easier than getting started with object orientation, in which a specific way of thinking has to be understood.

In the context of stARS, the integration of the concept of roles provides a suitable extension (cf. [KPB20]). It allows separating the data that is generated at runtime from the data that is specified by the (meta-)model. This could result in a positive effect when extending the concept, as large parts of the runtime could be created using code generation and only the roles have to be adjusted accordingly. Furthermore, it allows implementing different behavior for different user groups, e.g., as shown in the *OpenDiscussion*, in which further abilities are received if the role *OpenDiscussionModerator* is acquired. Finally, roles might also be used to restrict access to specific user groups, as shown by the *FilterRole*.

5.6 Adapting Scenarios at Runtime

Even though the implementation already includes expressive means of flexibility to navigate through the created workflow (e.g., *DecisionForks* can be used to choose a subsequent path, or blocks can even be activated and deactivated manually), the lecturer has to create all necessary (paths of) activities in advance. This is not always possible as unpredictable changes happen frequently: For example, the lecturer might want to conduct a survey question or a group interaction spontaneously if the progression of the lecture demands it. Similar situations occur in student-centered approaches, in which the progression strongly depends on students' submitted data that cannot be foreseen. Consequently, in section 4.6, a concept for runtime adaptation was presented, whose implementation will be described briefly in the following.

The general idea of this functionality is extending the scenario starting from a selected block. Thus, it is directly integrated into the options to manage the progression of the scenario, as visualized in figure 5.19 for both the model view (using the button with the plus icon) and the list view (using the *Extend* button).



(a) The option to extend the scenario using the button with the plus icon in the model view.

(b) The option to extend the scenario using the *Extend* button in the list view.

Figure 5.19: A comparison of the extension options in the model and list view.

After opening the extension dialog, three options are provided (cf. figure 5.20): The scenario can either be extended by a *public* or *private template* (the number of available templates is listed in the text) or a *custom extension* (i.e., a new model is created as an extension) can be made. However, even if a template is selected, it is only used as a starting point and loaded into the editor, in which it can be adjusted freely. This is required to adjust parameters such as the link to previous questions.

If the extension by a public or private template is selected, the already existing import dialog is used to choose a template, which is then loaded into the editor. If the custom extension is selected, an empty editor containing only a *StartBlock* is opened.

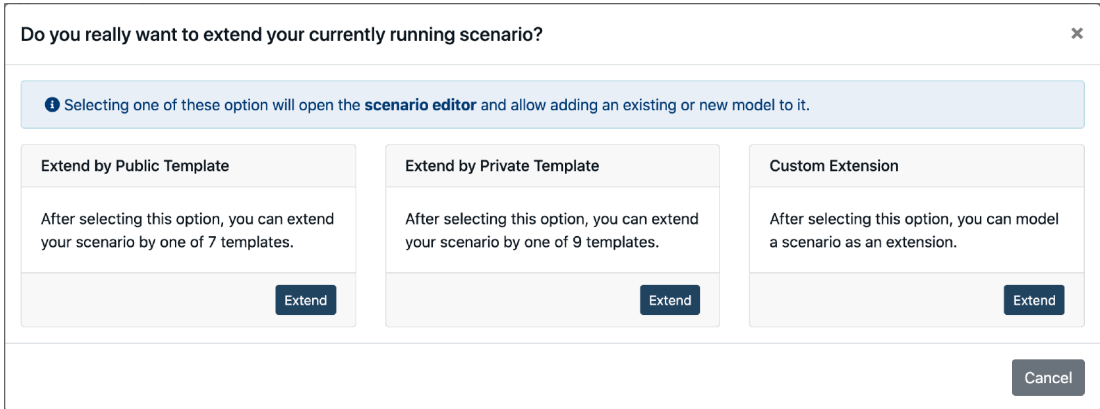
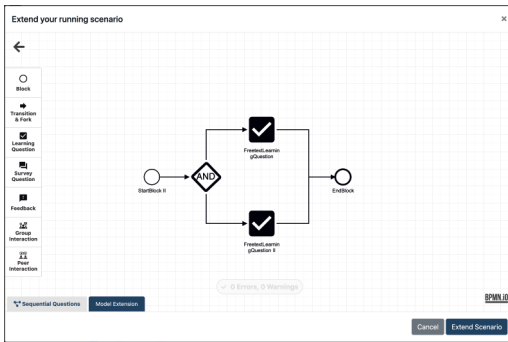
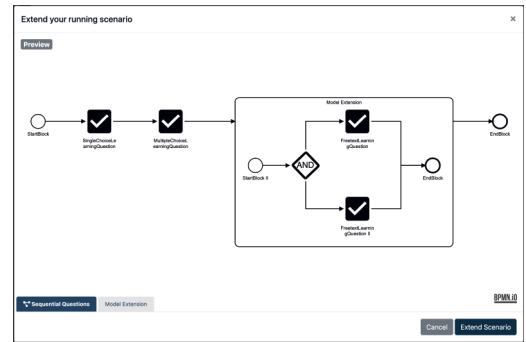


Figure 5.20: The extension dialog to select one of three options to extend the scenario.

In any case, the extension will be made within a sub-process that is integrated into the existing model, as visualized in figure 5.21.



(a) The sub-process view that is opened on extension, in which a scenario of two parallel *Free-text Learning Questions* is modeled.



(b) The general tab, in which the entire scenario is previewed, but no editing is possible.

Figure 5.21: The two views of the editor to extend the scenario, which can be switched using the tabs on the bottom left.

After pressing "Extend Scenario," the validity of the model is checked and a preview of the model is presented. Moreover, a hint is given that the sub-process is integrated when confirming the extension. This preview dialog is visualized in figure 5.22.

When confirming the preview dialog, the extension is loaded into the model, as visualized in figure 5.23. As described in the hint of the preview, the sub-process is integrated, i.e., the container and both *StartBlock* and *EndBlock(s)* are removed and connected accordingly. Furthermore, an "Undo Extension" button is added, which allows loading the state of the model before the extension took place. However, this button is removed as soon as a block is finished, as a block of the extension might be active afterward.

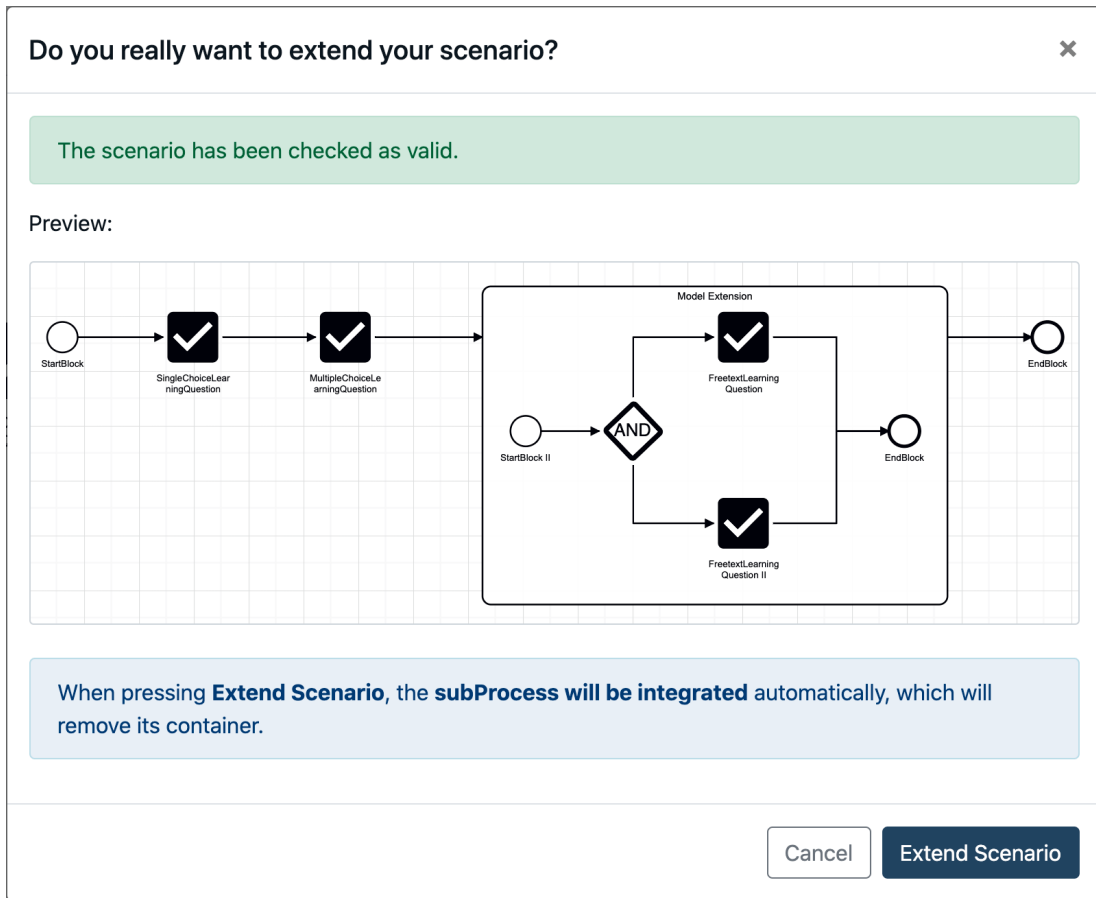


Figure 5.22: The extension preview dialog, in which the model is checked, previewed and a hint regarding the integration is given.

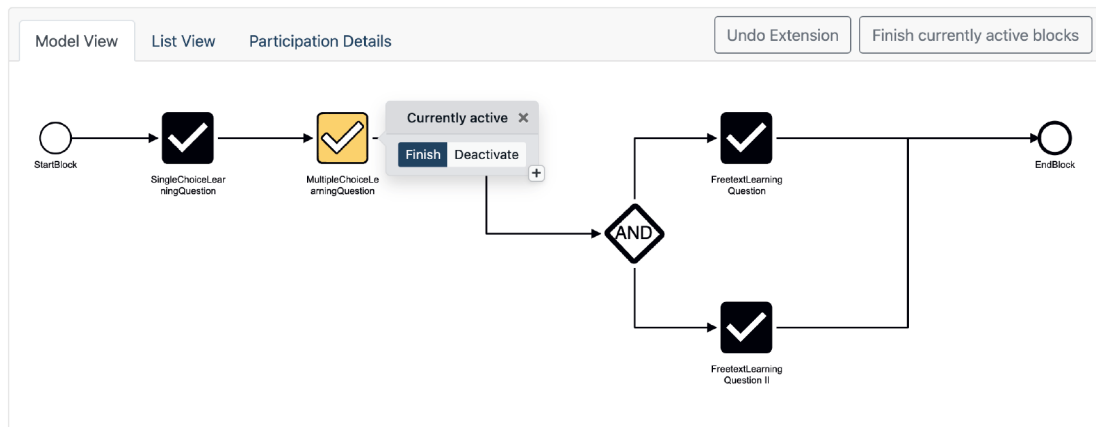


Figure 5.23: The extended scenario in which the previously created sub-process is integrated. Moreover, an option to undo the last extension is provided.

The evaluation has to show whether the extension of scenarios is sufficient in order to make changes to running scenarios or whether further functions are demanded. Nevertheless, we believe that the current implementation provides a good starting point, as it ensures that the extended scenario always remains valid.

5.7 Example Scenarios

This section presents the stARS-models of several didactic strategies that were described in section 2.1: *Interactive Learning Questions*, *Peer Instruction*, *Jigsaw Classroom*, *Think-Pair-Share*, *Learning Stations*, *Peer Feedback*, as well as *Learners-as-Designers*.

5.7.1 Interactive Learning Questions

A used practice is the integration of interactive learning questions at the beginning, the middle and the end of the lecture. In figure 5.24, a possible representation is visualized, whose execution is described in the following.

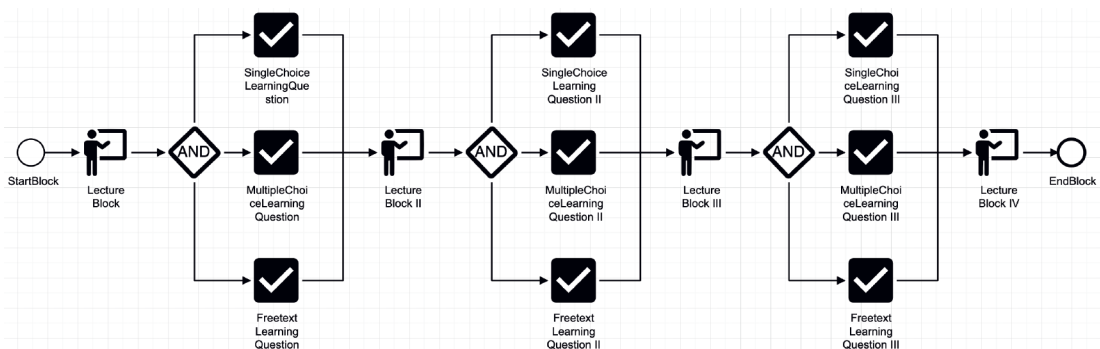


Figure 5.24: A possible representation of a lecture that is supported by *Interactive Learning Questions* at the beginning, the middle and the end of the lecture.

The start of the lecture is represented by the *StartBlock*. Next, a *LectureBlock* describes, for instance, the introduction to the lecture as well as the repetition of a previous lecture. This repetition is followed by three parallel interactive learning questions that allow students to check their gained knowledge as well as the lecturer to identify potential misunderstandings. Afterward, another *LectureBlock* is used to describe the presentation of the first topic of the lecture. This is again followed by three parallel interactive learning questions. The same procedure is used for the second topic. The

lecture ends with a final *LectureBlock*, in which a summary of the lecture is given, as well as the *EndBlock* of the scenario.

5.7.2 Peer Instruction

A well-known method that is often supported by *digital learning environments* is *Peer Instruction*. While most of the systems concentrate on supporting the *conceptTest* by *Interactive Learning Questions*, stARS allows supporting both *conceptTest* and *peer discussion*. A possible representation of one iteration³⁸ of a variant of *Peer Instruction* is visualized in figure 5.25 and will be described in the following.

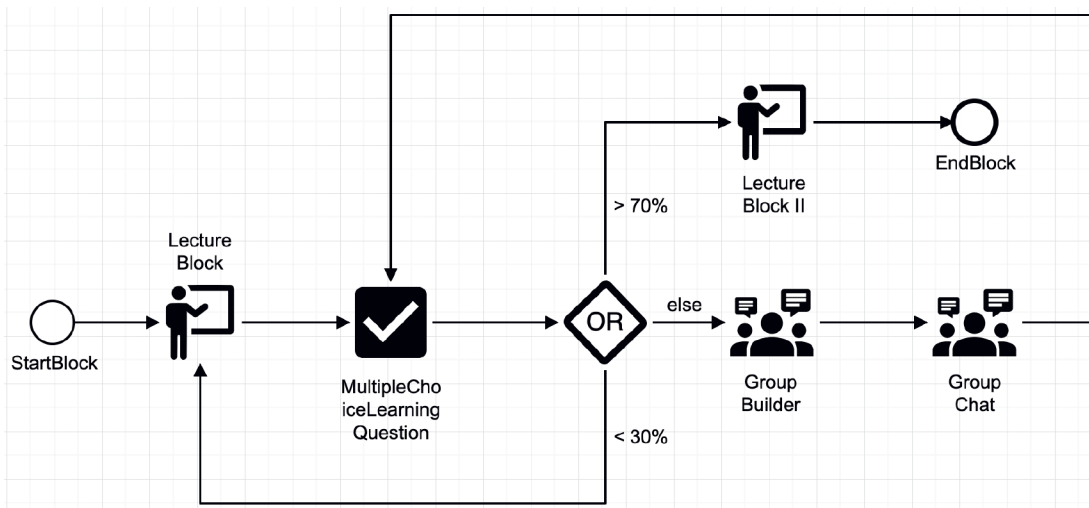


Figure 5.25: A possible representation of one iteration of a variant of *Peer Instruction* that allows supporting both *conceptTest* and *peer discussion*.

The start of this lecture part is represented by the *StartBlock*. This is followed by a *LectureBlock*, which describes the *brief lecture* of *Peer Instruction*. Next, the *conceptTest* is represented by a *MultipleChoiceLearningQuestion*, in which students have to select multiple correct answers. Although not visible in the graphical model, the parameter *answerFeedback* is set to *false*, as the possible correct answer should not be revealed to the students. Afterward, a conditional progression is described that will later be executed automatically: If the majority of students answered the previous *conceptTest* correctly (i.e., $> 70\%$), the lecture is concluded by a *LectureBlock* and proceeds to its end. If the minority of students answered correctly (i.e., $< 30\%$), the *brief lecture* has to be repeated and held in another version. In all other cases (i.e., *else*, or $30\% - 70\%$), a *peer discussion* is

³⁸ Within a 90-minute lecture, up to four iterations of *Peer Instruction* are used.

executed. Therefore, the students are grouped (with their peers) using the *GroupBuilder*. As a parameter, the *buildSchema* is set to *differentAnswer* in order to group students that gave different answers on the *conceptTest*. The *GroupBuilder* is followed by a *GroupChat*, which allows students within a group to exchange their opinions textually. The goal is to convince each other that the personally chosen answer is the correct one. After the *peer discussion* (i.e., *GroupBuilder* and *GroupChat*), the *conceptTest* is repeated.

Contrary to other approaches, stARS allows executing the *peer discussion* virtually, which is appropriate for both online or large classroom scenarios. The exchange of group members can either be realized by a *GroupChat* (i.e., textual) or a *GroupAudioVideoChat* (i.e., via audio and video), which uses a Jitsi integration.

In order to avoid endless loops, e.g., if students intentionally answer the *conceptTest* incorrectly, stARS allows to manually deactivate function blocks and continue the scenario by manually activating another function block. This way, it is possible to finish the repetition of the *brief lecture* or the *peer discussion* and continue with the *conclusion*.

5.7.3 Jigsaw Classroom

The *Jigsaw Classroom* relies on the collaboration between students. A possible representation of a lecture that integrates the *Jigsaw* teaching technique is visualized in figure 5.26 and will be described in the following.

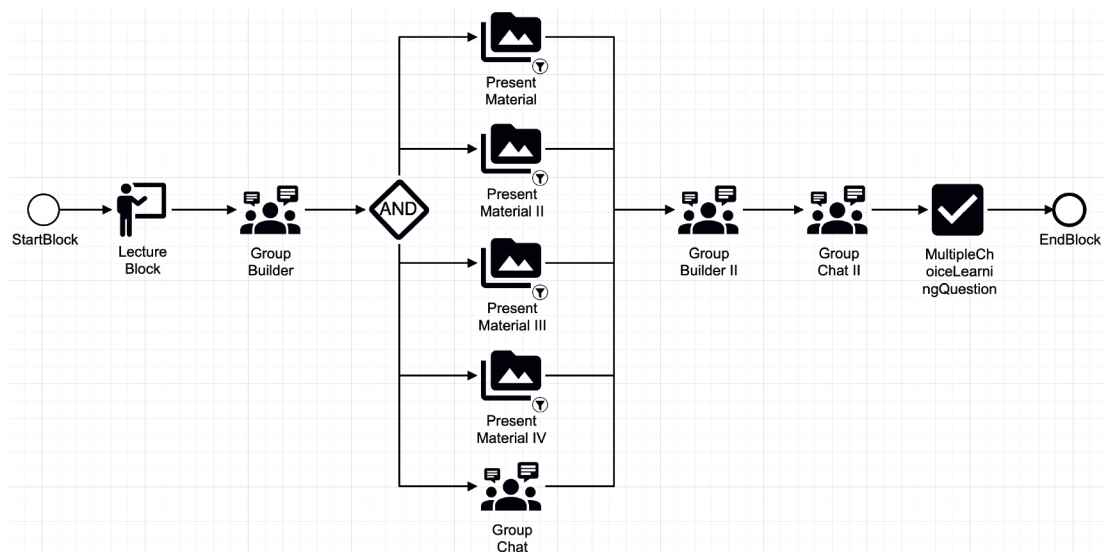


Figure 5.26: A possible representation of a lecture that integrates the *Jigsaw* teaching technique.

The lecture starts with a *StartBlock*. Next, a *LectureBlock* is added in which the procedure of the *Jigsaw* teaching technique is explained to the students. Afterward, *expert groups* are formed. In our example, the parameter *numberOfGroups* is set to 4, which results in four groups being created. Then, each group is presented with different materials, as visualized by the four *PresentMaterial* blocks. Each of these blocks has been set a parameter with the name *filter* that defines conditions under which a certain block is displayed. In this example, the first *PresentMaterial* has set the *filter* to *ConcreteGroupNumber* with a value of 1, which unlocks it for the first group. The other three *PresentMaterial* blocks have a similar *filter* defined but with another value (i.e., group number). In addition to the display of material, a *GroupChat* is displayed, in which each expert group can discuss the assigned topic. In the next step, another *GroupBuilder* follows that refers to the first *GroupBuilder*. This is done by setting the parameter *buildSchema* to *GroupShuffle* and defining a reference to the other *GroupBuilder*. This way, groups are built that contain one group member of each expert group. In the following, these groups can exchange their lessons learned in a *GroupChat*. The lecture ends with a *MultipleChoiceLearningQuestion*, in which all four topics have to be applied.

5.7.4 Think-Pair-Share

Think-Pair-Share is another didactic scenario that builds on the idea of collaboration between students. As can be seen in figure 5.27, the scenario includes several collaborative activities and is executed as described in the following.

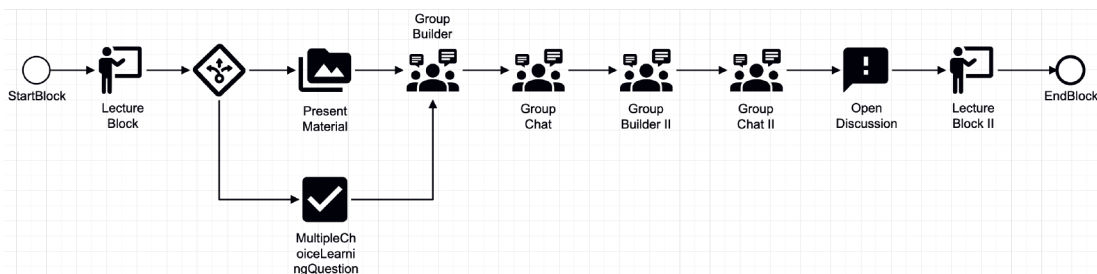


Figure 5.27: A possible representation of a lecture that integrates *Think-Pair-Share*.

The lecture starts with the *StartBlock*. Next, the procedure of the lecture is explained before a manual decision has been made on whether a material (e.g., a sample case) or an *Interactive Learning Question* is presented. Then, the students are grouped into pairs using the *GroupBuilder* with the parameter *groupSize* set to 2. Afterward, the students get the ability to discuss within these groups. In the next step, another *GroupBuilder* is added that merges the previously created groups by setting the parameter *buildSchema*

to *GroupMerge*, and another exchange within these groups follows. Finally, an *OpenDiscussion* allows all students to discuss issues that were not yet answered by the group interactions before a *LectureBlock* summarizes the lecture and enables the lecturer to clarify open problems (that were not answered within the previous discussion). The lecture ends with the *EndBlock*.

5.7.5 Learning Stations

Learning Stations are used to allow different groups of students to visit different stations and thus, enable them to learn in smaller groups. A straightforward representation of a lecture with four *Learning Stations* is visualized in figure 5.28 and is executed as follows.

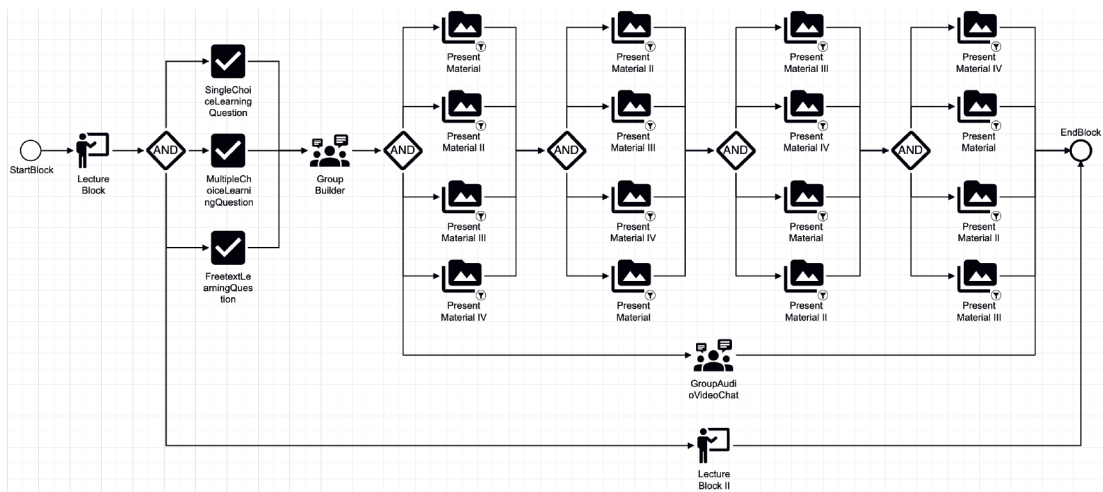


Figure 5.28: A possible representation of a lecture in which four *Learning Stations* are visited by different groups of students.

The lecture starts with the *StartBlock*. Next, a *LectureBlock* is used to allow describing the procedure of the lecture to the students. Afterward, several *Interactive Learning Questions* are displayed, as well as a parallel *LectureBlock* is activated in order to give the lecturer the ability to interrupt the students, if necessary. The *Interactive Learning Questions* are followed by a *GroupBuilder* that takes into account the results of these questions to form groups of students. In our example, the parameter *buildSchema* is set to *bestToWorst* to form groups of students with different prior knowledge. Moreover, the parameter *numberOfGroups* is set to 4, as four stations are present in the example. The *GroupBuilder* is followed by four *PresentMaterial* blocks as well as a *GroupAudioVideoChat*, which allows the group members to discuss the current station. Each of the four *PresentMaterial* blocks is only shown to one of the groups by setting the parameter *filter*

to a *ConcreteGroupNumber*. This is repeated four times in order to give each group the ability to visit each station once.

As the visualization presented in figure 5.28 includes repetitive blocks, an optimized and more flexible representation is proposed in figure 5.29 and works as follows.

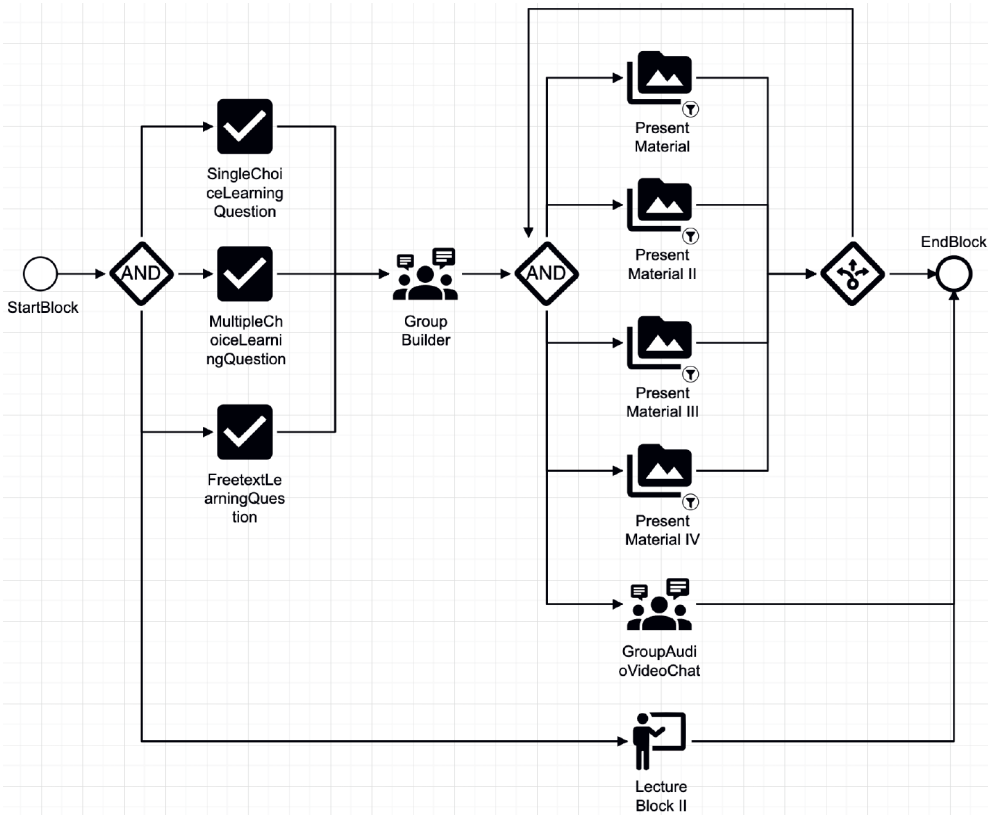


Figure 5.29: An optimized and more flexible representation of a lecture, in which four *Learning Stations* are visited by different groups of students.

While the initial blocks work as described in the scenario presented above, the *Present-Material* blocks are only displayed once and assigned the filter *GroupNumberForIteration*. This filter unlocks the block to a defined group number in the first iteration and increases the group number with each following iteration. At the same time, it takes into account the number of groups and thus proceeds with the first block if the resulting number is equal to *numberOfGroups* + 1. The iteration of these blocks is modeled by a *DecisionFork*, which allows to manually decide whether to proceed with the next station or end the lecture. This enables much more flexibility, as in real scenarios, not every station is visited by every group (e.g., due to time constraints).

5.7.6 Peer Feedback

Peer Feedback provides a promising opportunity in order to give students feedback on a task they had to solve. The complexity of this task can vary from simple questions that require short textual answers to very complex tasks, such as a seminar paper that had to be written on a specific topic. A possible representation of using *Peer Feedback* to implement the seminar use case is visualized in figure 5.30 and works as follows.

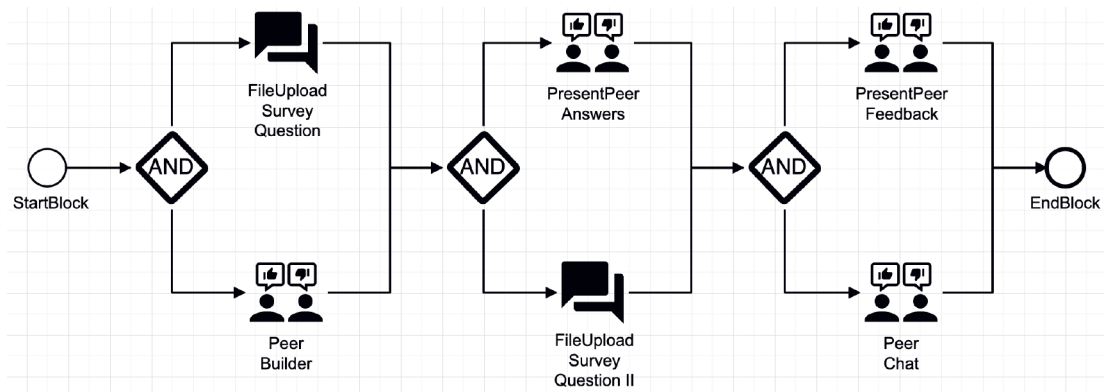


Figure 5.30: A possible representation of a *Peer Feedback* scenario that spans over multiple days.

In general, there exist three main stages that are visualized by two parallel blocks each. In the first stage, the students have to upload their seminar paper (represented by the *FileUploadSurveyQuestion*) and acknowledge that they want to join the *Peer Feedback* process (*PeerBuilder*), which is a crucial requirement in order to create the assignments. Within the *PeerBuilder*, different options can be set, e.g., in this use case, a *numberOfAssignments* of 2 is specified in order to provide each student with two feedbacks. Once the deadline for submission has passed, the lecturer can finish the first stage. This will automatically create the assignments. Unlike the *GroupBuilder*, two students do not necessarily have to give feedback to each other – instead, it may happen that, for instance, student A provides feedback to student B, student B to student C, and student C to student A. This is determined by the *buildSchema* that was defined. In the second stage, each student can see the assignments to which he/she is asked to give feedback (represented by *PresentPeerAnswers*). Additionally, another *FileUploadSurveyQuestion*, which refers to the *PeerBuilder* from stage one, enables them to upload a file that contains the *Peer Feedback* for each assignment. Finally, after the deadline for submitting the feedback has passed, the third stage can be unlocked, in which the received feedback can be seen by the students (represented by *PresentPeerFeedback*). Moreover, a *PeerChat* is added that allows sending questions to the provider of the feedback in order to clarify potential misconceptions.

While the described workflow takes multiple days to complete, there might also be use cases in which live feedback should be provided. Therefore, it is allowed to give feedback for and use any type of *SurveyQuestion*: For example, it could be used to let students define a term using their own words (*FreetextSurveyQuestion*) that is then rated by another student using grades (*SingleChoiceSurveyQuestion* with the choices 1, 2, 3, etc.), which motivates the flexibility of the entire approach.

5.7.7 Learners-as-Designers

Learners-as-Designers describes a learning approach in which “students are encouraged to design and produce media for other learners” [Jon96]. This student-centered approach usually spans an entire semester and consists of different stages executed by groups of students that can be repeated depending on the progress made. A possible representation of this use case is visualized in figure 5.31 and runs as follows.

Note on the Usage of ActivityBlocks

While the previous models used *PresentMaterial* to describe situations in which students receive a task that should be solved, this model uses the *ActivityBlock* instead. However, both blocks act similarly, even though the semantics are quite different. Moreover, an *ActivityBlock* can define an *audioVideoChat* to be used. However, the decision of using *PresentMaterial* or *ActivityBlock* should be made by the lecturer, as both are valid and reasonable.

First, students can choose a topic on which they want to create their media – in our example, a digital wallpaper has to be created. Afterward, the *GroupBuilder* is used to create a specified number of groups that has to be similar to the number of individual paths that follow (independently from the *GroupChat* that is active for all groups for multiple stages). In this example, the *numberOfGroups* is set to 3, as three groups of students should be created. In addition, the *buildSchema sameAnswer* is used to group students that selected the same topic. However, there might also be students that have to work on another topic that they did not select, as an optimal group formation (i.e., every student gets the topic he/she selected) strongly depends on the given answers. Next, for each group, a number of stages follow (that all have set the filter *groupNumber* to a specific group number): The first stage is the *Planning*-phase, in which, for example, the roles within the group are assigned, or a timeline is created. In our example, this stage is active for one day by setting the *timeout* accordingly and switching the parameter *autoFinishAfterTimeout* to *true*, which will automatically unlock the next step after the

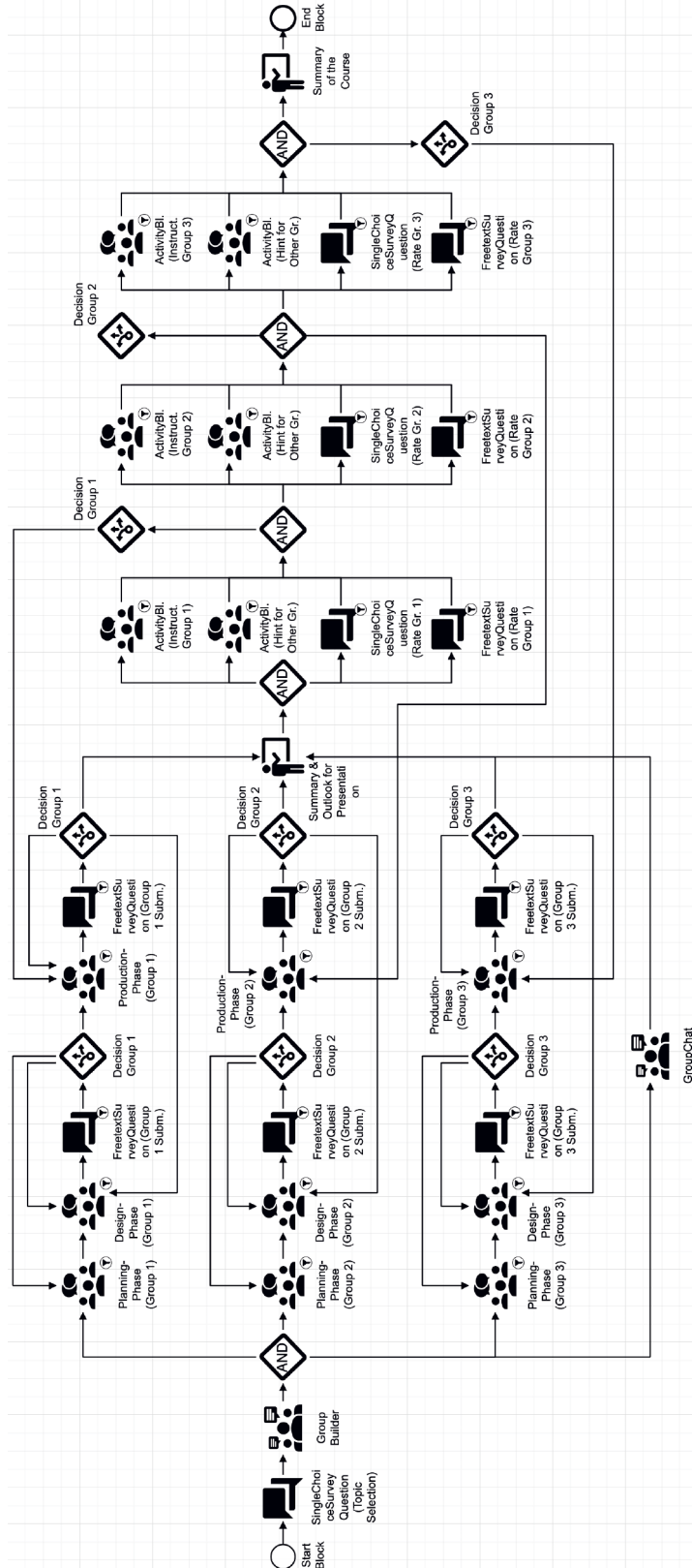


Figure 5.31: A possible representation of a student-centered *Learners-as-Designers* scenario that spans an entire semester.

timeout has finished. This stage is followed by the *Design*-phase, in which, for example, information is searched and segmented, or a structure of the project is created. After completing this stage, a *FretextSurveyQuestion* follows, in which one group member (the parameter *filter* is set to *positionOfAGroupMember* equal 1) submits the design of his/her group (which is in our use case a link to a platform allowing collaborative editing such as Padlet³⁹, Miro⁴⁰ or Trello⁴¹). Afterward, the lecturer has to decide whether both the *Planning* and *Design* of the group were successful or whether one of these stages has to be repeated. Unless there is a need for revision, the *Production*-stage follows, in which different representations (e.g., texts or HTML documents) are created. Again, another *FretextSurveyQuestion* follows, in which a link to the product has to be input and the lecturer decides whether the *Production*-stage must be repeated. If a revision is not required, the group can wait until all other groups have finished, too. Then, the lecturer summarizes the overall progress and gives an outlook on the *Presentation*-stage. This is structured as follows: For the presentation of each group, four parallel blocks exist. The first *ActivityBlock* gives instructions to the currently presenting group (the *filter* is set to a specific *groupNumber*). The second *ActivityBlock* informs all other groups which group is currently presenting (the same but negated filter from the first block is used). Additionally, two questions are shown to these groups, in which the presentation can be rated (same *filter* as for the second block). After completing the presentation, the lecturer uses the results to decide whether this group has to return to the *Production*-stage. In any case, the next presentation follows. This is done sequentially until each group has presented. Finally, a summary of the course is given.

Although using *DecisionForks* provides a lot of flexibility (due to individual decisions that can be made) and a variant of a *Learners-as-Designers* use case could be implemented at a certain degree, the adaptation at runtime is a crucial requirement for successfully implementing such a complex and student-centered approach. For example, it could be required to add a hint after the *Design*-stage, which provides further information on the schedule, or a *SurveyQuestion* after the *Production*-stage, in which the groups are asked whether the proposed time frame is sufficient. Moreover, it has to be noted that the general structure of this use case strongly depends on the students themselves. For example, it is not possible to estimate how much material students will generate or even how long a particular presentation will take. Thus, depending on the student's progress, it might be necessary to schedule a presentation after the *Design*-phase, which is not visualized in the model itself, as it does not make sense to model every single

³⁹ <https://en.padlet.com/> – last successful access on October 8, 2021

⁴⁰ <https://miro.com/en/> – last successful access on October 8, 2021

⁴¹ <https://trello.com/en> – last successful access on October 8, 2021

potential path in advance. Instead, a private template could be created that is then integrated into the running model if required. Furthermore, the usage of the manual activation and deactivation of blocks might be important, as depending on the time left, it might be the case that not every group receives the ability to present their product. Thus, some blocks have to be skipped.

Even if the modeled scenario is sufficient for a specific use case, limitations may still occur. For example, if the lecturer decides that a group has to revisit the *Planning*-stage, it might be required to adjust the parameters for this iteration (e.g., the *timeout* could be reduced) or add blocks for assistance (e.g., providing hints to the students using *PresentMaterial*). In this case, using adaptation at runtime is essential.

It is even possible to realize all the decisions that are currently made in the *DecisionForks* using runtime adaptation, as the scenario itself normally spans an entire semester and thus, the extensions can be made without time pressure (such as in a 90-minute lecture). This can be done as follows: Each stage that might be repeated could be created as a private template. For example, a template could exist, which includes both the *Planning*- and *Design*-phase as well as the *FreetextSurveyQuestion*, or only the *Design*-phase and the *FreetextSurveyQuestion*. In any case, when loading a template in the extension editor, it can also be adjusted freely, as described in section 5.6. Furthermore, even essential blocks such as the *GroupBuilder* or the different paths for the groups could be added after completing the *SingleChoiceSurveyQuestion* – this way, a suitable *numberOfGroups* could be selected, and thus, subsequent paths could be modeled. Finally, even the *Presentation*-phase could be structured more flexible, as depending on the time left, another presentation could be added or not. However, both ways of realizing a *Learners-as-Designers* use case are possible, i.e., having a predefined model, in which the general decisions are modeled using *DecisionForks* that can be extended at runtime if required, and building the entire model at runtime, depending on the incoming results.

Note on the Extendability of our Approach

In this section, we showed that our approach is able to implement different well-known didactic scenarios, whose support was added during the course of this thesis. This visualizes the extendability of our presented concept (cf. *NFR3*).

5.8 Summary

In this chapter, the prototypical implementation of an adaptable collaborative learning environment called *scenario-tailored Audience Response System*⁴² was described that allows lecturers to integrate technical tools in their individual teaching scenarios. In the following, the steps necessary to support a customized teaching scenario are briefly summarized:

First, the lecturer has to register a user with the role *Lecturer*, if not already done. Therefore, during sign-in, the “Lecturer Registration” has to be clicked and an arbitrary pseudonym (i.e., username) and password have to be selected. Specifying an email address is not necessary for the moment.

After signing in, a new scenario can be created by opening the scenario editor using the “Create Scenario” button. When opening the editor for the first time, an overlay is displayed describing the main functions and explaining the creation of a scenario (*Peer Instruction*) step by step.

The graphical editor allows the modeling of customized teaching scenarios. Therefore, each step of the lecture, i.e., a traditional presentation or an interactive activity, has to be represented by a block. Moreover, different types of transitions are used to realize the parallel, conditional, or manually decided progression of the lecture. In addition, a variety of parameters can be defined to also customize individual functions.

After completing the modeling, the scenario has to be saved. Therefore, either the button in the main menu or the shortcut (Ctrl + S) has to be pressed. Afterward, a preview of the scenario is displayed and the name of the scenario can be changed again. After confirming the saving, an instance of this scenario is created in the backend.

A list of all currently existing scenarios of the lecturer is displayed in the dashboard, from which they can be edited or deleted. Moreover, when pressing “Enter Scenario,” the scenario view is opened.

Within this scenario view, the instance can be started, which will create a docker container with the specified functional scope and make it available to be used. Moreover, the first block of the workflow is selected, i.e., the *StartBlock* of the scenario.

⁴² <https://stars-project.com/> – last successful access on October 8, 2021

The started scenario can now be managed by finishing one specific or all currently active blocks. Moreover, blocks can be manually activated or deactivated. All managing options for individual blocks are provided either in the model view or the list view.

Moreover, another view displaying the participation details is available, in which both a QR code and a link are presented. This view can be shared with the students (e.g., by sharing the screen), who can then join the scenario. Depending on the defined access control in the scenario, a pseudonym and password have to be input or not. After joining a specific scenario, the students can interact with the currently active block(s), e.g., answering survey or learning questions, or collaborating within groups. The view adapts as soon as the currently active blocks are changing.

The lecturer is presented with the results of the currently active blocks and thus, can discuss those with the students by sharing his/her screen. Moreover, he/she has different options in specific blocks, e.g., when a *GroupVoting* is active, the lecturer can manually assign the moderators of the groups by pressing a button.

In addition, the lecturer can manage “global” roles in order to give students additional privileges, such as managing the workflow or retrieving the results.

Furthermore, the lecturer can adapt the currently running scenario, i.e., he/she can extend the scenario. Therefore, a block has to be selected from which the scenario can either be extended by a private or public template, or by a new model. After a successful extension, the model can be used instantly without pausing or stopping (i.e., resetting) the scenario, which are two other options to choose from.

Finally, as soon as the scenario proceeds to an *EndBlock* and thus, terminates, the lecturer can export the results as a text file.

6 Evaluation

This chapter presents the evaluation of our approach, which was designed to validate the research theses presented in section 3.5.

Note on the Structure of this Chapter

The chapter starts with a brief summary of both conducted user studies (cf. section 6.1) and lecture experiments (cf. section 6.2). Afterward, the structure is based on the research theses derived in section 3.5, which are to be proven or disproven. Therefore, the sub-theses of each research thesis are repeated before each of them is discussed in a separate subsection. The contents of this chapter will later be used in chapter 7 to answer the overall research theses and, thus, the research question that was formulated for this thesis.

6.1 User Studies

In the course of this thesis, a variety of user studies were carried out as a part of students' work, e.g., as described in subsection 4.2.6 (the preliminary evaluation of the (meta-)model) or subsection 4.3.2 (the preliminary evaluation(s) of the graphical editor). As these evaluations were strongly focused on specific concepts and functionalities, three further user studies were conducted that were targeted to verify the overall research theses (cf. section 3.5).

In the following, these user studies are briefly described. However, conclusions on the results that consider the research theses will be drawn later in section 6.3, section 6.4 and section 6.5.

The contents of this section are partly published in [Kub+21].

6.1.1 User Study on Usability (Mid 2020)

Note on this User Study

This user study was supported by a questionnaire that can be retrieved in the Appendix C.2. Results that are not relevant for answering the research theses might be omitted.

The first user study¹ was conducted in mid-2020 after implementing the first prototype of stARS (that was published in [Kub+20b]). The goal was to check the lecturer's ability to use the graphical editor to create customized teaching scenarios. Due to the CoViD-19 pandemic, the study was conducted in virtual live sessions. A total of 20 lecturers participated.

Each session started with a short presentation (cf. Appendix C.1), in which the general approach of stARS was briefly introduced, and the structure of the overall evaluation was presented. The participant was told that if there are any open questions, he/she can ask at any time and was also thanked for his/her agreement to participate in the study. Afterward, the questionnaire (cf. Appendix C.2) was shared, and the participant was asked to answer the first part of it that included questions regarding his/her personal teaching strategies and prior knowledge on both teaching and stARS.

10 participants stated that they teach over ten years, 2 between five and ten years, 7 between one and three years, and 1 participant said that he/she teaches until one year. Regarding the technical support of lectures, 10 participants said that they use tools to activate students (e.g., *Audience Response Systems*), and 15 that they use questions for the preparation and post-processing of lectures. The concept of stARS was unknown to 10 participants, while 9 were already familiar with the (meta-)model and 7 with the graphical editor of stARS.

In the next part of the user study, the participants had to solve two modeling tasks: First, a simple scenario with three lecture blocks, whose presentation is supported by two rounds of questions, had to be modeled. Second, a more complex scenario had to be created, namely, a possible representation of the teaching method *Peer Instruction*. The participants were asked to share their screens, which allowed us to observe potential difficulties during usage. The tasks could either be retrieved from the questionnaire

¹ This user study was part of a larger study that included the evaluation of preliminary results from the master's thesis of Sinthujan Thanabalasingam (cf. [Tha21]) and a practical course of Lidia Roszko. While the overall study was conducted in two phases, the part that is described in this thesis remained exactly the same. Thus, it is described as a single user study.

or were presented orally (if no second screen was available). Depending on the prior knowledge of stARS, hints were given on how to solve the task. After each task, the result was analyzed and possible errors were discussed with the participant. The modeling tasks were followed by a *System Usability Scale (SUS)* in order to evaluate the perceived usability of the graphical editor. Therefore, the participants were asked to stop sharing their screen and answer the 10 questions of the SUS with regard to the sub-system of the *graphical editor*. Furthermore, the participants were able to give open feedback on what they considered positive or negative.

Every participant was able to solve the modeling tasks, although some required help, while others did not. It was particularly noticeable that some lecturers had difficulties in understanding the general approach, which made it hard for them to get started. However, after the idea of representing each part and activity of the lecture as a block was clear to them (which was mostly the case after completing the first task), they used the editor quite efficiently and solved the second task with fewer difficulties.

These findings are also recognizable when investigating the results of the SUS that are shown in table 6.1. For example, Q7, which is formulated as follows (translated into English): “I would imagine that most people would learn to use this system very quickly,” was rated by 7 participants as *neither agree nor disagree*, meaning that they are unsure if the participants are able to do so. Moreover, for Q10, which is formulated as follows (translated into English): “I needed to learn a lot of things before I could get going with this system,” the opinion of the participants differs clearly. A total of 6 participants gave a rating of 2 or below, meaning that they partly or fully agree with the statement and thus, think that a lot of things have to be learned before using the system. This is also noticeable when investigating the SUS score that ranges from 40% to 92,5%, showing that the opinion of the participants varies significantly.

For an interpretation of the SUS scores, the results of [BKM09] are used: In their experiment, the authors describe the addition of a seven-point adjective-anchored Likert scale as an eleventh question to nearly 1000 SUS surveys. This allowed for an interpretation of the scores, as shown in figure 6.1.

Considering the adjective ratings of [BKM09], the results of this user study can be interpreted as follows: Only one participant gave a rating that indicates poor usability (i.e., below 50,9%). However, the majority of the participants (12 out of 20) rated the graphical editor with a score above 71,4% (i.e., good usability). On average, a SUS score of 73,9% was submitted, which is just above the threshold for good usability. This result is even the same when comparing the group without prior knowledge on stARS with

Table 6.1: The results of the System Usability Scale for each participant (those without prior knowledge on stARS are marked bold). Each item can get a rating between 0 to 4.

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS Score
P1	2	3	2	1	3	2	3	2	2	1	52,5
P2	4	3	3	3	3	4	3	3	3	3	80,0
P3	3	2	3	3	2	2	4	3	3	3	70,0
P4	3	2	2	3	3	2	3	3	2	4	67,5
P5	4	4	3	4	4	3	3	4	4	4	92,5
P6	3	4	4	4	4	4	3	4	2	2	85,0
P7	3	4	4	4	3	3	4	4	4	4	92,5
P8	1	1	2	1	2	1	2	1	2	3	40,0
P9	4	3	3	3	4	4	3	3	3	2	80,0
P10	3	1	2	1	3	4	2	3	3	2	60,0
P11	3	4	3	4	3	3	2	3	2	4	77,5
P12	2	4	3	3	3	3	3	4	3	3	77,5
P13	2	3	3	3	3	4	2	3	3	3	72,5
P14	3	4	3	4	3	4	2	3	4	3	82,5
P15	3	1	3	3	2	2	3	3	3	4	67,5
P16	3	2	3	2	3	2	3	3	3	1	62,5
P17	3	3	4	3	4	4	2	4	3	2	80,0
P18	4	4	4	2	3	4	3	4	4	3	87,5
P19	4	3	3	1	4	4	3	4	3	3	80,0
P20	3	3	3	3	2	4	2	2	3	3	70,0

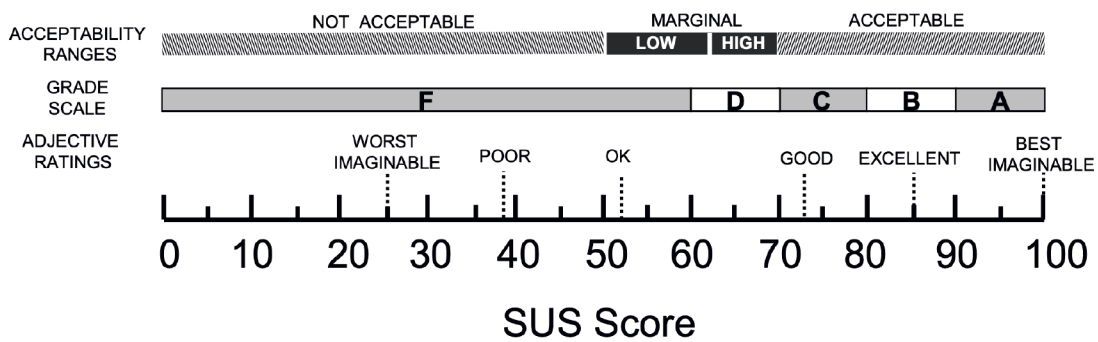


Figure 6.1: An interpretation of the SUS score into acceptability, school grades and adjectives (from top to bottom) [BKM09].

the group that is at least familiar with the (meta-)model or the graphical editor of it: Lecturers without prior experience on stARS even rated the editor better (75,5%) than those with experience (72,3%). Nevertheless, the difference is that small that the ratings of both groups can be considered similar.

By evaluating the textual feedback, the following statements can be made:

- Lecturers praised the idea of a graphical presentation that allows structuring their individual teaching scenarios.
- Moreover, they liked the ability to create teaching scenarios using drag-and-drop and position elements freely.
- However, some participants struggled to understand the conceptual idea and had problems finding the appropriate functions, as not every symbol was clear to them.
- In addition, the large manual effort required to create a scenario was criticized, e.g., when having to connect elements next to each other manually.
- Finally, the change of perspective was found to be incomprehensible, e.g., lecturers did not understand the representation of lecture presentations as (technical) *PauseBlocks*.

As a result, a number of changes were made to the graphical editor that were already described in section 4.3: For example, the icons of the *main menu* were adjusted, several components were added to support the lecturer in getting started more easily, an auto-connect function was implemented, as well as blocks to represent traditional parts of the lecture were added (namely a *LectureBlock* and an *ActivityBlock*).

6.1.2 User Study on Teaching Scenarios (Early 2021)

Note on this User Study

This user study was conducted using a questionnaire that can be retrieved in the Appendix D.1. Results that are not relevant for answering the research theses might be omitted.

The second user study was conducted at the beginning of 2021 after a variety of didactic scenarios were implemented in the prototype. It had several objectives:

1. First, it was intended to check whether lecturers expect the functionalities that are offered by stARS in a *digital learning environment* and also whether they expect further functionalities to be supported.
2. Next, it should be analyzed which teaching scenarios are known or even used by lecturers, which of them they would like to use more often, and which they would like to support technically. This should help to verify that the teaching scenarios supported by stARS are demanded by lecturers.
3. Third, it should be investigated whether lecturers are able to interpret different teaching scenarios that were represented in stARS. After having evaluated the graphical editor in subsection 6.1.1, this is intended to also analyze the resulting models for their comprehensibility.
4. Finally, it should be investigated whether lecturers consider the extension of teaching scenarios at runtime as relevant and whether they are interested in using the approach in their own lectures.

This user study was conducted using a questionnaire (cf. Appendix D.1) that was sent to lecturers and could be answered at any time. A total of 20 lecturers participated, while 1 lecturer quit after completing the first task of rating a model. Both lecturers with and without prior knowledge of stARS participated (see details below).

In the first part of the questionnaire, general questions regarding lecturers' experience in teaching as well as with *digital learning environments* and stARS were asked. 12 lecturers stated that they teach over ten years, 1 between five and ten years, 1 between three and five years, 5 between one and three years, and 1 participant said that he/she teaches until one year. Regarding the prior experience with *digital learning environments*, 8 participants agreed that they feel confident when using those, 9 partly agreed, 2 stated that they feel neither confident nor unconfident, and 1 participant said that he/she feels partly unconfident. When considering the prior knowledge on stARS, 3 participants agreed that they are familiar with this concept, 4 did partly agree, 4 did neither agree nor disagree, 2 did partly not agree, and 7 participants stated that they were not familiar with this concept.

This part also included a question regarding expected functionalities of *digital learning environments* that was formulated as follows (translated into English): "What function(s) do you expect when supporting your lecture by a digital learning environment?". The participants were able to select an arbitrary amount of 11 functions and could also specify further ones textually. The results are shown in figure 6.2.

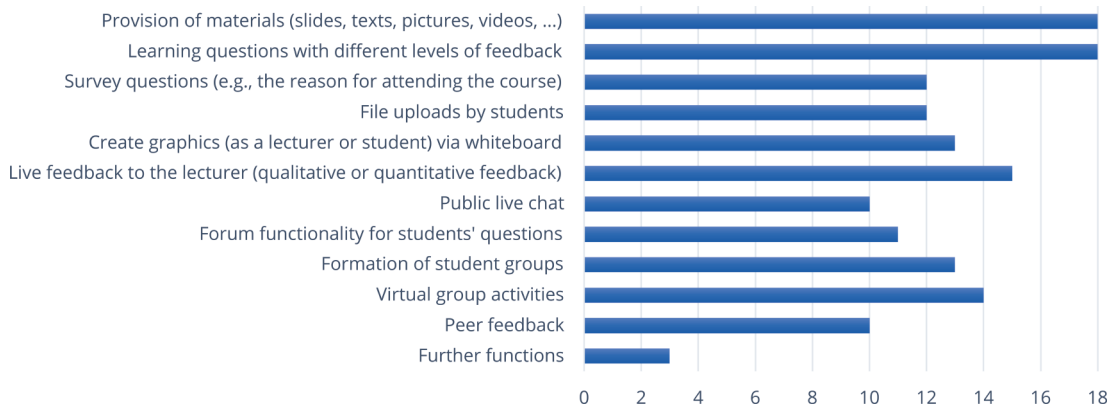


Figure 6.2: The results for lecturers' expected functionalities of *digital learning environments*.

It is remarkable that each of the proposed functions was rated by at least half of the participants as expected. This also holds for advanced functionalities (that are included in stARS), such as the *formation of student groups* (13), *virtual group activities* (14), and *peer feedback* (10). Moreover, only 3 lecturers proposed further functions to be expected, namely, the *collaboration of students on documents*, *peer review*², and *shared notes* (similar to Etherpad).

In the second part of the survey, the participants were presented with a list of 14 teaching scenarios and asked to select one of the following options for each (translated into English): "I do not know," "I know, but I do not use," "I have already used," or "I use regularly." Additionally, they could add further teaching scenarios (that they use regularly) in textual form. This should help to decide which of the teaching scenarios are actually used.

Note on the Selected Teaching Scenarios

The teaching scenarios were selected together with Dr. Gregor Damnik and Dr. Felix Kapp. It is important to note that this list does not rely on empirical data – however, scenarios were selected that we considered to be familiar ourselves. An explanation of the teaching scenarios that were not presented in subsection 2.1.2 can be retrieved from the overviews of [Rei12; Pro21].

This question was followed by two further questions, in which first, those scenarios should be selected that they would like to use more often, and second, the scenarios that they would like to support by a *digital learning environment*. Again, they were able

² *Peer review* is a term that frequently occurs in academic publishing processes for assessing the quality of papers.

to add teaching scenarios textually that they might miss. Both of these latter questions should help to select scenarios that are demanded by lecturers. The results of all three questions are summarized in table 6.2.

Table 6.2: A summary of lecturers' answers to their current usage, preference for future usage and opinion regarding technical support of different teaching scenarios (the individual questions are separated by vertical lines).

The following teaching scenario I do not know.	... I know, but I do not use.	... I have already used.	... I use regularly.	... I would like to use more often.	... I want to support technically.
Frontal Teaching	0	1	6	13	1	10
PowerPoint Presentations	0	0	2	18	3	11
Student Presentations	0	4	4	12	4	12
Interactive Learning Quest.	1	1	12	6	10	16
Peer Instruction	3	8	7	2	12	14
Jigsaw Classroom	4	12	4	0	4	7
Think-Pair-Share	8	7	3	2	9	10
Learning Stations	3	12	4	1	4	7
Classwide Discussion	0	6	6	8	4	6
World Cafe	9	7	4	0	4	5
Brainstorming	3	5	9	3	5	6
Problem-based Learning	1	5	9	5	9	9
Role Play	11	4	2	3	4	4
Cognitive Apprenticeship	1	15	1	3	3	5

Beyond the results visualized in table 6.2, several extensions were mentioned (that lecturers do currently use): The collaborative creation of documents (e.g., MindMaps), *Learners-as-Designers*, solving a problem step by step, and synchronize in groups with a direct discussion of any question that arises (similar to a *workshop*), as well as *project activity/work*. Moreover, *group work* was added when asking for didactic scenarios to be used more often.

By investigating the results of table 6.2 in more detail, the following statements can be made:

- Most participants already used *digital learning environments*, or more specifically, *Audience Response Systems*. 7 participants even used *Peer Instruction*, in which such questions (i.e., *Audience Response*) are combined with *peer discussions*.
- The most frequently used teaching scenarios are the traditional ones such as *Frontal Teaching*, *PowerPoint Presentations* and *Student Presentations*. However, only a few lecturers said that they would like to use them more often (i.e., between 1

and 4 lecturers depending on the scenario), and at least half of the participants demanded those to be supported technically.

- In contrast, 12 lecturers stated that they would like to use *Peer Instruction* more often. In addition, 10 selected *Interactive Learning Questions* and 9 each *Think-Pair-Share* and *Problem-based Learning*. All remaining scenarios were rated by less than 5 participants as wanting to be used more often.
- Similar, but even more interesting results can be recognized when analyzing the question regarding scenarios that lecturers would like to support technically through a *digital learning environment*: In addition to the traditional scenarios, both *Interactive Learning Questions* (16) and *Peer Instruction* (14) were selected frequently, especially when considering that *Peer Instruction* was unknown to 3 participants. Moreover, 10 out of 12 participants (that at least know *Think-Pair-Share*) want to support it technically. The scores for *Problem-based Learning* (9 out of 19), *Jigsaw Classroom* (7 out of 16) and *Learning Stations* (7 out of 17) are also quite interesting.

The third part of the questionnaire was about checking lecturers' ability to understand scenarios represented in stARS. Therefore, we selected five well-known teaching scenarios³ and modeled them, namely, *Interactive Learning Questions*, *Peer Instruction*, *Jigsaw Classroom*, *Think-Pair-Share*, as well as *Learning Stations*.

Note on the Models of the Teaching Scenarios

The models of these teaching scenarios can be retrieved in section 5.7. As can be recognized, no custom labels have been defined (i.e., the labels only contain the name of the element but do not describe the function of the block). This allows evaluating the comprehensibility of the model itself.

For each of these teaching scenarios, different steps were listed that are indirectly represented in the model (for further information on these steps, please refer to the Appendix D.1 (original German version) or [Kub+21] (English translation)). For sequences that contained loops, additional information was given (e.g., the percentage of correctly given answers), which allowed selecting the path to be executed. Moreover, for sequences that include parallel functional blocks, the stage that contains the *AndFork* also holds the information for the elements on the parallel path. These steps were added into five sorting questions, and lecturers were asked to sort them in the order that is

³ During the time of this evaluation, *Peer Feedback* and *Learners-as-Designers* were not yet implemented and are therefore not added in this evaluation.

represented in the model. This task should help to identify whether lecturers are able to understand the execution of the presented model. Furthermore, three questions were posted for each of the models: The first question was about the visualization of the model, the second question was regarding the understandability of the execution of it, and the final one allowed to give textual feedback.

In order to rate the answers of lecturers to these sorting questions, the following rating schema was used (the distinction between *nearly* and *partly* was made because mixing up two stages is less critical than placing a stage at the wrong position):

- *Full* states that all stages were sorted in the previously defined order (while the position of stages with the same name can vary),
- *nearly* means that two (groups of) stages were mixed up with each other,
- *partly* is used when stages were sorted incorrectly, but the remaining order is appropriate, and
- *incorrect* specifies that several stages were sorted incorrectly.

The questions regarding the intuitiveness of both the visualization and execution were answered using a 5-point Likert scale (having the options “does apply,” “rather does apply,” “neither nor,” “rather does not apply” and “does not apply”). As the formulation of one question was formulated oppositely, for evaluation purposes, the positive statement in each case (i.e., the participant agrees that the visualization/execution is intuitive) was rated with 5 points and the negative one with 1 point.

Note on the Number of Participants

The first sorting question was solved by 20 participants. However, 1 participant quit the questionnaire after completing this task. Thus, the following models were only evaluated by 19 participants.

The summarized results of this part of the evaluation are shown in table 6.3. As the summary of results may removes certain details, the results of every single participant for this part of the questionnaire can be retrieved in the Appendix D.2.

Table 6.3: The summarized results of the sorting questions as well as the questions regarding the intuitiveness of both graphical representation and execution.

The following teaching scenario was modeled <i>fully</i> correct.	... was modeled <i>nearly</i> correct.	... was modeled <i>partly</i> correct.	... was modeled <i>incorrect</i> has an intuitive graphical represent.	... is intuitive to execute.
Interact. Learn. Quest.	17	0	1	2	4,45	4,15
Peer Instruction	16	1	1	1	4,37	4,26
Jigsaw Classroom	10	6	3	0	4,11	4,32
Think-Pair-Share	14	0	5	0	3,68	3,95 ⁴
Learning Stations	15	2	2	0	3,47	3,63

In order to evaluate the results, it is also important to consider lecturers' prior knowledge of the specific teaching scenarios. Therefore, the number of participants that do not know a teaching scenario are listed in the following (as one participant quit the questionnaire after completing the first task, his/her answers are not listed for the remaining questions accordingly – thus, the results are slightly different to those presented in table 6.2): 1 participant stated that *Interactive Learning Questions* are unknown to him/her and 2 participants that they do not know *Peer Instruction*. Moreover, *Jigsaw Classroom* was unknown to 4 participants, *Think-Pair-Share* to 7 participants and *Learning Stations* to 2 participants.

While not every lecturer was familiar with every teaching scenario and the models were not adjusted by customized labels, most lecturers were still able to solve the sorting tasks properly. Although errors were made, in most cases, the order was either *nearly* (i.e., two (groups of) stages were mixed up with each other) or *partly* (i.e., up to three stages are misplaced, but the remaining order is accurate) correct. For example, in the model of the *Jigsaw Classroom*, the stages of building *learning groups* and *expert groups* were mixed up by 6 participants – however, the remaining task was solved properly. When considering the results of individual participants (cf. Appendix D.2), a correlation between the prior knowledge on stARS and solving the sorting tasks can be recognized: Especially those lecturers that are not or rather not familiar with the concept of stARS made the majority of mistakes when interpreting the models. However, only a total of 3 incorrect answers were retrieved among all questions and participants. These errors were specially made in the first two tasks, which could mean that the participants “learned” how to interpret the models while completing the questionnaire. Nevertheless, it could also mean that

⁴ This value slightly differs from the value published in [Kub+21]. Unfortunately, this error was detected after the final paper had already been submitted.

the participants tried to put the descriptions in a logical order and did not use the model for the interpretation at all.

Thus, the results of the remaining three questions are evaluated in further detail:

- The graphical representation of the first three models was rated quite positive, which is recognizable by an average rating above 4 (which indicates that the participants at least “rather agree” with the intuitiveness of the graphical representation).
- However, it is also obvious that this intuitiveness decreases with the increasing complexity of the models. In any case, the lowest rating is just between “neither agree nor disagree” and “rather agree,” which still tends in the direction of being intuitive.
- Nevertheless, the representation has to be improved. Thus, on the one side, an alternative representation of this model was proposed that is shown in figure 5.29. On the other side, several adjustments were made based on the textual feedback that was submitted by the participants: For example, the font size of models can now be adjusted in the settings of the editor, and comments for specific blocks can be defined that are visible in the graphical representation. Moreover, further adjustments are currently being investigated, e.g., the combination of elements into more abstract groups or the translation of “else” into a more intuitive representation.
- The values for the intuitiveness of the execution of scenarios vary less than those of the graphical representation and are mostly above or approximately 4, meaning that the participants “rather agree” on the intuitiveness of it.
- However, the same issue that occurred in the graphical representation is visible for the execution of models: As soon as the complexity of the execution increases (e.g., by defining *filters*), the intuitiveness of it decreases.

In any case, it could be shown that despite the fact that the scenarios were not known to every lecturer and the complexity of the models varied, several lecturers were still able to interpret those. We strongly believe that adding custom labels as well as suitable comments to these models⁵ helps to make them more easily to understand for further lecturers.

⁵ These models will be made accessible using *public templates*.

In the final part of the questionnaire, the participants were first asked for their opinion regarding the adjustment of didactic scenarios at runtime (e.g., in order to add questions during the ongoing lecture). 9 participants rated this functionality as “important,” 8 as “rather important,” and 2 as “neither important nor unimportant.” One participant also gave additional feedback on this functionality (translated into English): “The possibility to adjust a running lecture is quite important: e.g., if you want to interfere a group formation or want to adjust something. But I would always accept a short *stop* of the event in the system, edit it and then set it back to live. If this can be done well, it would even be better for the implementation, because then you cannot break anything *live*”.

Secondly, the participants were asked if they would like to use stARS in their lecture(s). 7 participants “agreed” that they would like to use it, 5 “partly agreed,” 3 did “neither agree nor disagree,” and 3 did “partly disagree.” Another participant abstained from answering, as he/she does no longer teach.

6.1.3 User Study on Runtime Adaptation (Mid 2021)

Note on this User Study

This user study was supported by a questionnaire that can be retrieved in the Appendix E.1. Results that are not relevant for answering the research theses might be omitted.

The third user study was carried out in mid-2021. Due to the CoViD-19 pandemic, it was conducted in virtual live sessions (similar to the first study). In total, 13 lecturers and students participated. The only prerequisite for participants was that they were already familiar with the general concept of stARS, as the function to be tested is a rather advanced one – more specifically, the adaptation at runtime. The objectives of the study were threefold:

- First, it had to be checked whether the participants are able to understand this functionality and can use it to extend a running scenario.
- Second, it should be assessed whether lecturers think that the extension of scenarios is sufficient to make changes to it or whether lecturers intend further changes to be made (such as editing parameters of existing blocks or deleting blocks).
- Finally, it should be evaluated whether suggestions about extensions that are made at runtime present a useful function or not.

The user study, or more specifically the questionnaire, was structured in eight parts, which will be briefly presented in the following paragraphs.

At the beginning, a short introduction was given, in which the structure of the remaining study was presented, an estimated time of 30 minutes was displayed and the participants were thanked for their attendance. Moreover, in order to allow studying “the cognitive problems that people have in learning to use a computer system” [Lew82], the participants were asked to *think aloud* (cf. [Pay94]). In this method, they should verbalize everything they are doing and thinking, what they are looking at, and what they are feeling while solving the tasks. This should help to identify issues as soon as they arise and find solutions to them.

In the next part, the prior experience of the participants with stARS, as well as their opinion regarding a function for adapting running scenarios, was evaluated. Regarding the general concept of stARS, 5 participants said that they are familiar with it, 4 that they are rather familiar, and 4 that they are neither familiar nor unfamiliar with this concept. Moreover, when asking for the functionality to load or import a scenario (which uses a similar dialog as the extension function), 11 participants said that they had already used it, while one said he/she did not, and another one that he/she is unsure about it. Furthermore, 7 participants stated that they had already executed a scenario in stARS, while 4 did not and 2 participants were unsure. Afterward, the participants were asked if they think that adapting running scenarios is relevant. 8 participants agreed with this, while the remaining 5 have rather agreed. In addition, a freetext question was added asking for their expectations as well as potential challenges for such a function. 5 participants stated that consistency could be a problem, i.e., that the state of the scenario still exists after the adaptation, without interrupting the scenario for the students. One participant specifically mentioned that the implementation of the user interface could be challenging. Another participant mentioned that it is important not to overwhelm the lecturer with such a functionality, as he/she must teach and use stARS at the same time. Regarding the expectations of the functionality, 2 participants mentioned the flexible adjustment of the scenario, while one participant specifically noted the extension by new blocks as well as the adaption of existing blocks. Furthermore, one participant only mentioned the extension by new blocks and another one the proposal of useful functions at runtime.

Afterward, the practical part of the user study started. Therefore, a short repetition of stARS was performed, in which the participants were asked to log in with given credentials, open the graphical editor, load a predefined template, save it as a scenario, navigate to this scenario and start it. This repetition was followed by four tasks that

had to be solved. Each task had a different goal to evaluate, i.e., a specific sub-function that had to be used or that had to be found. Thus, a specific question was added to verify this function. The remaining three questions were the same for all tasks: The first question asked whether solving the tasks was found to be easy, the second one whether the participant was satisfied with the result, and the third one allowed for open feedback. In the following, each of these four tasks is briefly summarized, and its individual results are presented before the results of the common questions are compared.

In the first task, the participants had to extend the scenario from a specified block by a *SingleChoiceSurveyQuestion*, in which the students can select one of three topics. This task was used in order to evaluate whether the function to extend the scenario was found by the participants. Thus, a question was added that is formulated as follows (translated into English): “I had difficulties in finding the function to extend the scenario.” 7 participants disagreed with this statement, 2 have rather disagreed, 2 have neither disagreed nor agreed, and one participant each has rather agreed and (fully) agreed. Using the rating for Likert scales that was already used in subsection 6.1.2 (a positive statement gets a rating of 5, while a negative one gets 1), an average value of 4 is retrieved⁶, which indicates that the participants rather had no problems in finding the extend function. However, one participant argued that he/she had found the button only because it was new in the interface. Although the button was intentionally quite discrete in terms of visibility, it could still be found by most participants.

In the second task, a *PresentResult*-block had to be added, which refers to a previous question. This task was used to evaluate whether the participants are confused by referring a block within a sub-process to the existing model (which is not visible when referring to the block, i.e., when setting the parameter). 6 participants stated that they were not confused, 3 were rather not confused, 2 were neither not confused nor confused, and one participant each was rather confused and (fully confused), which results in an average rating of 3.92. This indicates that the participants were rather not confused by this reference. Moreover, one participant stated that he/she would have expected an automatic reference to the previous question, which is, however, not always intended, as the *PresentResult*-block can refer to other questions, too.

The third task was about extending the scenario by a private template. The participants were asked whether they think that extending scenarios by private and public templates is useful. All of the 13 participants agreed with this statement.

⁶ In this case, the question was formulated oppositely, which had to be adjusted accordingly.

In the fourth task, the participants should undo the last extension and make another extension by a private template. Therefore, they were asked whether they consider the undo of the last extension to be useful. 10 participants agreed with this statement, while 2 have rather agreed and one participant has rather not agreed.

In table 6.4, the results of the question, whether a specific task could be solved easily, are summarized. It is noticeable that none of the participants disagreed with this statement. Moreover, even the lowest rating (3.85) indicates that the participants did rather found the task as easy to solve. This rating increased from one task to another as the participants got more familiar with the functionality as soon as they solved a task.

Table 6.4: The summarized results of the question, whether a task could be solved easily.

The task could be solved easily.	Agree	Rather Agree	Neither Nor	Rather Disagree	Disagree	Avg. Rating
Task 1	3	6	3	1	0	3.85
Task 2	7	3	1	2	0	4.15
Task 3	8	5	0	0	0	4.62
Task 4	11	0	2	0	0	4.69

Moreover, when considering the results of the question, whether the participants are satisfied with the results, a clear agreement becomes obvious, as listed in table 6.5.

Table 6.5: The summarized results of the question, whether the results are satisfactory.

The results are satisfactory.	Agree	Rather Agree	Neither Nor	Rather Disagree	Disagree	Avg. Rating
Task 1	11	2	0	0	0	4.85
Task 2	11	2	0	0	0	4.85
Task 3	10	2	1	0	0	4.69
Task 4	11	1	1	0	0	4.77

In general, all 13 participants were able to complete the user study and thus, solve the tasks, even though some of them experienced minor errors in the implementation that were fixed afterward. These issues included that the position of the sub-process was calculated incorrectly or that the parameters were not refreshed properly. Further improvements were made, such as the provision of the extension dialog for all existing blocks (which was previously limited to the currently active blocks) or the addition of a

confirmation dialog before closing. However, there exist further proposals that have to be investigated in the future, e.g., the automatic addition of *EndBlock(s)* in sub-processes.

In the final part of the evaluation, the participants were first asked whether a function to extend the scenario is sufficient to adapt the running scenario, or whether the participants want to adjust the scenario freely. 3 participants agreed that it is sufficient, 4 have partly agreed, and the remaining 6 have partly disagreed. The average value of 3.31 underlines the uncertainty of the participants and states that they have neither agreed nor disagreed. Moreover, several suggestions were made in the open feedback to improve this functionality. 8 participants stated that they would like to add a function to adjust the parameters of existing blocks, and 5 that they want to delete blocks as well. In addition, the extension of scenarios in advance of a block was proposed by 3 participants, which is currently not supported. 5 participants would furthermore expect the entire scenario to be edited while the currently active blocks are disabled. However, 2 of them argued that this free extension strongly depends on the scenario and is only useful if a certain break is added, in which this editing can take place. Regarding the adjustment of existing blocks itself, 2 participants mentioned that dependencies have to be checked in order not to invalidate the model. Another participant argued that the options in the model view are already way too much. As a solution for both of these issues, the distinction between a standard and an expert view was proposed by 3 participants, e.g., by adding a lock at the bottom right of the model view that can be opened. After unlocking, further functions would be provided, such as the deletion of blocks, extensions of the scenario in advance of blocks, or even the free editing of the scenario. The function to change parameters of existing blocks, however, seems to be a crucial extension, and thus, should be integrated into the standard view. Nevertheless, even for this functionality, the validity has to be checked, as references might exist.

In the second question of the final part, the participants were asked whether they think that functional proposals provide a suitable extension at runtime. 6 participants agreed with this statement, 4 have rather agreed, one has rather disagreed, and 2 participants disagreed. Even though the average rating of 3.85 indicates that the participants rather agreed with the statement, the 3 participants that have rather or fully disagreed are worse investigating. One of these participants argues that making proposals at runtime is not useful, as the lecturer has already planned his/her lecture and has no time left to integrate further activities. In addition, the lecturer cannot consider these proposals in such a short time. Thus, displaying proposals would most likely result in disturbing hints that have to be closed. However, displaying those proposals after the lecture has finished would be quite interesting, as they could be integrated into future scenarios.

6.2 Lecture Experiments

In this section, the applicability of our approach will be validated. Therefore, four lecture experiments are presented, in which the prototype was used to support different teaching scenarios.

Note on the Lecture Experiments

As the goal of these experiments was not to validate the ability to model a scenario correctly (which was already intensively discussed in section 6.1), the scenarios were created cooperatively with the lecturers. The focus was on ensuring that the idea the lecturer had in mind was implemented properly. Special thanks to all lecturers that agreed to conduct these experiments.

In order to retrieve feedback from the students, two questions were added at the end of each scenario: First, the students were asked, whether the system could be used analogously to related systems. Second, open feedback could be provided, e.g., a description of a bug they observed. In addition, the log files were scanned for possible misbehavior of the system. After each lecture experiment, the lecturer was asked to fill out a survey (cf. Appendix F), in which the experiment, as well as the perceived experience, should be described. Additionally, an interview was conducted when certain statements remained unclear.

6.2.1 Lecture Experiment on Group Interactions (June 14, 2021)

The first experiment was conducted on June 14, 2021 in a live-stream lecture of the BA Dresden⁷, in which students of the 2nd semester (in the study courses Information Technology and Media Computer Science) are taught the basics of Linux. The lecture was held by Dr. Marius Feldmann and included both a regular part (i.e., a presentation given by the lecturer) and a practical part. The goal was, on the one hand, to improve the structuring of the lecture and, on the other hand, to offer a simple way to assess students' level of knowledge in the practical part of it. Therefore, both parts of the lecture were represented in stARS and several interactive activities were added, as visualized in figure 6.3.

The regular part of the lecture was supported by two rounds of questions: In the first round, students were able to recapitulate their knowledge on the previous lecture and

⁷ <https://www.ba-dresden.de/> – last successful access on October 8, 2021

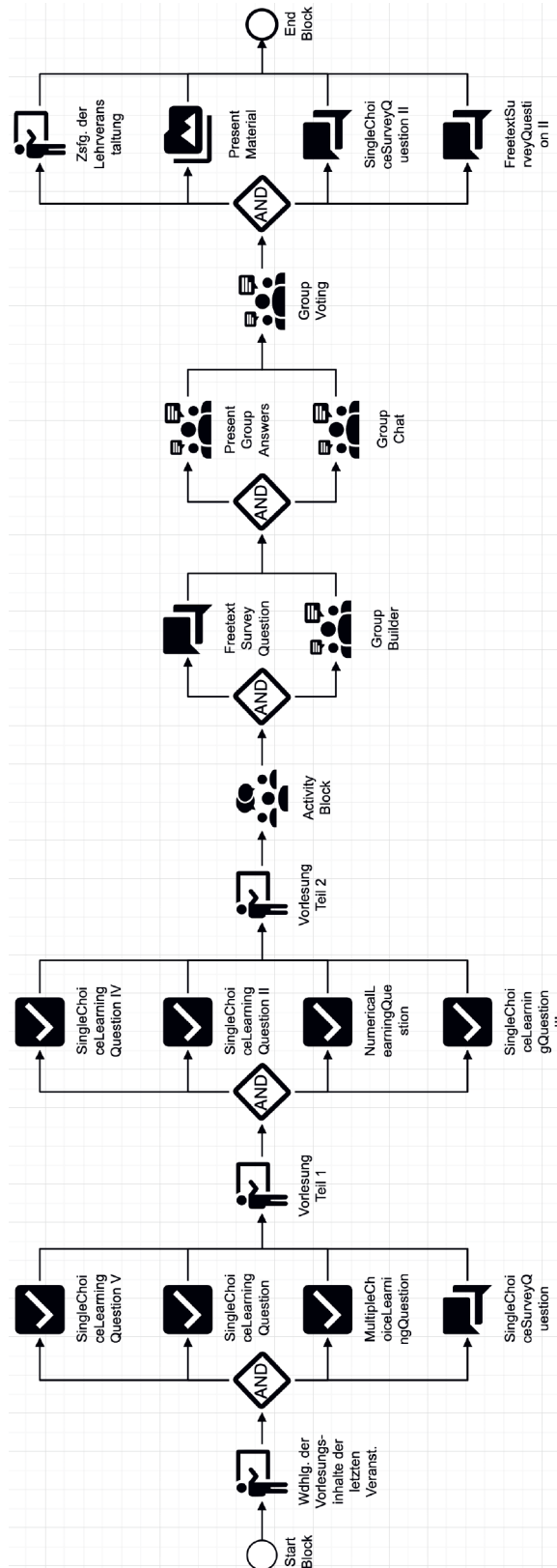


Figure 6.3: The scenario of the first lecture experiment conducted by Dr. Marius Feldmann, which includes two rounds of questions as well as an advanced group activity with textual discussions and group voting (several labels might contain German descriptions).

select a topic of interest to be discussed further (*SingleChoiceSurveyQuestion*). In the second round, which followed after introducing a new topic, students' gained knowledge was checked. While in the regular part, the system was used as a supporting tool, it took an essential role in the practical part, in which not only the general topic was described (*ActivityBlock*), but also a question had to be solved (*FreertextSurveyQuestion*). In order to make this part even more interactive, the students were grouped based on their previous knowledge (*GroupBuilder* with a *groupSize* set to 4 and the *buildSchema bestToWorst* referring to the prior learning questions) and could discuss their answers textually (*GroupChat*) while their group members' answers were displayed in parallel (*PresentGroupAnswers*). After completing this group discussion, they had the opportunity to select a common group answer (*GroupVoting*). This was targeted to reduce a large number of textual responses to a few that can be displayed and discussed by the lecturer. The scenario ended with a short summary of the lecture and two questions that were asked for evaluation purposes (as described previously).

The usage of the system was announced beforehand. In addition, an explanation was given at the beginning of the lecture, why the tool is used, and the structure of the scenario was presented to the students. Dr. Feldmann estimates that about 90% of the students have participated in the utilization of the system, which took around 20 to 30 minutes in total. While the question rounds were carried out without problems, technical problems were encountered during the practical part of the lecture:

1. As both the group discussion and the group voting were found on students' textual responses to the *FreertextSurveyQuestion*, it was essential that the majority of students responds. However, only 16 of 42 students (that acknowledged to join the group task) answered the question (another student answered the question but did not join). Even though an assignment would be possible such that each group is shown at least one answer, the *buildSchema bestToWorst* was defined to form groups of students with varying knowledge. This also created groups in which no student had given an answer, so only *Abstention* could be selected as the group answer⁸, which was clearly not the intention of this task. Possible solutions would be either to define the *FreertextSurveyQuestion* as a prerequisite to join the group task (i.e., only the 17 students who answered could have discussed and voted) or to extend the scenario at runtime by a *PresentResult* block such that groups, even if no group member has given an answer, have a starting point for the discussion. For the latter, it would be important to automatically select a

⁸ If *abstention* is selected as the group's answer, a group member is selected as a *moderator* that is allowed to answer again.

moderator that inputs the group answer. In contrast, the consideration of both given and not given answers is not useful in combination with the *buildSchema bestToWorst*. Instead, this availability of given answers could be realized as a *buildSchema* itself, which would implement the assignment that each group has at least one student that answered, if possible.

2. Moreover, as this case of missing answers to the referring question was not investigated in further detail before, a variety of students were displayed an error message, even if the submitted group answers were successfully recorded in the database (and 2 groups were assigned a moderator, as no common group answer could be found). This was later fixed after the experiment was completed.
3. Another error occurred during the group formation itself: An eleventh group with 2 students was created, which was not intended either. Instead, 10 groups should have been created, 2 of them having 5 group members. This was caused by the implementation itself that tested with 5 participants and a *groupSize* of 3. While in this case building two groups (having 3 and 2 group members) would be a suitable distribution, it does not make sense for a larger number of students, in which the remaining students can be evenly distributed among all groups. Consequently, the implementation was adjusted accordingly.

Despite these problems, Dr. Feldmann was satisfied with the “simple usage” of stARS and praised “the possibility of structuring the lecture and checking the level of knowledge of the students more easily.” Therefore, he invested approximately 1 to 2 hours in preparing the contents in stARS, although it has to be noted that he was rather not familiar with the approach itself or similar tools before. In future uses, Dr. Feldmann would like to add more interactive parts (e.g., learning questions), as he believes that “these are great for both ad-hoc assessment of whether students have understood the content and for repeating contents.” Another experiment conducted by Dr. Feldmann is described in subsection 6.2.4, in which the *GroupVoting* was used again.

The students also rated the approach for the most part quite positively, although the technical problems are getting obvious at this point. The question of whether the system could be used analogously to related systems without restrictions was answered as follows⁹: 5 students agreed with this statement, 18 partly agreed, 13 neither agreed

⁹ As answering these questions was optional, only 39 answers were recorded, while 2 of those were abstentions.

nor disagreed and 1 student partly disagreed. Using the scoring introduced in subsection 6.1.1, an average value of 2.73 (out of 4) can be retrieved, which indicates that students somehow agree but noted the difficulties during the *GroupVoting*.

This can be confirmed by evaluating the textual responses: While 15 out of 31 answers were abstentions, 7 out of the remaining 16 participants stated that they had problems with the *GroupVoting* – some of them made concrete suggestions about what they would expect from this function. Moreover, 1 student said that there exist more intuitive systems providing similar functions. However, several students also praised the benefits of the system: 5 participants stated that they liked the quiz functionality, while 2 explicitly noted the visualization of contents and solutions. In addition, 3 participants said that they like the idea of virtual group discussions.

In order to evaluate the results even further, the textual messages that were exchanged in the *GroupChat* were analyzed. All created 11 groups used the chat functionality, while the number of chat messages per group varies from 6 to 20. In total, 114 messages were sent. It is noticeable that the majority of group chats (9 out of 11) started with “hello” or “test” because, at this point of the implementation, students were not shown how many students they were in a group with. This is also partially reflected in the remaining messages, i.e., the confusion of students with whom they collaborate. As a result, the implementation was later adjusted to display both the number of group members as well as their pseudonyms. Moreover, several messages exist regarding the selection of a group answer, which was unlocked in the next part of this scenario (and not in parallel to the *GroupChat*). However, students were unsure when to expect this functionality. In a later scenario, this should be described, e.g., using a *PresentMaterial* block with instructions on the current stage. Regarding the group task itself, 10 of 11 groups discussed it (at a different level of degree). 3 groups actually discussed about finding a concrete answer. Nevertheless, the difficulty of the task is clearly recognizable. Due to the fact that only 16 answers were submitted in the *FreetextSurveyQuestion*, the majority of participants wrote that they had not yet found a solution or did not understand the topic. What is surprising, however, is the low number of spam. Although students could have entered as many or as long messages as they wanted and would have been protected by their anonymity, none of the participants took advantage of this. Similarly, no offenses or similar messages were posted.

6.2.2 Lecture Experiment on Peer Interactions (June 18 – June 28, 2021)

The second experiment was conducted between June 18, 2021 and June 28, 2021 in the seminar *Service and Cloud Computing* of the TU Dresden, which is addressed to students of computer science. The seminar was held virtually by Dr. Iris Braun and ran for a complete semester. After creating a paper about a scientific topic, a *Peer Feedback* task is included, in which students rate each other's papers. The experiment's goal was to represent this specific task in stARS, i.e., the entire process of submitting papers, assigning them to peers as well as submitting and downloading the reviews. Therefore, the scenario was modeled as visualized in figure 6.4 (each step was active for several days): In the first step, the students were allowed to upload their papers (*FileUploadSurveyQuestion (Upload)*). In parallel, a hint was given (*PresentMaterial*) that the students have to join the peer task (*PeerBuilder*) if it is not already done automatically after uploading the paper. After finishing this step, random assignments were created, and each student was shown another student's submission¹⁰ (*PresentPeerAnswers*) and he/she was allowed to upload the review of it (*FileUploadSurveyQuestion (Review)*). After the completion of this step, the students were presented their received review (*PresentPeerFeedback*) and were able to ask open questions to the provider of it (*PeerChat*). Moreover, the same two questions for evaluation purposes were posted in parallel (as formulated in the introduction of this section).

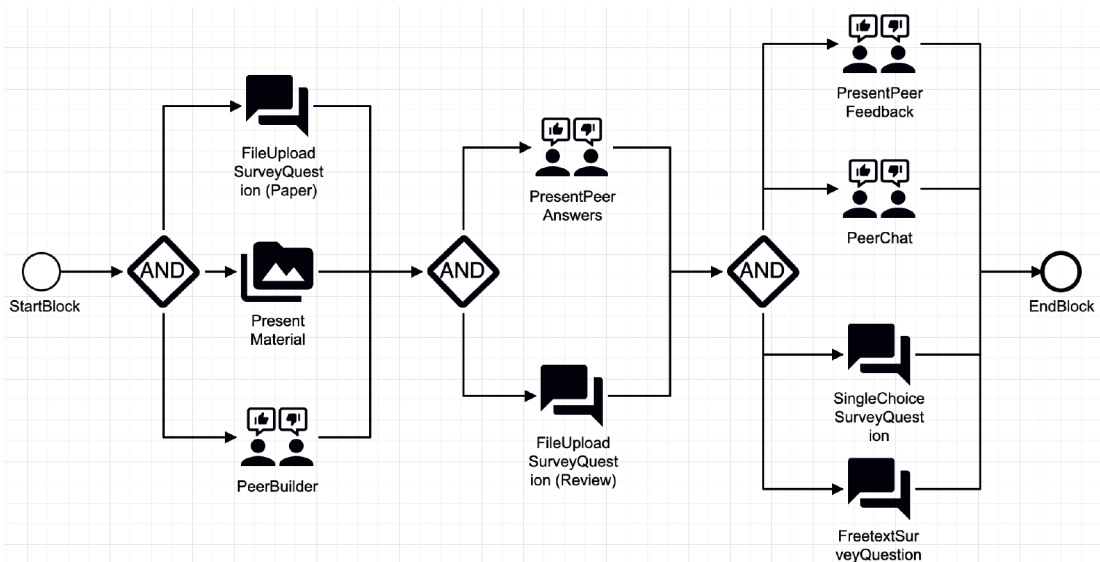


Figure 6.4: The scenario of the second lecture experiment conducted by Dr. Iris Braun, which includes a traditional *Peer Feedback* situation running over multiple days.

¹⁰ The *PeerBuilder* had set the parameter *numberOfAssignments* to 1 – thus, one submission had to be rated.

The usage of stARS was announced via OPAL (the Learning Management System used by the TU Dresden), and an instruction of the scenario was provided to the students, in which each step was described in further detail. This was important, as the usage itself was mandatory for all students¹¹. In total, 14 students uploaded their papers in the first step and acknowledged joining the peer task. However, several days after finishing this step (i.e., also after the assignments were created), one student had to leave the seminar because his/her submitted paper did not meet the requirements to pass. Although the student's supervisor was willing to write the review assigned to him/her, this had to be realized in stARS. Therefore two options were available: Either the student's password could be changed, allowing the supervisor to log in, or a new assignment could be added manually so that the supervisor would be able to create the review. As the student had to leave the seminar, we decided to change the account's password in order to also avoid potential spam being submitted. In future scenarios, this could also be handled as follows: With the possibility to manage the assignments manually, the assignment of the student could be removed, and instead, an assignment to the supervisor could be defined, keeping the student's account untouched and still avoiding spam. Finally, after all reviews were uploaded, the last step was unlocked, in which the students were presented their feedback, could discuss with each other and optionally give feedback to stARS.

Besides the difficulty with one student dropping out, there were no technical errors that negatively influenced the usage of the system. Thus, Dr. Braun was satisfied with the scenario and praised both the automatic assignment of peers as well as the ability to chat with each other after completing the *Peer Feedback* task. However, in future uses, she would like to add more textual descriptions which give instructions on how to use the currently active step (i.e., *PresentMaterial* blocks). Moreover, the access of multiple lecturers (i.e., supervisors) to the same scenario would be preferred. Therefore, the following changes were made: As not every supervisor should be able to manage the workflow, the global user role *Lecturer* was replaced by two roles *Controller* (that can manage the workflow) and *Evaluator* (that can see the results), which allows customizing privileges more fine-grained.

Out of 13 students that completed the task, only 4 gave feedback on stARS. 2 of them fully agreed that the system could be used analogously to related systems. The remaining 2 students partly agreed. However, due to the low number of responses, this has to be seen as an impression of a few students instead of representing the opinion of the majority. Considering the textual feedback, 2 students rated the anonymity feature

¹¹ Therefore, the students were also asked to use their university identification as a "pseudonym."

positively, while also 2 students argued that the blocks should be described better, e.g., after uploading a file, the “submitted” state should be visualized.

Moreover, by investigating the log files, the *PeerChat* can be evaluated: In total, 8 messages were exchanged. While 2 students chatted with each other about their given feedback, another student tried it, but his/her *Peer Feedback* partner did not reply. In any case, we still believe that this function is a useful extension to let students share their opinions. In future scenarios, the ability to chat could be possible as soon as the assignments are created to also allow discussing earlier (i.e., the *PeerChat* has to be modeled in parallel to the remaining steps).

In order to retrieve further feedback on the usage of stARS, the students were asked for their opinions during the presentation of their papers. In general, they stated that they could use the system as described. However, some students expected further functions to be supported, such as different questions to rate the paper. Furthermore, it was suggested to compare the scenario with an implementation in OPAL, which allows realizing similar tasks. Nevertheless, OPAL does not support automatic assignments of peers, and students might feel insecure when revealing their identity to the other student¹². For future semesters, it is planned to represent the entire seminar (i.e., the full semester) in stARS, e.g., by providing material or giving hints for upcoming deadlines. Furthermore, questions could be created (e.g., asking for the course of study) that can be used to assign the peers more meaningful (in this experiment, the *buildSchema random* was selected because there were no previous questions).

6.2.3 Lecture Experiment on Peer Interactions (June 23, 2021)

The third experiment was conducted as part of a block seminar on educational-psychological interventions, which was held virtually at the TU Berlin by Dr. Felix Kapp. The seminar runs over 4 days with 4 to 6 teaching units each. In order to improve the interactivity, Dr. Kapp used several tools: For example, AMCS¹³ was used on all 4 dates to assess students’ gained knowledge. stARS was used on the last day (June 23, 2021) and was targeted to execute a task (“create an intervention of your own”) involving students receiving and giving peer feedback, which goes beyond the traditional *Audience Response* functionality (e.g., as provided by AMCS). Therefore, a scenario that takes 2 teaching units (i.e., 90 minutes) was modeled as shown in figure 6.5. After an

¹² Although the students had to use their university identification during login, it was not revealed to the assigned student. Instead, stARS uses a randomly generated pseudonym, as shown in figure 5.17

¹³ <https://amcs.website/> – last successful access on October 8, 2021

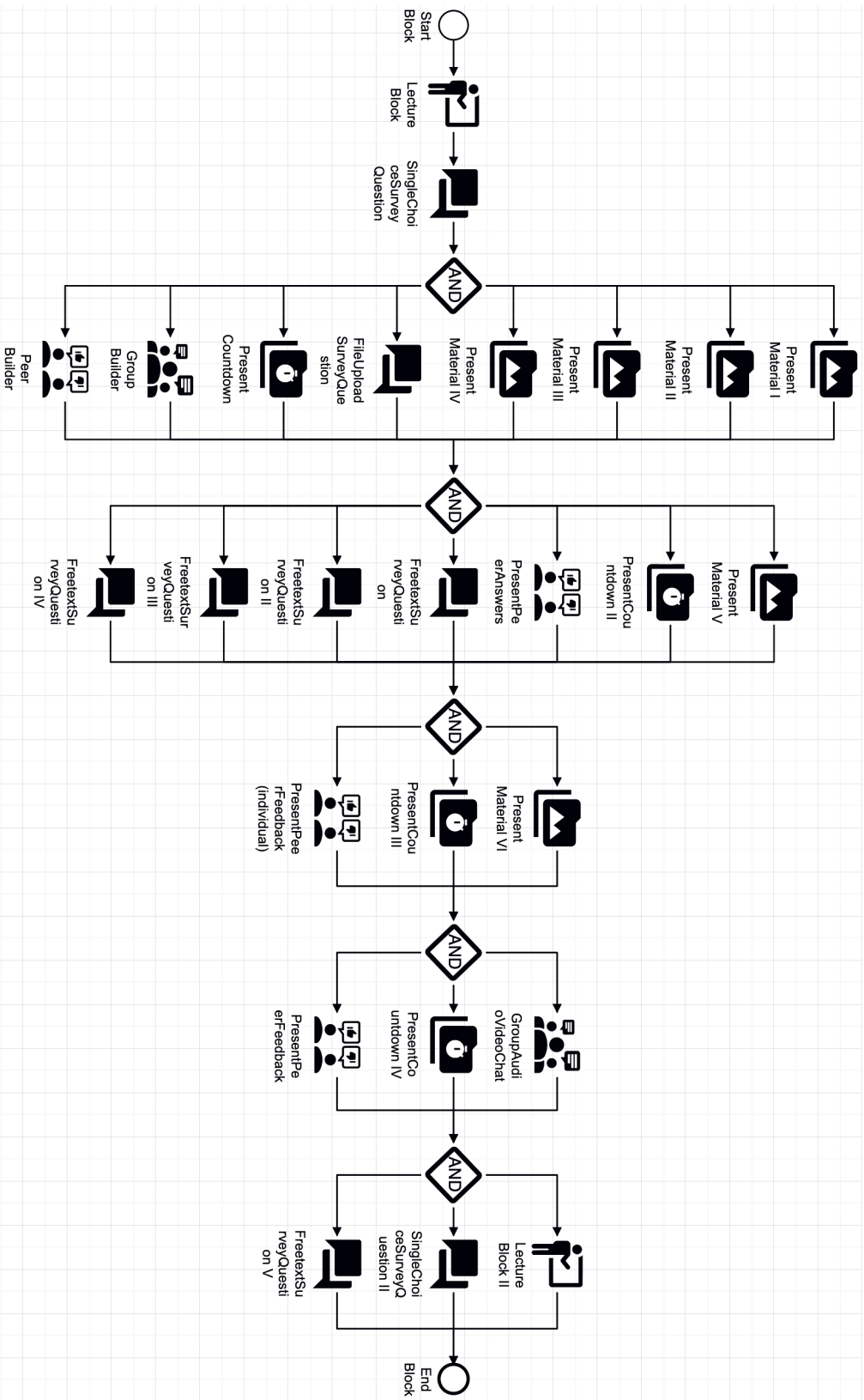


Figure 6.5: The scenario of the third lecture experiment conducted by Dr. Felix Kapp, in which an advanced *Peer Feedback* situation is represented.

introduction to stARS as well as the general task (*LectureBlock*), the students had to select a topic for their intervention (*SingleChoiceSurveyQuestion* with 4 options). Depending on the selected option, they were shown different material in the next step (*PresentMaterial* (I – IV)), which was realized using the *filter* parameter, and had the task to create an intervention on their own. Therefore, a time frame of 20 minutes was defined, which was presented to the students using a *PresentCountdown* block. In addition, a *FileUploadSurveyQuestion* was added to allow uploading the solution. Furthermore, both a *GroupBuilder* and a *PeerBuilder* were modeled in parallel, which was caused by the intention of the lecturer. The goal of the assignment was that two students who choose the same topic for their intervention would give each other feedback and later have the opportunity to discuss this feedback in an audio-video chat. Since this cannot be realized by a simple *PeerBuilder* (as only textual discussions are allowed and the assignment of two students with each other is not necessarily ensured), a *GroupBuilder* was used that creates groups of two students using the *buildSchema sameAnswer* and a *PeerBuilder* that refers to this *GroupBuilder*.

After the completion of this step, the students were displayed the assigned submission (*PresentPeerAnswers*) and they were asked to rate it using four open-ended questions (*FreetextSurveyQuestion* (II, III, IV)). Again, they were displayed a countdown (*PresentCountdown* II with 15 minutes) as well as a description of the current step (*PresentMaterial*). In the next step, the students received their feedback (*PresentPeerFeedback* (individual)) and had a fixed time frame to review it (*PresentCountdown* III with 5 minutes) before they were able to discuss this feedback another 5 minutes (*PresentCountdown* IV) with their peer using an audio-video chat (*GroupAudioVideoChat*). In the final step, the scenario was concluded by Dr. Kapp (*LectureBlock* II) and the students were asked to rate stARS using the two questions presented earlier.

The usage of the approach was announced on the third day, however, without specifying the exact system. Additionally, an explanation at the beginning of the teaching unit was provided, and the login was carried out collectively in order to resolve questions as soon as they arise. According to Dr. Kapp, 11 of 15 students that were enrolled in the seminar were present on that day, while 10 of them participated in the scenario. The schedule closely followed the defined countdowns – the students were only once given an extra 3 minutes to complete a task. Regardless of one student who had problems with his/her computer (updates were installed), which partly caused other students to wait (so the formation of peers would work properly), the system could be used without problems: 5 groups with two students that gave each other feedback were created.

Dr. Kapp praised the usage of stARS as an approach “to create a very organized *Peer Feedback* situation online, which is not yet supported by existing systems.” In a later interview, he additionally pointed out the importance of such scenarios, which were commonly used in traditional settings but could not be used that effectively in virtual settings anymore. Furthermore, he emphasized the ability of stARS to make decisions for problems that he did not consider in detail before, e.g., the assignment of students, if groups of two cannot be formed for all topics. While, according to Dr. Kapp, the scenario (which took around 3 hours to prepare) can be reused 1:1, its representation is not trivial. This is caused by the combination of *Group-* and *PeerBuilder* (which was required to create the targeted peer assignment) as well as the technical perspective that is created in stARS, e.g., that the *GroupBuilder* is modeled in parallel with the submission of the task. Moreover, during the usage, Dr. Kapp would have liked to access students’ submissions adaptively during usage (even after the step was completed) and display them during a later debriefing phase. For future uses, which he says are quite likely, he would plan more time, allowing students to work on their interventions and assignments.

The students themselves rated the approach quite differently. While 5 students fully agreed that the system could be used analogously to related approaches, 2 partly agreed, 2 did neither agree, nor disagree and 1 student did partly disagree. This results in an average value of 4.1, which is just above the threshold for a partial agreement. The textual responses (which were submitted by 9 out of 10 students) are even more positively: Even though one student at a time said the time given was not enough, he/she was alone in the audio-video chat and uploading the file was just possible after reloading the page, the positive feedback of the students dominates. For example, 3 students said that everything worked fine, 3 liked the structuring and 2 stated that the approach is intuitive to use. Furthermore, one participant praised the novelty of the approach and referred to the fact that he/she did not see a similar approach before. Another student even stated that the usage “felt like real partner work.”

After considering the log files, we can verify that all groups, and thus, the peer assignments, were built properly: While 5 students selected topic 1, 2 students selected topic 2, one student topic 3 and 2 students topic 4. Thus, two groups were created with two students from topic 1, one group each for students from topics 2 and 4, and only one mixed group with a student from topic 1 and topic 3. As a random Jitsi link is generated per group for the *GroupAudioVideoChat*, there are no log messages to be evaluated for this part of the scenario. Even though one student reported that his/her partner did not join this chat, another student praised the ability to discuss in this way.

6.2.4 Lecture Experiment on Group Interactions (July 22, 2021)

The fourth experiment was conducted as a part of an exam consultation that was virtually held by Dr. Marius Feldmann for the lecture “Internet and Web Application” at the TU Dresden. The goal of using stARS was on the one side to structure the meeting more efficiently and on the other side to recapitulate the topic of *Kademlia*, which is a subject relevant to the exam. In addition to stARS, a student thesis was presented, which should also be represented in the scenario. The modeled scenario can be seen in figure 6.6. In the first step, the welcome and introduction was represented by a traditional *LectureBlock* with a specific name and description. Next, the student presented his/her master thesis (also represented as a *LectureBlock*) and the participants were asked to submit their email addresses if they were willing to help evaluating the approach (*FreetextSurveyQuestion*). Then, the exam consultation started by summarizing exam information before students could ask their own questions via audio or chat in the associated audio-video conference (in this case, Zoom was used by Dr. Feldmann). After all questions were clarified, the interactive part started: First, the students were asked to check their gained knowledge on *Kademlia* using two learning questions (*SingleChoiceLearningQuestion* and *MultipleChoiceLearningQuestion*). After the results were discussed, the next step started, in which the students should describe the functioning of *Kademlia* in no more than 500 characters. Therefore, a reference to the slides as well as to a paper was presented, which should help students to repeat the topic. In parallel, the *GroupBuilder* was added, which used the *buildSchema bestToWorst* referring to the previous learning questions in order to form groups of 5 students each with different prior knowledge. After the completion of this step, the students got the ability to see their group members’ given answers and discuss those textually. In addition, a countdown of 5 minutes was presented in this step (*PresentCountdown*) to inform students about the duration of this group activity. In the next step, each group member could vote for a specific answer in order to find a common group answer. In parallel, a description of this functionality was presented (*PresentMaterial II*). Finally, after the common group answers were found and discussed by Dr. Feldmann, the scenario was completed and the students were asked to provide feedback on stARS. However, some of the students misunderstood those questions and rated the entire lecture, which was quickly clarified by Dr. Feldmann.

The usage of the system was not announced beforehand. Instead, it was introduced at the beginning of the meeting by presenting the workflow (cf. figure 6.6). While around 60 participants were present at the start, a variety of them left after the questions about the exam were clarified. Thus, only 32 answers were recorded for the *SingleChoiceLearn-*

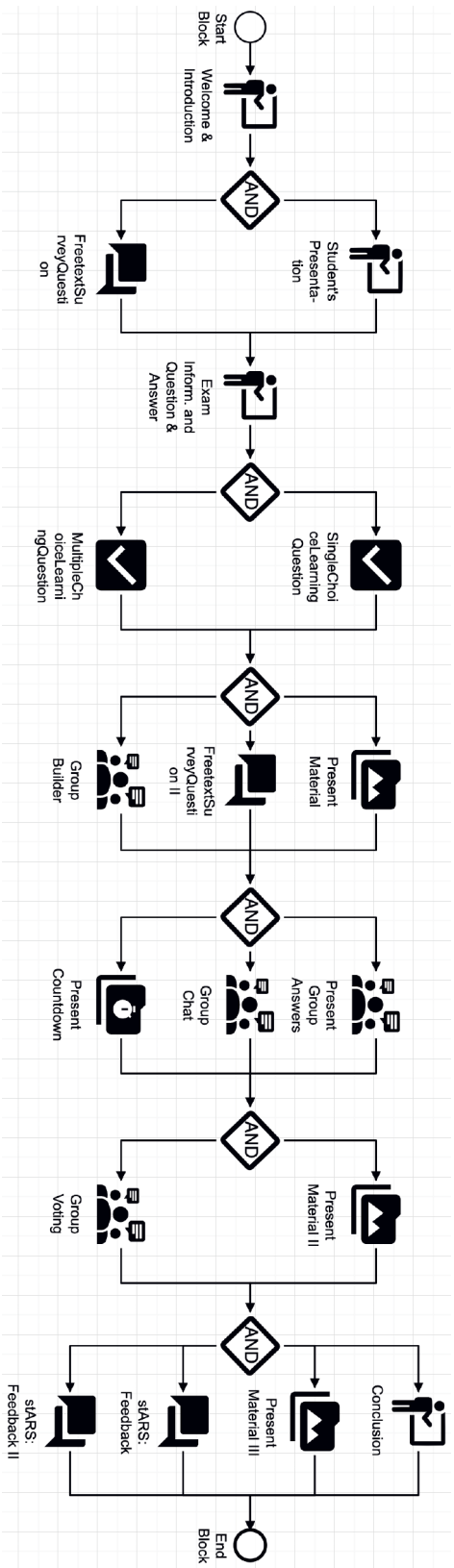


Figure 6.6: The scenario of the fourth lecture experiment conducted by Dr. Marius Feldmann, which includes both a question round and an advanced group activity with a group voting.

ingQuestion and 33 for the *MultipleChoiceLearningQuestion*. This further decreased in the next step, in which only 13 students answered the *FreetextSurveyQuestion (II)*, while 16 joined the group activity (*GroupBuilder*). As groups of 5 students should be built, three groups were created, with one group having 6 participants. However, those 16 participants actively used the system. Moreover, another 7 students passively joined the task, i.e., they did not participate in the group activity but rated the system at the end of the scenario. As there were no errors detected during the execution that prevented students from participating, we believe that several students were actively preparing for their exams and did not recognize the importance of the task. However, Dr. Feldmann was satisfied with the usage of stARS and praised the ability to structure the meeting more efficiently as well as the opportunity to discuss an important topic (i.e., *Kademlia*).

This positive feedback is also reflected in the students' answers. 10 students agreed that they could use stARS similar to other systems without limitations. Moreover, 9 students partly agreed, 1 student each did rather not agree and disagree, and 2 students did abstain from answering. Thus, an average rating of 4.24 can be recorded, which is just above the "partly agree" threshold. In any case, according to the textual answers, there are at least two students that rated the overall lecture series instead of the system. However, the remaining textual answers are quite positive: 1 student stated that he/she likes the model, which is easy to understand. Another student liked the "anonymity" and made suggestions for a useful extension for spell checking. Furthermore, 2 students emphasized the necessity of a second screen, while 1 of them rated it as "great" if this precondition is met. Finally, another student praised that using stARS is "more interactive rather than just watching videos."

Evaluating the log files results in similar positive results: While the groups were created properly (i.e., according to the *buildSchema bestToWorst*, each group contains a similar "knowledge distribution" among its members), students also used the *GroupChat* quite efficiently. A total of 22 messages were exchanged, each group's chat having between 4 and 9 messages. In addition, only one *GroupChat* started with "Hi," indicating that the students recognized the number of students inside their group. However, one group was unsure how the selection of a group answer works (that followed afterward). In the future, this should be better highlighted in the function itself or described by a *PresentMaterial* block. Furthermore, the *GroupVoting* also worked without problems. Each group was able to select a common group answer, while in two groups, all group members selected the same answer, and in one group 3 out of 5 students selected it. This selection allowed reducing the number of 13 freetext answers to 3 that could be discussed with the students.

Note on the Measurement of the Learning Success

The lecture experiments presented in this section were targeted to prove the general functioning of the system. However, future evaluations have to be conducted to reason about the ability of stARS to improve students' learning success. While traditional functions such as Audience Response were empirically proven to foster learning, this should also be proven for the functions introduced by this thesis. Therefore, students could either be asked to state whether these functions help to improve their learning, or the learning success could be measured by comparing two similar groups (one using stARS and one traditional teaching). However, the latter strongly depends on the model of the scenario (cf. section 7.2).

6.3 RT1: Modeling Adaptation for Customized Teaching Scenarios

This section will elaborate on RT1 that “modeling adaptation allows lecturers to create customized teaching scenarios that support their individual teaching strategies.” Therefore, each subsection will discuss one of the sub research theses that were formulated in section 3.5.

RT1.1: Lecturers want to use teaching scenarios in which students are actively involved more often and are willing to support these with technical tools.

In the second user study (cf. subsection 6.1.2), the participants (i.e., 20 lecturers) were presented a list of 14 teaching scenarios (including both traditional and interactive ones) and were asked on the one side to select those that they want to use more often and on the other side those that they want to support by technical tools. While all traditional teaching scenarios (namely *Frontal Teaching*, *PowerPoint Presentations* and *Student Presentations*) were known but rarely selected to be used more often (i.e., 1, 3 and 4 lecturers selected those), all remaining (more interactive) teaching scenarios were selected by at least 3 participants (cf. table 6.2). Moreover, especially *Peer Instruction* (12), *Interactive Learning Questions* (10), *Think-Pair-Share* (9) and *Problem-based Learning* (9) were selected more frequently, even though some of them were unknown to several participants (as stated by 3, 1, 8 and 1 participant(s)). Although both scenarios exist that the majority of lecturers do not want to use more often, and the most frequently selected scenarios were not chosen by every single lecturer, we can still verify the first part of RT1.1, stating that lecturers are willing to support teaching scenarios, in which students

are actively involved, more often. At this point, it is worth noting that stARS does not attempt to influence lecturers in using specific teaching scenarios more frequently. Instead, these scenarios are provided for inspiration, allowing lecturers to represent their individual teaching scenarios in stARS more intuitively. In the second part of RT1.1, it has to be checked whether the participants are willing to support teaching scenarios that involve students more actively by technical tools. While all traditional teaching scenarios were selected by at least half of the participants (indicating that those lack interactivity), all other scenarios were also selected by at least 4 participants. Moreover, *Interactive Learning Questions* (16), *Peer Instruction* (14), *Think-Pair-Share* (10) and *Problem-based Learning* were selected quite frequently, especially when considering that not all of them were known to every single lecturer (cf. table 6.2). Also *Jigsaw Classroom*, *Learning Stations* and *World Cafe* were selected by almost half of the lecturers who knew them. These results allow verifying the second part of RT1.1 as well. Thus, RT1.1 can be evaluated as fulfilled.

RT1.2: By defining elements and parameters for interactive activities through a (meta-)model, a variety of scenarios can be expressed, but the model is still easy to understand and extendable.

In section 4.2, the concept of the (meta-)model to express different teaching scenarios was created and could be evaluated as easy to understand (cf. subsection 4.2.6). In order to represent a variety of teaching scenarios, the (meta-)model includes not only traditional activities (such as *Audience Response*) but was also extended by collaborative activities (i.e., *group-* and *peer interactions*). The expressiveness of the (meta-)model was shown in section 5.7, in which models of seven well-known teaching scenarios were created, while one of those is a rather complex one in order to show that the approach is not limited to simple scenarios. Moreover, it is possible to model customized teaching scenarios, as it was done in the lecture experiments described in section 6.2. Especially in the third lecture experiment (cf. subsection 6.2.3), the potential of the approach was shown by expressing a complex assignment of peers that was not specifically considered before. Therefore, two blocks, namely the *PeerBuilder* and the *GroupBuilder*, were combined systematically. However, it has to be noted that the comprehensibility decreases with an increasing complexity of the model. This does not only become obvious for the third lecture experiment (cf. subsection 6.2.3) but also within the second user study conducted early-2021 (cf. subsection 6.1.2), in which five models with increasing complexity were evaluated. While the participants were able to understand simple scenarios with ease, they had problems with complex scenarios. However,

several participants were still able to understand even those models. Thus, we can confirm *RT1.2* that a (meta-)model can be used to express a variety of teaching scenarios, which is still understandable and extendable.

RT1.3: The integration of standard formats to express interactive activities will ensure the quality of the approach.

In subsubsection 4.2.4, the usage of standard formats in order to express interactive activities was discussed. Integrating standard formats was motivated by the fact that for common activities, such as learning questions, different format specifications exist, e.g., the *IMS Question and Test Interoperability specification (QTI)*¹⁴. However, these are rarely supported by current systems, as can be seen in [Kub+19]: For example, the import of questions in the QTI format was (at the time the paper was written) only supported by 3 out of 50 systems. One reason for this could be the following: Although it is possible to describe different types of questions with different behaviors, there exist several use cases, which cannot be expressed (without using custom format extensions), e.g., the definition of feedback for different choices or the provision of common misconceptions. Moreover, current formats are mostly limited to describing questions (e.g., QTI) or general learning activities (e.g., SCORM or xAPI), which would make it challenging to specify all the activities that are planned by the (meta-)model. Furthermore, each standard distinguishes from the format that is targeted by the (meta-)model and would not be consistent with it (for example, the inheritance would not necessarily be supported). Finally, there even exist further arguments to not rely on these standard formats, as described by [Gar+04; Pio11]. As a result, no standard format has been used during the development of our approach, which makes it impossible to verify *RT1.3*. However, we strongly believe that using standard formats would not have resulted in an improved quality of the approach and instead rather would have decreased its comprehensibility and consistency.

RT1.4: A graphical user interface enables lecturers to express their individual teaching strategies by allowing them to model customized teaching scenarios.

Although the (meta-)model offers a simple way to describe both elements (e.g., *functional blocks*) and transitions, creating customized teaching scenarios is still a challenging task. Thus, a graphical editor was developed (cf. section 4.3) that is targeted to ease this

¹⁴ At the time of writing, the current version of QTI was 3.0

task. The created prototype was evaluated as having a “good usability” (i.e., an average SUS score of 73,9% was submitted in the first user study by 20 lecturers), meaning that lecturers were already able to create teaching scenarios¹⁵. Furthermore, five models with different complexity were proven to be understood by the majority of lecturers (cf. subsection 6.1.2), even though the comprehensibility of models decreases with an increased complexity. This also holds for special use cases, in which blocks have to be combined systematically in order to realize a specific setting (cf. subsection 6.2.3). Nevertheless, *RT1.4* can be confirmed, i.e., that a graphical editor (i.e., user interface) allows lecturers to model customized teaching scenarios in order to support their individual teaching strategies.

RT1.5: Supporting lecturers in getting started and during modeling eases the understanding of the modeling process.

Even though both the graphical editor and the created models were proven to be understood by the majority of lecturers, some of them had problems getting started or during the modeling itself. Consequently, several supporting concepts were investigated, such as an initial overlay, automatic connection of elements, or functions to ease the modeling of complex scenarios (cf. subsection 4.3.3), as described in [KRT20]. Within different evaluations that were conducted as a part of students’ work (i.e., a practical course of Lidia Roszko, a practical course of Niclas Zellerhoff, as well as the master thesis of Sinthujan Thanabalasingam [Tha21]), almost every component has proven to speed up the modeling process. The component that was rated as only partially helpful (i.e., the *decision tree*) was later replaced by an advanced proposal-based function that suggests useful next (groups of) elements based on the currently existing model (cf. [Pei21]). In addition, the *reminder messages* were disabled for the moment, as those require further investigations to be conducted. However, with the variety of components developed, we are able to prove *RT1.5*: Supporting lecturers in getting started and during modeling eases the understanding of the modeling process.

¹⁵ The prototype of the graphical editor was later extended by different supporting concepts that should further increase the usability (cf. *RT1.4*)

RT1.6: The approach can be used by both lecturers and students without limitations to similar approaches.

In the final sub research thesis of *RT1*, it had to be verified whether the approach can be used by both lecturers and students without limitations to similar approaches. Therefore, four lecture experiments were conducted, in which the system was used to support different scenarios (cf. section 6.2). While the first experiment suffered from minor bugs in the implementation and therefore, the students' opinions differed, the students in the other experiments mostly agreed that they were able to use the approach without limitations to similar approaches (i.e., the average value indicates *partly agreement*). Some participants also gave textual feedback that ranges from suggestions for useful adjustments to praise for different functions: This is specifically interesting for those functions that differ from traditional activities, namely, the group- and peer interactions, which were considered helpful by different students. Furthermore, the lecturers' opinions were assessed using a questionnaire as well as optional interviews. It is remarkable that all lecturers rated the approach quite positively, even if minor technical difficulties arose in the first experiment. However, each lecturer was satisfied with the range of functions that is supported by stARS and did at least partly agree that he/she will likely use it again. Besides minor technical problems that were solved after completing the first lecture experiment (which was later checked in the fourth experiment), both lecturers and students were able to use the approach without major limitations in comparison to similar approaches. Thus, *RT1.6* can be validated as confirmed.

6.4 RT2: Runtime Adaptation for Adjusting Running Teaching Scenarios

This section will elaborate on *RT2* that "runtime adaptation allows adjusting teaching scenarios on the fly in order to respond to real-time results." Therefore, each subsection will discuss one of the sub research theses that were formulated in section 3.5.

RT2.1: Lecturers want to change their teaching scenarios during the execution.

In order to evaluate *RT2.1*, whether lecturers are willing to change their teaching scenarios during the execution, a corresponding question was added in the second user study (cf. subsection 6.1.2). 9 out of 19 lecturers fully agreed that they consider the

subsequent changing of didactic scenarios in stARS as important, 8 partly agreed and 2 did neither agree nor disagree. Using the scoring introduced in subsection 6.1.1, an average value of 4.37 can be retrieved, indicating that the lecturers consider changing the scenarios as at least partly important (even though the value tends towards full agreement). Similar results were retrieved in subsection 6.1.3, in which 8 participants agreed that such a function is useful and 5 rather agreed. Furthermore, the importance of the function was highlighted in the use case of *Learners-as-Designers* (cf. subsection 5.7.7) as well as the lecture experiments (cf. section 6.2), in which different use cases for suitable extensions (at runtime) were motivated: For example, if groups were built, in which no group member has given an answer, a *PresentResult* block could be added to display the answers of all students – this way, even groups without given answers would have a starting point for discussion. These findings allow evaluating RT2.1 to be confirmed.

RT2.2: Changing teaching scenarios on the fly allows implementing teaching scenarios that rely on student-generated data.

In subsection 5.7.7, the implementation of a complex student-centered learning approach called *Learners-as-Designers* was discussed. Therefore, both using a static model with a variety of *DecisionForks* as well as building the scenario at runtime (by extending it through templates) were investigated to be valid options for implementation. However, it has to be noted that extending the scenario was found to be a crucial requirement, as there might be situations that cannot be foreseen if the entire scenario depends on students' input. In any case, RT2.2 can be confirmed, i.e., the extension of models allows implementing even student-centered approaches.

RT2.3: Limiting the changing of teaching scenarios to additions will avoid errors during execution and still be expressive enough to make changes.

Even though it is true that extending a valid model by another valid model results in a valid model again, which will avoid errors during the execution, at the end of subsection 6.1.3, it could not be confirmed with certainty that extensions alone are expressive enough to make all required changes. Instead, the participants did neither agree nor disagree on this statement. However, several suggestions were made to improve this function. Future work has to integrate these improvements step by step and check the statement again. However, we strongly believe that lecturers benefit from

a “reduced” functionality, especially when adjusting an ongoing lecture. Instead, if a certain break is present in the lecture, a free extension might be suitable as well.

RT2.4: Functional proposals provide a suitable extension in order to respond to real-time results even if the lecturer is not aware of the necessity.

Although most participants of subsection 6.1.3 did at least rather agree that functional proposals are a suitable extension to respond to real-time results, 3 lecturers did rather or fully disagree. The reason for this is that lecturers coming to class already have a fixed schedule. Often, there is no time left to integrate further activities, and even if it is, considering such (previously unknown) proposals at runtime is not possible. Instead, lecturers might have their own templates that are integrated at runtime. However, even though *RT2.4* cannot be confirmed, functional proposals could be an interesting extension after the lecture has finished. This way would allow finding suitable templates that can be integrated into future lectures and thus, even improve the teaching of the lecturer itself.

6.5 RT3: Role Concept as a Promising Extension for Integrating Adaptation

This section will elaborate on *RT3* that “the concept of roles provides a promising extension to integrate the means of adaptation in *digital learning environments*.” Therefore, each subsection will discuss one of the sub research theses that were formulated in section 3.5.

RT3.1: The concept of roles provides a useful extension to the (meta-)model in order to model a variety of different teaching scenarios.

In subsection 4.5.2, the extension of the concept of roles within the (meta-)model was discussed. Even though it might be a possible option, we had concerns about the comprehensibility of the approach for inexperienced users as well as about its expressiveness, as the modeling of arbitrary teaching scenarios seems to be impossible. Consequently, it was decided not to directly integrate the role concept into the (meta-)model, which makes verifying *RT3.1* not possible.

RT3.2: The concept of roles provides a useful extension at runtime in order to allow for changes (i.e., role transfers) within single functions.

Subsection 5.5.2 described a variety of different roles that were implemented in stARS. For example, in the compartment of the *GroupVoting*, *GroupMembers* of the same group try to find a common group answer. However, if no common group answer can be found (i.e., a tie exists between two answers or the most popular answer is *abstention*), one of the *GroupMembers* is automatically assigned the role of a *GroupVotingModerator*. When holding this role, the user can select or input the group's answer. Further use cases exist in which similar functions are provided. This allows confirming RT3.2.

RT3.3: The concept of roles improves the extendability of the approach, as runtime data that is not specified by the (meta-)model can be added.

As already elaborated in subsection 5.5.3, using the concept of roles as an extension at runtime can increase the extendability of the approach. Thus, RT3.3 can be verified positively.

6.6 Summary

In this chapter, the evaluation of our approach was presented as follows:

In section 6.1, three different user studies were summarized that were specifically conducted to validate the research theses. While the first user study checked lecturers' ability to use the graphical editor, the second study elaborated on the comprehensibility of more complex scenarios, and the third one investigated an advanced concept for runtime adaptation.

Afterward, in section 6.2, four different lecture experiments carried out to verify the correct functioning of the system were presented. All of these scenarios included novel functions presented by stARS, i.e., group- and peer interactions.

Finally, in the last three sections (section 6.3, section 6.4 and section 6.5), the sub research theses of RT1, RT2 and RT3 were validated by either referring to previous chapters or using the results of the user studies and lecture experiments presented before. In the following chapter, these investigations will be used to answer the research theses and thus, the main research question of this thesis.

7 Conclusions

This thesis intended to answer the overall research question that is formulated as follows: “How can different levels of adaptation support the lecturer in properly using digital learning environments?” Therefore, three research theses were defined that were investigated throughout the thesis:

RT1) Modeling adaptation allows lecturers to create customized teaching scenarios that support their individual teaching strategies.

RT2) Runtime adaptation allows adjusting teaching scenarios on the fly in order to respond to real-time results.

RT3) The concept of roles provides a promising extension to integrate the means of adaptation in digital learning environments.

After presenting both the fundamentals (cf. chapter 2) and the current state of *digital learning environments* (cf. section 3.1, section 3.2 and section 3.3, as published in [Kub+19]), related work was investigated in section 3.4. The results were not only used to visualize the research gap tackled by this thesis in section 3.5 but also to formulate sub theses that had to be verified as well as a variety of functional and non-functional requirements for developing a solution. In chapter 4, the concept of an adaptable collaborative learning environment (cf. [KSS19; Kub19a; Kub19b]) was created before it was implemented in a prototype called scenario-tailored Audience Response System (stARS) (cf. chapter 5, as published in [Kub+20b]) that integrates the concept of roles (cf. [KPB20]) and allows supporting a variety of teaching scenarios (cf. [Kub+20a; KRT20]). This prototype was evaluated in several user studies (cf. section 6.1, as published in [Kub+21]) and lecture experiments (cf. section 6.2), which made it possible to verify the sub theses of RT1 (cf. section 6.3), RT2 (cf. section 6.4) and RT3 (cf. section 6.5).

7.1 Findings of this Thesis

In this section, the results of section 6.3, section 6.4 and section 6.5 will be used to verify the research theses and, thus, answer the overall research question.

In table 7.1, both the sub research theses of *RT1* (i.e., modeling adaptation allows lecturers to create customized teaching scenarios that support their individual teaching strategies) and their validity are summarized.

Research Thesis	Confirmation
RT1.1: Lecturers want to use teaching scenarios in which students are actively involved more often and are willing to support these with technical tools.	✓
RT1.2: By defining elements and parameters for interactive activities through a (meta-)model, a variety of scenarios can be expressed, but the model is still easy to understand and extendable.	✓
RT1.3: The integration of standard formats to express interactive activities will ensure the quality of the approach.	○
RT1.4: A graphical user interface enables lecturers to express their individual teaching strategies by allowing them to model customized teaching scenarios.	✓
RT1.5: Supporting lecturers in getting started and during modeling eases the understanding of the modeling process.	✓
RT1.6: The approach can be used by both lecturers and students without limitations to similar approaches.	✓

Table 7.1: The verification of the sub research theses of *RT1*.

Even though *RT1.3* could not be confirmed, it does not influence the validity of *RT1*, as it just argues that using standard formats ensures the quality of the approach. However, it could be shown that the developed approach is not only able to be used by lecturers (*RT1.4*, *RT1.5* and *RT1.6*) but also expressive enough (*RT1.2*), without using a standard format. Instead, we believe that integrating standard formats in our approach would not have improved and rather decreased its comprehensibility. In any case, *RT1* can be confirmed, i.e., modeling adaptation allows creating customized teaching scenarios to support individual teaching strategies.

Next, *RT2* (i.e., runtime adaptation allows adjusting teaching scenarios on the fly in order to respond to real-time results) has to be investigated. Thus, the results of the sub theses that are summarized in table 7.2 will be discussed.

Research Thesis	Confirmation
RT2.1: Lecturers want to change their teaching scenarios during the execution.	✓
RT2.2: Changing teaching scenarios on the fly allows implementing teaching scenarios that rely on student-generated data.	✓
RT2.3: Limiting the changing of teaching scenarios to additions will avoid errors during execution and still be expressive enough to make changes.	○
RT2.4: Functional proposals provide a suitable extension in order to respond to real-time results even if the lecturer is not aware of the necessity.	×

Table 7.2: The verification of the sub research theses of *RT2*.

Although both *RT2.1* and *RT2.2* could be confirmed, no clear statement can be made on *RT2.3*. While some lecturers agreed that extending scenarios is sufficient in order to make changes to a running scenario, others did not. However, it seems like adding another function to adjust the parameters of already existing blocks is a proper extension. Furthermore, *RT2.4* had to be declined, even if it was rated positively by most lecturers. Reasons for this are both the limited time available for extensions as well as lecturers' uncertainty of the extension's functionality. Instead, making proposals after completing the lecture might be an interesting function to add. Even if not all sub theses could be confirmed, *RT2* can be confirmed for situations in which own extensions are made. Instead, proposing suitable extensions seems not to be a proper functionality to add at runtime.

Finally, *RT3* (i.e., the concept of roles provides a promising extension to integrate the means of adaptation in digital learning environments) will be discussed. Therefore, the results of the sub theses are summarized in table 7.3.

Research Thesis	Confirmation
RT3.1: The concept of roles provides a useful extension to the (meta-)model in order to model a variety of different teaching scenarios.	○
RT3.2: The concept of roles provides a useful extension at runtime in order to allow for changes (i.e., role transfers) within single functions.	✓
RT3.3: The concept of roles improves the extendability of the approach, as runtime data that is not specified by the (meta-)model can be added.	✓

Table 7.3: The verification of the sub research theses of *RT3*.

As the concept of roles was not integrated fundamentally in the (meta-)model due to concerns about its comprehensibility and expressiveness (cf. subsection 4.5.2), *RT3.1* cannot be validated. Instead, it is an integral part of the runtime, which allows confirming *RT3.2*. Furthermore, by considering the distinction between data specified by the (meta-)model and runtime data that can be made by using roles, also *RT3.3* could be validated positively. Thus, *RT3* can be at least confirmed for the realization of the runtime and might also be a possible (but not optimal solution) for the modeling part, as discussed in subsection 4.5.2.

These findings allow answering the overall research question of this thesis (i.e., “how can different levels of adaptation support the lecturer in properly using digital learning environments?”) as follows:

Modeling adaptation allows creating customized teaching scenarios that are able to support lecturers’ individual teaching strategies. However, the approach has to be easy to understand or to learn, as the modeling itself introduces challenges on its own. In any case, if an approach is designed accordingly, lecturers strongly benefit from the capabilities the modeling process introduces.

Runtime adaptation presents a useful extension to adjust teaching scenarios on the fly, e.g., in order to respond to real-time results. Therefore, a tradeoff has to be found between the expressiveness of the extension and both the validity of the model and the ability to use this function in parallel to the running lecture.

Therefore, we propose a reduced function to be used, e.g., the extension of scenarios by other scenarios or the adjustment of parameters of already existing blocks. Instead, the free extension of scenarios might be interesting for scenarios in which certain breaks are added. In any case, validity has to be ensured. Furthermore, functional proposals should not be made at runtime, as those would rather annoy lecturers instead of helping them.

In order to integrate such adaptations in *digital learning environments*, the concept of roles offers an interesting paradigm to be used. However, we do not believe that having the modeling entirely relying on the role concept would result in the same comprehensibility and expressiveness as it was achieved for our approach. Instead, at runtime, the concept of roles could be proven as a useful extension.

7.2 Outlook

Although a vast amount of potential future research topics exist, this section will present the four most important ones that provide logical steps to be conducted next.

Validate the Improved Learning Success of Using stARS

Even though four lecture experiments were conducted in the course of this thesis (cf. section 6.2), and the students could have been asked whether they think that using stARS improves their learning, such questions were intentionally omitted. Instead, it was evaluated whether lecturers' individual teaching strategies could be supported and the system behaves as expected.

However, validating the learning success of using stARS is a future topic of research, of which we think that it requires more than asking simple questions regarding students' expectations. Although those results could indicate an improved learning success, actual measurements are required to validate it. Therefore, the success of two similar groups of students (one group that uses stARS during the entire semester and one that uses traditional teaching) could be compared. Nevertheless, there are different challenges that might arise: First, the similarity of both groups has to be proven in order to compare them. Secondly, as stARS includes different functions (such as *Audience Response*) that were already proven empirically, the results could heavily rely on the modeled scenario(s) in stARS. Even though it is of interest to investigate the novel functions of stARS, those work best when being combined with empirically proven

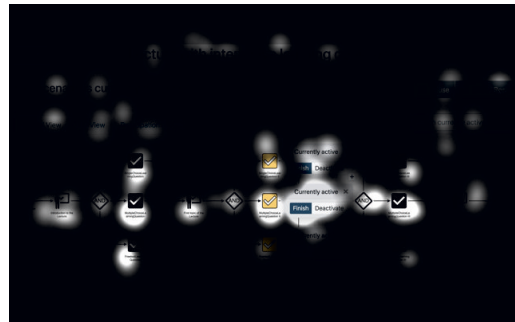
functions (*Audience Response*), e.g., if the group formation should rely on students' prior knowledge. An idea would be either the comparison with another group that uses traditional teaching combined with *Audience Response*, or to fully omit functions that were already empirically proven, e.g., by randomly forming groups of students.

Further Investigations of the Runtime Adaptation

During the investigation of *RT2.3*, it remained open which extensions have to be made to the functionality for runtime adaptation in order for it to be sufficient for all lecturers. We believe that slightly extending this functionality, e.g., by allowing to adjust parameters of existing blocks, could make lecturers agree to this function. Therefore, it should be adjusted accordingly and investigated by advanced methodologies in order to get further insights into the thoughts of the lecturers when using it. In figure 7.1, the means of using eye-tracking in order to trace the view path of lecturers when solving a task are motivated.



(a) The heat map of a lecturer searching a function.



(b) The focus map of a lecturer searching a function.

Figure 7.1: The usage of eye-tracking to trace the view paths of lecturers when searching a functionality.

Extract Best Practice Templates

Although in [Hon21], a prototype for extracting *used practice* templates was proposed, a lot of templates that are created do not have any didactically meaningful use. Instead, incomplete scenarios or scenarios that are specified for individual use cases are exported. Thus, a challenge will be the extraction of “best practice”-templates¹ that can later be used to provide suggestions for useful next elements (cf. [Pei21]). This could be realized

¹ However, whether a template is a best practice or not has again be evaluated separately.

by only taking into account scenarios that were successfully executed by lecturers, or by applying ratings to the generated templates that are used as an input for the following generation processes.

Support Asynchronous, Student-Paced Scenarios

As the general approach intends that lecturers control the scenario and thus unlock activities to all their students (i.e., lecturer-paced), it is currently rather limited to the support of synchronous teaching scenarios. However, asynchronous teaching scenarios are not least because of the CoViD-19 pandemic a proven strategy to be used (especially in digital learning) (cf. [Dan20]) whose support should be examined as well. Since students often struggle to self-organize in asynchronous scenarios, supporting them by providing an interactive workflow that they can follow and proceed by themselves would be a promising topic to investigate.

Bibliography

- [AA18] Gökçe Akçayır and Murat Akçayır. "The flipped classroom: A review of its advantages and challenges." In: *Computers & Education* 126 (2018), pp. 334–345. doi: 10.1016/j.compedu.2018.07.021.
- [ABO10] Hans Aagard, Kyle Bowen, and Larisa Olesova. "Hotseat: Opening the Backchannel in Large Lectures." In: *Educause Quarterly* 33.3 (2010), p. 2.
- [AG13] Antognoni Albuquerque and Giancarlo Guizzardi. "An ontological foundation for conceptual modeling datatypes based on semantic reference spaces." In: *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)*. IEEE. 2013, pp. 1–12. doi: 10.1109/RCIS.2013.6577693.
- [Álv+17] Claudio Álvarez, Nelson Baloian, Gustavo Zurita, and Fabio Guarini. "Promoting Active Learning in Large Classrooms: Going Beyond the Clicker." In: *CYTED-RITOS International Workshop on Groupware*. Springer. 2017, pp. 95–103. doi: 10.1007/978-3-319-63874-4_8.
- [AR20] Ammar Y Alqahtani and Albraa A Rajkhan. "E-Learning Critical Success Factors during the COVID-19 Pandemic: A Comprehensive Analysis of E-Learning Managerial Perspectives." In: *Education Sciences* 10.9 (2020), p. 216. doi: 10.3390/educsci10090216.
- [Arn+13] Ketil Arnesen, Guri Sivertsen Korpas, Jon Eirik Hennissen, and John Birger Stav. "Experiences with Use of Various Pedagogical Methods Utilizing a Student Response System – Motivation and Learning Outcome." In: *Electronic Journal of e-Learning* 11.3 (2013), pp. 169–181.
- [Aro78] Elliot Aronson. *The Jigsaw Classroom*. Sage, 1978.
- [BD77] Charles W. Bachman and Manilal Daya. "The role concept in data models." In: *Proceedings of the third international conference on Very large data bases*. Vol. 3. 1977, pp. 464–476.
- [Ber15] Stefan Bernhardt. *Vergleich unterschiedlicher Abstimmungssysteme*. 2015. URL: http://ilias.uni-giessen.de/ilias/goto.php?target=cat_25037. Last access on 10/8/2021.
- [BGR20] Raphaël Benitte, Sacha Greif, and Michael Rambeau. *State of JS 2020: Front-end Frameworks*. 2020. URL: <https://2020.stateofjs.com/en-US/technologies/front-end-frameworks/>. Last access on 10/8/2021.
- [BKM09] Aaron Bangor, Philip Kortum, and James Miller. "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale." In: *Journal of usability studies* 4.3 (2009), pp. 114–123.
- [BP17] François Bry and Alexander Yong-Su Pohl. "Large class teaching with Backstage." In: *Journal of Applied Research in Higher Education* (2017). doi: 10.1108/JARHE-06-2015-0042.
- [Bra+18] Iris Braun, Tenshi Hara, Felix Kapp, Lucas Braeschke, and Alexander Schill. "Technology-Enhanced Self-Regulated Learning: Assessment Support Through an Evaluation Centre." In: *Proceedings of the 42nd International Computer Software and Applications Conference (COMPSAC)* 1 (2018), pp. 1032–1037. doi: 10.1109/COMPSAC.2018.00180.
- [Bra+19] Lucas Braeschke, Iris Braun, Felix Kapp, and Tenshi Hara. "Integrate Confidence Ratings in Audience Response Systems in Order to Help Students to Self-regulate Their Learning Process." In: *Proceedings of the 11th International Conference on Computer Supported Education (CSEDU)* (2019). doi: 10.5220/0007731404090415.
- [Bra16] Cynthia Brame. *Active learning*. 2016. URL: <https://cft.vanderbilt.edu/active-learning/>. Last access on 10/8/2021.

- [Bru09] Derek Bruff. *Teaching with Classroom Response Systems: Creating Active Learning Environments*. John Wiley & Sons, 2009. ISBN: 978-0-470-28893-1.
- [Bru19] Derek Bruff. *Intentional Tech: Principles to Guide the Use of Educational Technology in College Teaching*. West Virginia University Press, 2019. ISBN: 978-1949199154.
- [BS12] Jonathan Bergmann and Aaron Sams. *Flip Your Classroom: Reach Every Student in Every Class Every Day*. International Society for Technology in Education, 2012. ISBN: 978-1564843159.
- [CA14] Ricardo Caceffo and Rodolfo Azevedo. *LSQuiz: A Collaborative Classroom Response System to Support Active Learning through Ubiquitous Computing*. ERIC, 2014. ISBN: 978-989-8533-23-4.
- [CBP21] Tim Coughlan, Simon Ball, and Leigh-Anne Perryman. *Synchronous and asynchronous modes of teaching*. 2021. URL: <https://www.open.edu/openlearn/ocw/mod/oucontent/view.php?id=77528§ion=1>. Last access on 10/8/2021.
- [CCC16] Yu-Ta Chien, Yueh-Hsia Chang, and Chun-Yen Chang. “Do we click in the right way? A meta-analytic review of clicker-integrated instruction.” In: *Educational Research Review* 17 (2016), pp. 1–18. doi: 10.1016/j.edurev.2015.10.003.
- [CDV15] Clemens H. Cap, Christian Delfs, and Jonas Vetterick. “Tweedback goes Smart Watch – Why Classroom Response Systems need Smart Watch User Interfaces.” In: *Workshop Proceedings of the 11th International Conference on Intelligent Environments*. 2015, pp. 273–280. doi: 10.3233/978-1-61499-530-2-273.
- [Cer+19] Cinzia Cervato *et al.* “Web-based student response systems and peer instruction: a review and case study.” In: *Journal of Academic Development and Education* 11 (2019). doi: 10.21252/41wc-kt98.
- [CGC16] Gary Cheng, Yuanyuan Guan, and Juliana Chau. “An empirical study towards understanding user acceptance of bring your own device (BYOD) in higher education.” In: *Australasian Journal of Educational Technology* 32.4 (2016). doi: 10.14742/ajet.2792.
- [CI14] Wilmax Marreiro Cruz and Seiji Isotani. “Group Formation Algorithms in Collaborative Learning Contexts: A Systematic Mapping of the Literature.” In: *CYTED-RITOS International Workshop on Groupware*. 2014. doi: 10.1007/978-3-319-10166-8_18.
- [CM01] Catherine H Crouch and Eric Mazur. “Peer instruction: Ten years of experience and results.” In: *American journal of physics* 69.9 (2001), pp. 970–977. doi: 10.1119/1.1374249.
- [Cra+20] Joseph Crawford, Kerryon Butler-Henderson, Jürgen Rudolph, Bashar Malkawi, Matt Glowatz, Rob Burton, Paulo Magni, and Sophia Lam. “COVID-19: 20 countries’ higher education intra-period digital pedagogy responses.” In: *Journal of Applied Learning & Teaching* 3.1 (2020), pp. 1–20. doi: 10.37074/jalt.2020.3.1.7.
- [Dam17] Gregor Damnik. “Preconditions and Critical Factors for a Successful Acquisition of Content Knowledge in the Learners-as-Designers Approach.” PhD thesis. Technische Universität Dresden, 2017.
- [Dan20] John Daniel. “Education and the COVID-19 pandemic.” In: *Prospects* 49.1 (2020), pp. 91–96. doi: 10.1007/s11125-020-09464-3.
- [Dem15] Muhammet Demirbilek. “Social media and peer feedback: What do students really think about using Wiki and Facebook as platforms for peer feedback?” In: *Active Learning in Higher Education* 16.3 (2015), pp. 211–224. doi: 10.1177/1469787415589530.
- [DG18] Fedor Duzhin and Anders Gustafsson. “Machine Learning-Based App for Self-Evaluation of Teacher-Specific Instructional Style and Tools.” In: *Education Sciences* 8.1 (2018), p. 7. doi: 10.3390/educsci8010007.
- [Dil99] Pierre Dillenbourg. *What do you mean by collaborative learning?* 1999.
- [Doc21] Sirius Documentation. *Specifying Viewports*. 2021. URL: https://www.eclipse.org/sirius/doc/specifier/general/Specifying_Viewpoints.html. Last access on 10/8/2021.

- [Ebn+14] Martin Ebner, Christian Haintz, Karin Pichler, and Sandra Schön. *Technologiegestützte Echtzeit-Interaktion in Massenvorlesungen im Hörsaal. Entwicklung und Erprobung eines digitalen Backchannels während der Vorlesung*. 2014. ISBN: 978-3-8309-3142-3.
- [Ebn+20] Martin Ebner, Sandra Schön, Clarissa Braun, Markus Ebner, Ypatios Grigoriadis, Maria Haas, Philipp Leitner, and Behnam Taraghi. "COVID-19 epidemic as E-learning boost? Chronological development and effects at an Austrian university against the background of the concept of "E-Learning Readiness"." In: *Future Internet* 12.6 (2020), p. 94. doi: 10.3390/fi12060094.
- [Ed-20] Hamza Ed-douibi. *10+ JavaScript libraries to draw your own diagrams (2020 edition)*. 2020. URL: <https://modeling-languages.com/javascript-drawing-libraries-diagrams/>. Last access on 10/8/2021.
- [El 17] Abir El Shaban. "The use of Socrative in ESL classrooms: Towards active learning." In: *Teaching English with Technology* 17.4 (2017), pp. 64–77.
- [Erd+15] Sebastian Erdweg, Tijs Van Der Storm, Markus Völter, Laurence Tratt, Remi Bosman, William R Cook, Albert Gerritsen, Angelo Hulshout, Steven Kelly, Alex Loh, *et al.* "Evaluating and comparing language workbenches: Existing results and benchmarks for the future." In: *Computer Languages, Systems & Structures* 44 (2015), pp. 24–47. doi: 10.1016/j.cl.2015.08.007.
- [Fli17] Jonas Flint. "Feedback-Proxys Zur Digitalisierung Von Classroom Response Systemen." PhD thesis. Universität Rostock, 2017, pp. 22–23.
- [Fre+14] Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. "Active learning increases student performance in science, engineering, and mathematics." In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8410–8415. doi: 10.1073/pnas.1319030111.
- [FS20] Hong Fan and Xiaofei Song. "The advantages of combining mobile technology and audience response systems." In: *Journal of Accounting Education* 50 (2020), p. 100657. doi: 10.1016/j.jaccedu.2020.100657.
- [FWB13] Linus Feiten, Katrin Weber, and Bernd Becker. "SMILE: Smartphones in der Lehre—ein Rück- und Überblick." In: *INFORMATIK 2013—Informatik angepasst an Mensch, Organisation und Umwelt* (2013).
- [Gar+04] Rocio Garcia-Robles, Josep Blat, Sergio Sayago, Dai Griffiths, Francis Casado, and Juanjo Martinez. "Limitations of some eLearning standards for supporting learning." In: *Proceedings of the International Conference of Interactive Computer-Aided Learning, ICL*. 2004.
- [Gob19] Gobalo. *Definition of a Domain-Specific Modelling Language*. 2019. URL: <https://miuc.org/definition-of-a-domain-specific-modelling-language/amp/>. Last access on 10/8/2021.
- [Gon18] Arturo González. "Turning a traditional teaching setting into a feedback-rich environment." In: *International Journal of Educational Technology in Higher Education* 15.1 (2018), p. 32. doi: 10.1186/s41239-018-0114-1.
- [Gra16] David Granada. *Comparing tools to build graphical modeling editors*. 2016. URL: <https://modeling-languages.com/comparing-tools-build-graphical-modeling-editors/>. Last access on 10/8/2021.
- [HAB16] Nathaniel J Hunsu, Olusola Adesope, and Dan James Bayly. "A meta-analysis of the effects of audience response systems (clicker-based technologies) on cognition and affect." In: *Computers & Education* 94 (2016), pp. 102–119. doi: 10.1016/j.compedu.2015.11.013.
- [Hal98] Terry Halpin. "Object-Role Modeling (ORM/NIAM)." In: *Handbook on Architectures of Information Systems* (1998), pp. 81–103. doi: 10.1007/3-540-26661-5_4.
- [Har16] Tenshi Hara. "Analyses on tech-enhanced and anonymous Peer Discussion as well as anonymous Control Facilities for tech-enhanced Learning." PhD thesis. Technische Universität Dresden, 2016.

- [Har96] William S. Harwood. "The one-minute paper." In: *Journal of chemical education* 73.3 (1996), p. 229.
- [Her05] Stephan Herrmann. "Programming with Roles in ObjectTeams/Java." In: *Proceedings of the AAAI Fall Symposium*. 2005.
- [HK14] Rolf Hennicker and Annabelle Klarl. "Foundations for Ensemble Modeling – The Helena Approach." In: *Specification, Algebra, and Software*. Springer, 2014, pp. 359–381. doi: 10.1007/978-3-642-54624-2_18.
- [HK19] Shelley Hymel and Jennifer Katz. "Designing Classrooms for Diversity: Fostering Social Inclusion." In: *Educational Psychologist* 54.4 (2019), pp. 331–339. doi: 10.1080/00461520.2019.1652098.
- [HMB19] Niels Heller, Sebastian Mader, and François Bry. "More than the sum of its parts: designing learning formats from core components." In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019, pp. 2473–2476. doi: 10.1145/3297280.3297638.
- [Hoc20] Allensbach Hochschule. *Fernstudieren in Deutschland: Zahlen und Fakten*. 2020. URL: <https://www.allensbach-hochschule.de/fernstudieren-in-deutschland-zahlen-und-fakten/>. Last access on 10/8/2021.
- [Hon21] Chang Hong. *The generation of templates based on existing scenarios in the digital learning environment stARS*. 2021. URL: https://www.rn.inf.tu-dresden.de/amcs/arbeiten/MA_2021_Hong_-_The_generation_of_templates_based_on_existing_scenarios_in_the_digital_learning_environment_stARS.pdf. Last access on 10/8/2021.
- [Hoo14] André van Hoorn. *Model-Driven Online Capacity Management for Component-Based Software Systems*. Department of Computer Science, Kiel University, 2014. ISBN: 978-3-7357-5118-8.
- [HPE14] Christian Haintz, Karin Pichler, and Martin Ebner. "Developing a Web-Based Question-Driven Audience Response System Supporting BYOD." In: *Journal of Universal Computer Science* 20.1 (2014), pp. 39–56.
- [Hra08] Stefan Hrastinski. "Asynchronous and synchronous e-learning." In: *Educause quarterly* 31.4 (2008), pp. 51–55.
- [Hun17] Hsiu-Ting Hung. "Clickers in the flipped classroom: Bring your own device (BYOD) to promote student learning." In: *Interactive Learning Environments* 25.8 (2017), pp. 983–995. doi: 10.1080/10494820.2016.1240090.
- [Ish18] Ishaniagarwal. *What is Lazy Loading?* 2018. URL: <https://www.geeksforgeeks.org/what-is-lazy-loading/>. Last access on 10/8/2021.
- [ITW13] ITWissen. *OCL (object constraint language)*. 2013. URL: <https://www.itwissen.info/OCL-object-constraint-language.html>. Last access on 10/8/2021.
- [JB19] Tomislav Jagušć and Ivica Botički. "Mobile learning system for enabling collaborative and adaptive pedagogies with modular digital learning contents." In: *Journal of Computers in Education* 6.3 (2019), pp. 335–362. doi: 10.1007/s40692-019-00139-3.
- [Jir+15] Peerumporn Jiranantanagorn, Haifeng Shen, Robert Goodwin, and Kung-Keat Teoh. "Clas-Sense: A Mobile Digital Backchannel System for Monitoring Class Morale." In: *International Journal of Learning and Teaching* 1.2 (2015), pp. 161–167. doi: 10.18178/ijlt.1.2.161-167.
- [JM13] Alan Januszewski and Michael Molenda. *Educational Technology: A Definition with Commentary*. Routledge, 2013. ISBN: 978-0805858617.
- [Jon96] David H. Jonassen. "Learning with technology: Using computers as cognitive tools." In: *Handbook of research for educational communications and technology* (1996).
- [Kad13] Mahmoud Kaddoura. "Think Pair Share: A Teaching Learning Strategy to Enhance Students' Critical Thinking." In: *Educational Research Quarterly* 36.4 (2013), pp. 3–24.
- [Kap+14] Felix Kapp, Iris Braun, Hermann Körndle, and Alexander Schill. "Metacognitive Support in University Lectures Provided via Mobile Devices." In: *Proceedings of the 6th International Conference on Computer Supported Education (CSEDU)* (2014). doi: 10.5220/0004936901940199.

- [Kap14] Felix Kapp. "Effects of Learning Questions on Self-regulated Learning." PhD thesis. Technische Universität Dresden, 2014.
- [KK11] Felix Kapp and Hermann Körndle. *Was lerne ich aus einer Lernaufgabe? a) gar nichts, b) Faktenwissen, c) etwas über meine Lernstrategien, d) Antwort b und c sind richtig*. 2011. doi: 10.25656/01:11659.
- [KL09] Robin H. Kay and Ann LeSage. "Examining the benefits and challenges of using audience response systems: A review of the literature." In: *Computers & Education* 53.3 (2009), pp. 819–827. doi: 10.1016/j.compedu.2009.05.001.
- [Kle08] Anneke Kleppe. *Software language engineering: creating domain-specific languages using meta-models*. Pearson Education, 2008.
- [KNP04] Hermann Körndle, Susanne Narciss, and Antje Proske. *Konstruktion interaktiver Lernaufgaben für die universitäre Lehre*. 2004. doi: 10.25656/01:11265.
- [Kok20] Lau Tiam Kok. *Hands-on Nuxt.js Web Development: Build universal and static-generated Vue.js applications using Nuxt.js*. Packt Publishing Ltd, 2020.
- [KPB20] Tommy Kubica, Robert Peine, and Iris Braun. "Role-Based Group Formations and Interactions to Foster Collaboration in Different Classrooms." In: *Proceedings of the 32nd Conference on Software Engineering Education and Training (CSEET)*. IEEE. 2020, pp. 277–278. doi: 10.1109/CSEET49119.2020.9206230.
- [KR08] Kathy Kotiadis and Stewart Robinson. "Conceptual modelling: knowledge acquisition and model abstraction." In: *2008 Winter Simulation Conference*. IEEE. 2008, pp. 951–958. doi: 10.1109/WSC.2008.4736161.
- [KRT20] Tommy Kubica, Lidia Roszko, and Sinthujan Thanabalasingam. "Towards the Creation of Customized Teaching Scenarios to Support Classroom Interaction." In: *Proceedings of the 32nd Conference on Software Engineering Education and Training (CSEET)*. IEEE. 2020, pp. 275–276. doi: 10.1109/CSEET49119.2020.9206190.
- [KSS19] Tommy Kubica, Ilja Shmelkin, and Alexander Schill. "Towards a Development Methodology for adaptable collaborative Audience Response Systems." In: *Proceedings of the 18th International Conference on Information Technology Based Higher Education and Training (ITHET)*. IEEE. 2019. doi: 10.1109/ITHET46829.2019.8937354.
- [Kub+17] Tommy Kubica, Tenshi Hara, Iris Braun, Felix Kapp, and Alexander Schill. "Guided selection of IT-based education tools." In: *Proceedings of the 47th Frontiers in Education Conference (FIE)*. 2017. doi: 10.1109/FIE.2017.8190530.
- [Kub+19] Tommy Kubica, Tenshi Hara, Iris Braun, Felix Kapp, and Alexander Schill. "Choosing the appropriate Audience Response System in different Use Cases." In: *Proceedings of the 10th International Conference on Education, Training and Informatics (ICETI)*. 2019.
- [Kub+20a] Tommy Kubica, Tenshi Hara, Iris Braun, and Alexander Schill. "An Approach to Support Interactive Activities in Live Stream Lectures." In: *Proceedings of the 15th European Conference on Technology Enhanced Learning (EC-TEL)*. Springer. 2020, pp. 432–436. doi: 10.1007/978-3-030-57717-9_40.
- [Kub+20b] Tommy Kubica, Ilja Shmelkin, Robert Peine, Lidia Roszko, and Alexander Schill. "stARS: Proposing an Adaptable Collaborative Learning Environment to support Communication in the Classroom." In: *Proceedings of the 12th International Conference on Computer Supported Education (CSEDU)*. 2020. doi: 10.5220/0009489103900397.
- [Kub+21] Tommy Kubica, Gregor Damnig, Iris Braun, Robert Peine, and Alexander Schill. "Lecturers' Perceptions of Supporting Digital Teaching Scenarios by an Adaptable Learning Environment." In: *Proceedings of the 13th annual International Conference on Education and New Learning Technologies (EDULEARN)*. IATED. 2021. doi: 10.21125/edulearn.2021.2546.
- [Kub19a] Tommy Kubica. "Adaptable Collaborative Learning Environments." In: *Proceedings of the 14th EC-TEL Doctoral Consortium (DCECTEL)*. 2019.

- [Kub19b] Tommy Kubica. "Adaptierbare kollaborative Lernumgebungen zur gezielten Unterstützung universitärer Präsenzlehre." In: *Beiträge der Doktorandenkolloquiums zur DELFI* (2019).
- [Küh+14] Thomas Kühn, Max Leuthäuser, Sebastian Götz, Christoph Seidl, and Uwe Aßmann. "A Metamodel Family for Role-Based Modeling and Programming Languages." In: *International Conference on Software Language Engineering*. Springer. 2014, pp. 141–160. doi: 10.1007/978-3-319-11245-9_8.
- [Küh+15] Thomas Kühn, Stephan Böhme, Sebastian Götz, and Uwe Aßmann. "A combined formal model for relational context-dependent roles." In: *Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering*. 2015, pp. 113–124. doi: 10.1145/2814251.2814255.
- [Küh17] Thomas Kühn. "A family of role-based languages." PhD thesis. Technische Universität Dresden, 2017.
- [Kun+12] Dennis Kundisch, Michael Sievers, Andrea Zoyke, Philipp Herrmann, Michael Whittaker, Marc Beutner, Gregor Fels, and Johannes Magenheimer. "Designing a web-based application to support peer instruction for very large groups." In: (2012).
- [Kun+13] Dennis Kundisch, Johannes Magenheimer, Marc Beutner, Philipp Herrmann, Wolfgang Reinhardt, and Andrea Zoyke. "Classroom Response Systems." In: *Informatik-Spektrum* 36.4 (2013), pp. 389–393. doi: 10.1007/s00287-013-0713-0.
- [LC06] Ngar-Fun Liu and David Carless. "Peer feedback: the learning element of peer assessment." In: *Teaching in Higher Education* 11.3 (2006), pp. 279–290. doi: 10.1080/13562510600680582.
- [Len+15] Peter H. Lenz, Jennifer W. McCallister, Andrew M. Luks, Tao T. Le, and Henry E. Fessler. "Practical Strategies for Effective Lectures." In: *Annals of the American Thoracic Society* 12.4 (2015), pp. 561–566. doi: 10.1513/AnnalsATS.201501-024AR.
- [Leu17] Max Leuthäuser. "A Pure Embedding of Roles." PhD thesis. Technische Universität Dresden, 2017.
- [Lew+16] Justin D. Lewin, Erin L. Vinson, MacKenzie R. Stetzer, and Michelle K. Smith. "A Campus-Wide Investigation of Clicker Implementation: The Status of Peer Discussion in STEM Classes." In: *CBE—Life Sciences Education* 15.1 (2016), ar6. doi: 10.1187/cbe.15-10-0224.
- [Lew82] Clayton Lewis. *Using the "Thinking-aloud" Method in Cognitive Interface Design*. IBM TJ Watson Research Center Yorktown Heights, NY, 1982.
- [Lin+14] Debra L. Linton, Wiline M. Pangle, Kevin H. Wyatt, Karli N. Powell, and Rachel E. Sherwood. "Identifying Key Features of Effective Active Learning: The Effects of Writing and Peer Discussion." In: *CBE—Life Sciences Education* 13.3 (2014), pp. 469–477. doi: 10.1187/cbe.13-12-0242.
- [Liu+17] Cui Liu, Sufen Chen, Chi Chi, Kuei-Pin Chien, Yuzhen Liu, and Te-Lien Chou. "The Effects of Clickers With Different Teaching Strategies." In: *Journal of Educational Computing Research* 55.5 (2017), pp. 603–628. doi: 10.1177/0735633116674213.
- [LKF15] Kristine Ludvigsen, Rune Krumsvik, and Bjarte Furnes. "Creating formative feedback spaces in large lectures." In: *Computers & Education* 88 (2015), pp. 48–63. doi: 10.1016/j.compedu.2015.04.002.
- [LMW08] Nathaniel Lasry, Eric Mazur, and Jessica Watkins. "Peer instruction: From Harvard to the two-year college." In: *American journal of Physics* 76.11 (2008), pp. 1066–1069. doi: 10.1119/1.2978182.
- [LPT00] Maureen J Lage, Glenn J Platt, and Michael Treglia. "Inverting the Classroom: A Gateway to Creating an Inclusive Learning Environment." In: *The journal of economic education* 31.1 (2000), pp. 30–43. doi: 10.1080/00220480009596759.
- [LR97] Min Liu and Keith Rutledge. "The effect of a "learner as multimedia designer" environment on at-risk high school students' motivation and learning of design knowledge." In: *Journal of Educational Computing Research* 16.2 (1997), pp. 145–177. doi: 10.2190/27AT-YLJ3-3PVV-L0JY.

- [LSL17] Katja Lehmann, Matthias Söllner, and Jan Marco Leimeister. "Increasing Learner Interaction in Large-Scale Lectures by Using a Mobile Learning Application." In: *Blended Learning: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2017, pp. 1453–1471. doi: 10.4018/978-1-5225-0783-3.ch070.
- [LY11] Wen-Chuan Lin and Shu Ching Yang. "Exploring Students' Perceptions of Integrating Wiki Technology and Peer Feedback into English Writing Courses." In: *English Teaching: Practice and Critique* 10.2 (2011), pp. 88–103.
- [Mac18] Callum Macrae. *Vue.js: Up and Running: Building Accessible and Performant Web Apps*. O'Reilly Media, Inc., 2018.
- [Mar07] Margie Martyn. "Clickers in the Classroom: An Active Learning Approach." In: *Educause quarterly* 30.2 (2007), p. 71.
- [Mar17] Tom Marrs. *JSON at work: practical data integration for the web*. O'Reilly Media, Inc., 2017.
- [MB19] Sebastian Mader and François Bry. "Audience Response Systems Reimagined." In: *International Conference on Web-Based Learning*. Springer, 2019, pp. 203–216. doi: 10.1007/978-3-030-35758-0_19.
- [MH16] Kalpani Manathunga and Davinia Hernández-Leo. "PyramidApp: Scalable Method Enabling Collaboration in the Classroom." In: *European Conference on Technology Enhanced Learning*. Springer, 2016, pp. 422–427. doi: 10.1007/978-3-319-45153-4_37.
- [MH97] Eric Mazur and Robert C. Hilborn. "Peer instruction: A user's manual." In: *Physics Today* 50.4 (1997), p. 68.
- [Mil+15] Kelly Miller, Julie Schell, Andrew Ho, Brian Lukoff, and Eric Mazur. "Response switching and self-efficacy in Peer Instruction classrooms." In: *Physical Review Special Topics-Physics Education Research* 11.1 (2015), p. 010104. doi: 10.1103/PhysRevSTPER.11.010104.
- [MMN18] Manfred Meyer, Thomas Müller, and Andrea Niemann. "Serious Lecture vs. Entertaining Game Show - Why we need a Combination for improving Teaching Performance and how Technology can help." In: *Proceedings of the 10th International Conference on Education and New Learning Technologies (EDULEARN)*. 2018.
- [MX10] Qiao Mengduo and Jin Xiaoling. "Jigsaw Strategy as a Cooperative Learning Technique: Focusing on the Language Learners." In: *Chinese Journal of Applied Linguistics (Foreign Language Teaching & Research Press)* 33.4 (2010).
- [Nar08] Susanne Narciss. "Feedback Strategies for Interactive Learning Tasks." In: *Handbook of research on educational communications and technology* 3 (2008), pp. 125–144.
- [NE18] Stavros A. Nikou and Anastasios A. Economides. "Mobile-based assessment: A literature review of publications in major referred journals from 2009 to 2018." In: *Computers & Education* 125 (2018), pp. 101–119. doi: 10.1016/j.compedu.2018.06.006.
- [NM06] David J. Nicol and Debra Macfarlane-Dick. "Formative assessment and self-regulated learning: a model and seven principles of good feedback practice." In: *Studies in higher education* 31.2 (2006), pp. 199–218. doi: 10.1080/03075070600572090.
- [Nus+19] Aliya Nusrath, Shilpashree Yeliyur Dhananjaya, Namitha Dyavegowda, Rajeshwari Arasegowda, Asharani Ningappa, and Rafiya Begum. "Jigsaw Classroom: Is it an Effective Method of Teaching and Learning? Student's Opinions and Experience." In: *Journal of Clinical & Diagnostic Research* 13.2 (2019).
- [OK13] James Oigara and Jared Keengwe. "Students' perceptions of clickers as an instructional tool to promote active learning." In: *Education and Information Technologies* 18.1 (2013), pp. 15–28. doi: 10.1007/s10639-011-9173-9.
- [OMG06] Object Management Group (OMG). *About the Object Constraint Language Specification Version 2.0*. 2006. URL: <https://www.omg.org/spec/OCL/2.0/>. Last access on 10/8/2021.

- [OP15] Jacqueline O’Flaherty and Craig Phillips. “The use of flipped classrooms in higher education: A scoping review.” In: *The Internet and Higher Education* 25 (2015), pp. 85–95. doi: 10.1016/j.iheduc.2015.02.002.
- [Pay94] John W. Payne. “Thinking aloud: Insights into information processing.” In: *Psychological Science* 5.5 (1994), pp. 241–248.
- [PCG17] Saheed Popoola, Jeffrey Carver, and Jeff Gray. “Modeling as a Service: A Survey of Existing Tools.” In: *MODELS (Satellite Events)*. 2017, pp. 360–367.
- [PD20] Franziska Perels and Laura Dörrenbächer. “Selbstreguliertes Lernen und (technologiebasierte) Bildungsmedien.” In: *Handbuch Bildungstechnologie. Konzeption und Einsatz digitaler Lernumgebungen* (2020), pp. 81–92. doi: 10.1007/978-3-662-54368-9_5.
- [Pei21] Robert Peine. *Untersuchung einer vorschlagsbasierten Hilfsfunktion zur Unterstützung der Modellierung in der digitalen Lernumgebung stARS*. 2021. URL: https://www.rn.inf.tu-dresden.de/amcs/arbeiten/MA_2021_Peine_-_Untersuchung_einer_vorschlagsbasierten_Hilfsfunktion_zur_Unterstuetzung_der_Modellierung_in_der_digitalen_Lernumgebung_stARS.pdf. Last access on 10/8/2021.
- [Pio11] Michael Piotrowski. “QTI: A Failed E-Learning Standard?” In: *Handbook of research on E-learning standards and interoperability: Frameworks and issues*. IGI Global, 2011, pp. 59–82. doi: 10.4018/978-1-61692-789-9.ch004.
- [PNP17] Stefan Parker, Gerhard Nussbaum, and Stephan Pölzer. “The WebACS – An Accessible Graphical Editor.” In: *Harnessing the Power of Technology to Improve Lives*. IOS Press, 2017, pp. 890–893. doi: 10.3233/978-1-61499-798-6-890.
- [Pri04] Michael Prince. “Does Active Learning Work? A Review of the Research.” In: *Journal of engineering education* 93.3 (2004), pp. 223–231. doi: 10.1002/j.2168-9830.2004.tb00809.x.
- [Pro21] Projekt Weiterbildung selbstorganisiert. *Methodenkoffer SGL*. 2021. URL: <https://methodenkoffer-sgl.de/enzyklopaedie/>. Last access on 10/8/2021.
- [PSW08] Artem Polyvyanny, Sergey Smirnov, and Mathias Weske. “Process Model Abstraction: A Slider Approach.” In: *2008 12th International IEEE Enterprise Distributed Object Computing Conference*. IEEE, 2008, pp. 325–331. doi: 10.1109/EDOC.2008.17.
- [Qui16a] Klaus Quibeldey-Cirkel. “Einsatzoptionen des Audience-Response-Systems ARSnova in den ICM-Phasen Vorbereitung und Präsenz.” In: *Das Inverted Classroom Modell* (2016), p. 109.
- [Qui16b] Klaus Quibeldey-Cirkel. “Lernwiderstände sichtbar machen mit dem Audience Response System ARSnova.” In: *Wi(e)derstände: Digitaler Wandel in Bildungseinrichtungen. fraMediale, Bd 5* (2016), pp. 183–198.
- [Rad+20] Rajapandian Radha, K Mahalakshmi, V Sathish Kumar, and AR Saravanakumar. “E-Learning during lockdown of Covid-19 pandemic: A global perspective.” In: *International Journal of Control and Automation* 13.4 (2020), pp. 1088–1099.
- [Rei+12] Wolfgang Reinhardt, Michael Sievers, Johannes Magenheimer, Dennis Kundisch, Philipp Herrmann, Marc Beutner, and Andrea Zoyke. “PINGO: Peer Instruction for Very Large Groups.” In: *EC-TEL 2012 – Proceedings of the 7th European Conference of Technology Enhanced Learning*. Springer, 2012, pp. 507–512. doi: 10.1007/978-3-642-33263-0_51.
- [Rei12] Kersten Reich. *Methodenpool*. 2012. URL: http://methodenpool.uni-koeln.de/frameset_uebersicht.htm. Last access on 10/8/2021.
- [Rei18] Robert A. Reiser. “What field did you say you were in? Defining and naming our field.” In: *Trends and Issues in Instructional Design and Technology* 4 (2018).
- [Riv19] Christopher J Rivera. “Using ClassDojo as a Mechanism to Engage and Foster Collaboration in University Classrooms.” In: *College Teaching* 67.3 (2019), pp. 154–159. doi: 10.1080/87567555.2018.1505710.
- [Rob00] Lorraine J Robertson. “Twelve tips for using a computerised interactive audience response system.” In: *Medical Teacher* 22.3 (2000), pp. 237–239. doi: 10.1080/01421590050006179.

- [Rob06] Kristina Robertson. *Increase Student Interaction with "Think-Pair-Shares" and "Circle Chats"*. 2006. URL: <https://www.colorincolorado.org/article/increase-student-interaction-think-pair-shares-and-circle-chats>. Last access on 10/8/2021.
- [Rob11] Steven Robbins. "Beyond clickers: using ClassQue for multidimensional electronic classroom interaction." In: *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (2011), pp. 661–666. doi: 10.1145/1953163.1953347.
- [Rod+16] María Jesús Rodríguez-Triana, Adrian Holzer, Luis P. Prieto, and Denis Gillet. "Examining the Effects of Social Media in Co-located Classrooms: A Case Study Based on SpeakUp." In: *European Conference on Technology Enhanced Learning*. Springer. 2016, pp. 247–262. doi: 10.1007/978-3-319-45153-4_19.
- [Ros19] Lidia Roszko. *Entwicklung eines graphischen Editors zur Erstellung von beliebigen Lernszenarien in Audience Response Systemen*. 2019. URL: https://www.rn.inf.tu-dresden.de/amcs/arbeiten/BA_2019-Roszko_-_Entwicklung_eines_graphischen_Editors_zur_Erstellung_von_beliebigen_Lernszenarien_in_Audience_Response_Systemen.pdf. Last access on 10/8/2021.
- [Sch+17] Beat A. Schwendimann, Luis P. Prieto, Mina Shirvani Boroujeni, Pierre Dillenbourg, Maria Jesus Rodriguez-Triana, Andrii Vozniuk, Adrian Holzer, and Denis Gillet. "Perceiving Learning at a Glance: A Systematic Literature Review of Learning Dashboard Research." In: *IEEE Transactions on Learning Technologies* 10.1 (2017), pp. 30–41. issn: 1939-1382. doi: 10.1109/TLT.2016.2599522.
- [Sch11] Bjarne Schmidt. "Teaching engineering dynamics by use of peer instruction supported by an audience response system." In: *European Journal of Engineering Education* 36.5 (2011), pp. 413–423. doi: 10.1080/03043797.2011.602185.
- [Sch16] Daniel Schön. "Customizable Teaching on Mobile Devices in Higher Education." PhD thesis. 2016.
- [Sch20] Hendrik Schön. "Role-based Adaptation of Business Reference Models to Application Models." PhD thesis. Technische Universität Dresden, 2020.
- [Sch95] Janet Schweitzer. "The Use of Learning Stations as a Strategy for Teaching Concepts by Active-Learning Methods." In: *Journal of Geological Education* 43.4 (1995), pp. 366–370. doi: 10.5408/0022-1368-43.4.366.
- [SDG19] Eleni Seralidou, Christos Douligeris, and Christina Gralista. "EduApp: A Collaborative Application for Mobile Devices to Support the Educational Process in Greek Secondary Education." In: *2019 IEEE Global Engineering Education Conference (EDUCON)*. IEEE. 2019, pp. 189–198. doi: 10.1109/EDUCON.2019.8725175.
- [Shm18] Ilja Shmelkin. *Untersuchung der Adaptierbarkeit webbasierter Audience Response Systeme*. 2018. URL: https://www.rn.inf.tu-dresden.de/amcs/arbeiten/SCC_HS_WS2018-Shmelkin_-_Untersuchung_der_Adaptierbarkeit_webbasierter_Audience_Response_Systeme.pdf. Last access on 10/8/2021.
- [Shm19] Ilja Shmelkin. *Entwicklung eines Metamodells zur Beschreibung kontext-sensitiver Audience Response Systeme*. 2019. URL: https://www.rn.inf.tu-dresden.de/amcs/arbeiten/MA_2019-Shmelkin_-_Entwicklung_eines_Metamodells_zur_Beschreibung_kontext-sensitiver_Audience_Response_Systeme.pdf. Last access on 10/8/2021.
- [Siv+19] Rebecca T. Sivarajah, Nicole E. Curci, Elizabeth M. Johnson, Diana L. Lam, James T. Lee, and Michael L. Richardson. "A Review of Innovative Teaching Methods." In: *Academic radiology* 26.1 (2019), pp. 101–113. doi: 10.1016/j.acra.2018.03.025.
- [Sla80] Robert E. Slavin. "Cooperative Learning." In: *Review of educational research* 50.2 (1980), pp. 315–342. doi: 10.3102/00346543050002315.
- [SM15] Julie Schell and Eric Mazur. "Flipping the Chemistry Classroom with Peer Instruction." In: *Chemistry Education: Best Practices, Opportunities and Trends*. Wiley Online Library (2015). doi: 10.1002/9783527679300.ch13.

- [SMB13] Bruno Sampaio, Carmen Morgado, and Fernanda Barbosa. "Building collaborative quizzes." In: *Proceedings of the 13th Koli Calling International Conference on Computing Education Research*. 2013, pp. 153–159. doi: 10.1145/2526968.2526985.
- [SR78] John Stuart and R.J. Desmond Rutherford. "Medical student concentration during lectures." In: *The Lancet* (1978). doi: 10.1016/S0140-6736(78)92233-X.
- [Ste00] Friedrich Steimann. "On the representation of roles in object-oriented and conceptual modelling." In: *Data & Knowledge Engineering* 35.1 (2000), pp. 83–106. doi: 10.1016/S0169-023X(00)00023-9.
- [Str+98] Susanne Strahringer *et al.* "Ein sprachbasierter Metamodellbegriff und seine Verallgemeinerung durch das Konzept des Metaisierungsprinzips." In: *Modellierung*. Vol. 98. 1998, pp. 15–20.
- [Str16] Susanne Strahringer. "Modelle und Metamodellierung." In: *Geschäftsprozessorientierte Systementwicklung*. Springer, 2016, pp. 11–23. doi: 10.1007/978-3-658-14826-3_2.
- [Swa17] Ruth Swart. "Critical thinking instruction and technology enhanced learning from the student perspective: A mixed methods research study." In: *Nurse education in practice* 23 (2017), pp. 30–39. doi: 10.1016/j.nepr.2017.02.003.
- [Tai18] Nguonly Taing. "Run-time Variability with Roles." PhD thesis. Technische Universität Dresden, 2018.
- [Tea21] Vue.js Team. *Vue.js Documentation - Comparison*. 2021. URL: <https://vuejs.org/v2/guide/comparison.html#Complexity>. Last access on 10/8/2021.
- [Tha21] Sinthujan Thanabalasingam. *Support of Lecturers in Modeling Complex Learning and Teaching Scenarios in Audience Response Systems*. 2021. URL: https://www.rn.inf.tu-dresden.de/amcs/arbeiten/MA_2021_Thanabalasingam_-_Support_of_Lecturers_in_Modeling_Complex_Learning_and_Teaching_Scenarios_in_Audience_Response_Systems.pdf. Last access on 10/8/2021.
- [Tre21] Google Trends. *E-Learning*. 2021. URL: <https://trends.google.com/trends/explore?date=today%205-y&q=%2Fg%2F121tbbp8>. Last access on 10/8/2021.
- [UE16] Harald Urwalek and Martin Ebner. "Potentiale von Smartwatches für Audience-Response-Systeme." In: *Zeitschrift für Hochschulentwicklung* 11.4 (2016), pp. 39–50.
- [UNE20] UNESCO. *Higher Education*. 2020. URL: <http://uis.unesco.org/en/topic/higher-education>. Last access on 10/8/2021.
- [Vet+14] Jonas Vetterick, Martin Garbe, Andreas Dähn, and Clemens H. Cap. "Classroom Response Systems in the Wild: Technical and Non-Technical Observations." In: *International Journal of Interactive Mobile Technologies (ijIM)* 8.1 (2014), p. 21. issn: 1865-7923. doi: 10.3991/ijim.v8i1.3454.
- [VM17] Valentin Velikov and Ivelin Mitev. *Graphical Editor for Android*. 2017. URL: http://suruse.uni-ruse.bg/files/Inf_ValVelikov_IMitev_1_2017.pdf. Last access on 10/8/2021.
- [WGF17] Manuel Wimmer, Irene Garrigós, and Sergio Firmenich. "Towards Automatic Generation of Web-Based Modeling Editors." In: *International Conference on Web Engineering*. Springer. 2017, pp. 446–454. doi: 10.1007/978-3-319-60131-1_31.
- [Wol+11] Bjørn H.K. Wolter, Mary A. Lundeborg, Hosun Kang, and Clyde F. Herreid. "Students' Perceptions of Using Personal Response Systems ("Clickers") With Cases in Science." In: *Journal of College Science Teaching* 40.4 (2011), p. 14.
- [Wol+15] Margaret Wolff, Mary Jo Wagner, Stacey Poznanski, Jocelyn Schiller, and Sally Santen. "Not Another Boring Lecture: Engaging Learners with Active Learning Techniques." In: *The Journal of emergency medicine* 48.1 (2015), pp. 85–93. doi: 10.1016/j.jemermed.2014.09.010.
- [Wol+17] Christian Wolters, Daniel Wessel, Finn Jacobsen, and Michael Herczeg. "Moving freely while staying on track—Smart Glasses to Support Lecturers." In: *Bildungsräume 2017* (2017).
- [Wut18] Markus Wutzler. "On-Demand Composition of Smart Service Systems in Decentralized Environments." PhD thesis. Technische Universität Dresden, 2018.

- [Yar08] Sarita Yardi. *Whispers in the Classroom*. MacArthur Foundation Digital Media and Learning Initiative, 2008.
- [Zim00] Barry J Zimmerman. "Attaining Self-Regulation: A Social Cognitive Perspective." In: *Handbook of self-regulation*. Elsevier, 2000, pp. 13–39. doi: 10.1016/B978-012109890-2/50031-7.

List of Figures

1.1	The global search interest of the topic “E-Learning” over the past five years.	1
2.1	The schematic representation of a variant of <i>Peer Instruction</i>	10
2.2	The formation of home groups and expert groups in a Jigsaw Classroom for 25 students.	11
2.3	The definition of (meta-)models based on (meta-)model languages. . .	19
3.1	The classification of digital learning environments by the type of usage.	22
3.2	The classification of digital learning environments by the type of interaction.	23
3.3	The comparison of two questions with the same type but different purpose of application.	24
3.4	The comparison of two <i>backchannel</i> functions that allow students to initiate interactions.	27
3.5	The filter method used as well as an excerpt of the resulting classification of 50 systems.	30
3.6	A summary of the first search query to confirm the problem statements.	35
3.7	A summary of the second search query to explore the state of the art of the problem statements and objective.	36
3.8	A summary of research fields that were investigated by the state of the art.	38
3.9	The composition of core components into teaching methods that are composed into teaching formats.	40
3.10	A proposal-based help functionality to support lecturers in choosing an appropriate range of functions.	42
3.11	A generic model to define customized teaching scenarios.	44
3.12	The <i>question & answer</i> functionality provided by Backstage2.	46
3.13	A collaborative group activity with different roles defined for different tasks.	47
4.1	A summary of the conceptual idea for an adaptable collaborative learning environment.	54
4.2	A paper-based construction kit to build customized teaching scenarios.	67
4.3	A representation of <i>Peer Instruction</i> that was used in the evaluation of the (meta-)model.	68
4.4	A summary of the evaluation results for the (meta-)model.	69
4.5	A preliminary concept of the graphical editor that was created as a result of the initial user study.	73

4.6	A refined concept of the graphical editor that was created after the second user study.	75
4.7	The final mockup of the graphical editor, in which the general design and concept decisions are visualized.	77
4.8	The first step of an <i>overlay</i> that is opened on the first usage of the editor.	81
4.9	The initial concept of a <i>decision tree</i> that is able to select suitable elements based on the type of interaction.	81
4.10	The concept of a save dialog to store a template on the server that can later be filtered by its name, description or category.	85
4.11	The concept of a load/import dialog that enables to either load templates into the existing scenario or replace them entirely.	86
4.12	The concept of visualizing added templates as separate containers that can later be edited, integrated or swapped.	87
4.13	The concept of an infrastructure for an adaptable collaborative learning environment.	89
4.14	Mockup of the scenario overview with the abilities to create new scenarios and edit or delete existing ones.	90
4.15	Mockup of the scenario management with the abilities to start, pause, resume and stop (i.e., reset) instances, as well as manage the workflow by finishing blocks.	91
4.16	Mockup of the student view with the ability to interact with the currently active blocks.	92
4.17	An exemplary role model for the didactic scenario of <i>Peer Instruction</i>	97
4.18	A visualization of possible advantages of role-based modeling when implementing our (meta-)model.	98
4.19	Mockup of the adjusted scenario management, in which the scenario can be extended next to a selected element when pressing the button with the plus icon.	105
4.20	Mockup of the extension preview, which checks the model and gives hints.	106
5.1	A global picture of the main components involved in the implementation of stARS.	110
5.2	A simplified version of the (meta-)model that shows the first and second hierarchical levels of inheritance.	114
5.3	An overview of the (meta-)model to visualize its size, as well as an excerpt.	116
5.4	The final version of the graphical editor of stARS.	124
5.5	The <i>main menu</i> of the graphical editor of stARS.	124
5.6	A comparison of the <i>element palette</i> of bpmn-js and stARS.	126
5.7	A comparison of the <i>context pad</i> of bpmn-js and stARS.	127
5.8	A comparison of the <i>properties panel</i> of the extension of bpmn-js and stARS.	128
5.9	The implementation of a suggestion function for suitable next blocks.	129
5.10	The implementation of <i>snap lines</i> , a <i>grid</i> as well as the <i>automatic connection</i> of blocks.	130
5.11	An invalid scenario that is validated by the model checking functionality.	131

5.12	A scenario in which a template was added to the existing model. . . .	132
5.13	The result of the template generation, in which different representations of templates as well as popularity, frequency and size are displayed. .	132
5.14	The infrastructure of the prototype called scenario-tailored Audience Response System (stARS).	134
5.15	The scenario overview of a lecturer with the options to create a new scenario as well as edit or delete existing scenarios.	136
5.16	The scenario view with the options to start, pause, resume or stop the scenario, as well as managing the workflow and showing the real-time results of the currently active blocks.	137
5.17	The student view, in which the currently active blocks can be used. Moreover, a random pseudonym that was generated for the scenario is displayed.	138
5.18	The role hierarchy that is implemented in stARS.	139
5.19	A comparison of the extension options in the model and list view. . . .	142
5.20	The extension dialog to select one of three options to extend the scenario.	143
5.21	The two views of the editor to extend the scenario, which can be switched using the tabs on the bottom left.	143
5.22	The extension preview dialog, in which the model is checked, previewed and a hint regarding the integration is given.	144
5.23	The extended scenario in which the previously created sub-process is integrated. Moreover, an option to undo the last extension is provided.	144
5.24	A possible representation of a lecture that is supported by <i>Interactive Learning Questions</i> at the beginning, the middle and the end of the lecture.	145
5.25	A possible representation of one iteration of a variant of <i>Peer Instruction</i> that allows supporting both <i>conceptTest</i> and <i>peer discussion</i>	146
5.26	A possible representation of a lecture that integrates the <i>Jigsaw</i> teaching technique.	147
5.27	A possible representation of a lecture that integrates <i>Think-Pair-Share</i> . .	148
5.28	A possible representation of a lecture in which four <i>Learning Stations</i> are visited by different groups of students.	149
5.29	An optimized and more flexible representation of a lecture, in which four <i>Learning Stations</i> are visited by different groups of students. . . .	150
5.30	A possible representation of a <i>Peer Feedback</i> scenario that spans over multiple days.	151
5.31	A possible representation of a student-centered <i>Learners-as-Designers</i> scenario that spans an entire semester.	153
6.1	An interpretation of the SUS score into acceptability, school grades and adjectives.	162
6.2	The results for lecturers' expected functionalities of <i>digital learning environments</i>	165
6.3	The scenario of the first lecture experiment conducted by Dr. Marius Feldmann, which includes two rounds of questions as well as an advanced group activity with textual discussions and group voting. . . .	177

6.4	The scenario of the second lecture experiment conducted by Dr. Iris Braun, which includes a traditional <i>Peer Feedback</i> situation running over multiple days.	181
6.5	The scenario of the third lecture experiment conducted by Dr. Felix Kapp, in which an advanced <i>Peer Feedback</i> situation is represented.	184
6.6	The scenario of the fourth lecture experiment conducted by Dr. Marius Feldmann, which includes both a question round and an advanced group activity with a group voting.	188
7.1	The usage of eye-tracking to trace the view paths of lecturers when searching a functionality.	204

List of Tables

2.1	A summary of the features of roles.	16
2.2	A summary of CROM's <i>meta-types</i> that are distinguished by their ontological <i>meta-properties</i>	17
3.1	A summary of the related work for lecturer support, collaboration and adaptation in <i>digital learning environments</i>	48
4.1	The fulfilled requirements of different approaches that can be used as an underlying (meta-)model.	57
4.2	An overview of the structural blocks and their parameters included in our concept.	58
4.3	An overview of the transition blocks and their parameters included in our concept.	60
4.4	An overview of the functional blocks and parameters to initiate interactions as a lecturer.	62
4.5	An overview of the functional blocks and parameters to initiate interactions as a student.	65
4.6	An overview of the blocks for visualization and their parameters that are included in our concept.	66
5.1	A summary of <i>graphical language workbenches</i>	112
5.2	The requirements for selecting a library to create diagrams.	122
5.3	A comparison of requirements supported by libraries for graphical editors.	123
6.1	The results of the System Usability Scale for each participant.	162
6.2	A summary of lecturers' answers to their current usage, preference for future usage and opinion regarding technical support of different teaching scenarios.	166
6.3	The summarized results of the sorting questions as well as the questions regarding the intuitiveness of both graphical representation and execution.	169
6.4	The summarized results of the question, whether a task could be solved easily.	174
6.5	The summarized results of the question, whether the results are satisfactory.	174
7.1	The verification of the sub research theses of RT1.	200
7.2	The verification of the sub research theses of RT2.	201
7.3	The verification of the sub research theses of RT3.	202

A.1	A comprehensive list of all parameters, the function and the block(s), in which they occur.	E
B.1	The different build schemes of the <i>GroupBuilder</i> with a description and an example.	H
B.2	The different build schemes of the <i>PeerBuilder</i> with a description and an example.	J
D.1	The results of the sorting question for <i>Interactive Learning Questions</i> . . .	AS
D.2	The results of the sorting question for <i>Peer Instruction</i>	AT
D.3	The results of the sorting question for <i>Jigsaw Classroom</i>	AU
D.4	The results of the sorting question for <i>Think-Pair-Share</i>	AV
D.5	The results of the sorting question for <i>Learning Stations</i>	AW

List of Listings

- 5.1 An Acceleo transformation template that converts the (meta-)model into JSON. 118
- 5.2 An excerpt of the JSON output that was generated by the Acceleo template. 119

Appendix A (Meta-)Model Parameters

Parameter	Description	Block(s)
<i>accessControl</i>	The way students can access the scenario.	StartBlock
<i>advertise</i>	Advertise the scenario on the project's landing page.	StartBlock
<i>allowAbstention</i>	Is it allowed to abstain from answering?	LearningQuestion, SurveyQuestion
<i>allowAnswering</i>	Will answering questions be allowed?	OpenDiscussion
<i>allowedMinNumber</i>	The minimum number that is allowed to input.	NumericalLearningQuestion, NumericalSurveyQuestion
<i>allowedMaxNumber</i>	The maximum number that is allowed to input.	NumericalLearningQuestion, NumericalSurveyQuestion
<i>allowMarkCorrect-Answer</i>	Will the questioner be able to mark a correct answer?	OpenDiscussion
<i>allowVotingAnswers</i>	Will voting of answers be allowed?	OpenDiscussion
<i>allowVotingQuestions</i>	Will voting of questions be allowed?	OpenDiscussion
<i>anonymity</i>	What level of anonymity should be used?	StartBlock
<i>answerFeedback</i>	Should feedback on the correctness of the answer be given?	LearningQuestion
<i>audioVideoChatUrl</i>	Does an URL already exist? Otherwise a Jitsi link is generated.	ActivityBlock, LectureBlock

Parameter	Description	Block(s)
<i>autoFinishAfterTimeout</i>	Should the block be finished automatically after the <i>time-out</i> is reached?	FunctionBlock, ActivityBlock, LectureBlock, PauseBlock, VisualizationBlock
<i>buildSchema</i>	What building schema should be used?	GroupBuilder, PeerBuilder
<i>caseSensitive</i>	Differentiate between upper and lower case letters?	FreertextLearningQuestion
<i>characterLimit</i>	Set a fixed limit of characters.	FreertextLearningQuestion, FreertextSurveyQuestion
<i>characterMinimum</i>	Set a fixed minimum of characters.	FreertextLearningQuestion, FreertextSurveyQuestion
<i>choices</i>	The answers that can be selected.	ClosedFeedback, SingleChoiceLearningQuestion, MultipleChoiceLearningQuestion, OrderLearningQuestion, SingleChoiceSurveyQuestion, MultipleChoiceSurveyQuestion
<i>conditions</i>	The conditions for determining the subsequent path.	OrFork
<i>comment</i>	Set a comment for the graphical representation.	FunctionBlock, ActivityBlock, LectureBlock, PauseBlock, VisualizationBlock
<i>completeText</i>	The complete text without gaps.	GapTextQuestion
<i>content</i>	Define content using Mark-down syntax.	PresentMaterial
<i>cooldown</i>	The cooldown to give feedback again.	ClosedFeedback
<i>correctMinNumber</i>	The minimum number that is correct.	NumericalLearningQuestion
<i>correctMaxNumber</i>	The maximum number that is correct.	NumericalLearningQuestion
<i>correctOrderArray</i>	Define the correct order.	OrderLearningQuestion
<i>correctPairArray</i>	Define the correct pairs.	MatchingLearningQuestion
<i>correctRange</i>	Define a range for the coordinate that is set in <i>correctValue</i> .	HotspotLearningQuestion

Parameter	Description	Block(s)
<i>correctText</i>	Define the correct text/word.	FreetextLearningQuestion
<i>correctValue</i>	Define the correct coordinate.	HotspotLearningQuestion
<i>defaultDestinationID</i>	The default target of a conditional fork.	DecisionFork, OrFork
<i>destinationID</i>	The ID of the target element of a transition.	DefaultTransition
<i>destinationIDs</i>	The IDs of the (potential) target elements of a fork.	AndFork, DecisionFork, OrFork
<i>displayPolicy</i>	What policy should be considered when displaying a block?	Feedback, LearningQuestion, SurveyQuestion
<i>displayType</i>	How should the block be displayed?	ClosedFeedback, LearningQuestion, SurveyQuestion
<i>feedbackText</i>	The formulation for the feedback.	ClosedFeedback
<i>feedbackTexts</i>	Define feedback that is provided on multiple answer repetitions.	LearningQuestion
<i>filter</i>	Limit the block to specific (groups of) users.	FunctionBlock, ActivityBlock, LectureBlock, PauseBlock, VisualizationBlock
<i>functionBlock</i>	The function block that is taken into account.	GroupVoting, PresentGroupAnswers
<i>functionBlocks</i>	The function blocks that are taken into account.	GroupBuilder, PeerBuilder, PresentResult
<i>gapPositions</i>	Define the words that are replaced by gaps.	GapTextQuestion
<i>groupBuilder</i>	To which group builder does the interaction refer?	GroupAudioVideoChat, GroupChat, GroupVoting, PresentGroupAnswers
<i>groupSize</i>	How many members should each group have?	GroupBuilder
<i>hasAudioVideoChat</i>	Should an audio/video chat be added?	ActivityBlock, LectureBlock

Parameter	Description	Block(s)
<i>image</i>	An uploaded image can be referenced.	HotspotLearningQuestion, HotspotSurveyQuestion
<i>imageUrl</i>	An external image can be referenced.	HotspotLearningQuestion, HotspotSurveyQuestion
<i>isAnswerChangeable</i>	Can answers be changed after submission?	SurveyQuestion
<i>misdirectionWords</i>	Define additional words that do not match a gap.	GapTextQuestion
<i>numberOfAssignments</i>	The number of assignments for the peer feedback.	PeerBuilder
<i>numberOfGroups</i>	How many groups should be formed?	GroupBuilder
<i>numberOfRepetitions</i>	The number of attempts to answer a question.	LearningQuestion
<i>peerBuilder</i>	To which peer builder does the interaction or question refer?	PeerChat, PresentPeerAnswers, PresentPeerFeedback, SurveyQuestion
<i>pin</i>	The PIN that is set for the scenario, if <i>pin</i> is selected as the <i>accessControl</i> .	StartBlock
<i>predefinedAnswers</i>	Can learners choose from given words to put into the gaps?	GapTextQuestion
<i>questions</i>	Select survey questions for which feedback is collected.	PeerBuilder
<i>questionText</i>	The formulation of the question.	LearningQuestion, SurveyQuestion
<i>recordConfidence</i>	Should an input of a confidence be required?	LearningQuestion
<i>scenarioDate</i>	The starting date/time of the scenario.	StartBlock
<i>scenarioName</i>	The name of the scenario.	StartBlock
<i>showAggregate</i>	Should the aggregated answers of other students be shown?	ClosedFeedback, SingleChoiceSurveyQuestion, MultipleChoiceSurveyQuestion

Parameter	Description	Block(s)
<i>shortAnswer</i>	Should the input be just one word?	FreertextLearningQuestion, FreertextSurveyQuestion
<i>showCorrectPercentage</i>	Should the correct percentage of other students be shown?	LearningQuestion
<i>sourceID</i>	The ID of the source element of a transition.	TransitionBlock
<i>storagePolicy</i>	What policy should be considered when storing answers from a block?	Feedback, LearningQuestion, SurveyQuestion
<i>task</i>	Which task should be solved in the activity or discussed in the group / peer chat?	ActivityBlock, GroupChat, PeerChat
<i>textType</i>	Which type of text should be input?	FreertextLearningQuestion, FreertextSurveyQuestion
<i>topic</i>	Which topic does the lecture has?	LectureBlock
<i>timeout</i>	Set a fixed timeout (in seconds) for this block.	FunctionBlock, ActivityBlock, LectureBlock, PauseBlock, VisualizationBlock
<i>visibleForAll</i>	Should the discussion be visible for all students?	OpenDiscussion
<i>voteCount</i>	How many votes students got?	SurveyQuestion

Table A.1: A comprehensive list of all parameters, the function and the block(s), in which they occur. If an abstract block is mentioned under block(s), the parameter hold for all specific types, e.g., if *LearningQuestion* is listed, the parameter is available for *SingleChoiceLearningQuestion*, *MultipleChoiceLearningQuestion*, etc.

Appendix B Build Schemes

B.1 The Build Schemes of the Group Builder

Build Schema	Description	Example
<i>random</i>	The students are randomly divided into groups.	Let students interact in groups without asking prior questions.
<i>bestToWorst</i>	Forms groups of students based on their average correctness in one or more previously answered <i>LearningQuestions</i> – the best students are grouped with the worst students. If only one prior <i>LearningQuestion</i> exists, using <i>differentAnswer</i> is recommended, as the percentage of correctness is either 0% or 100%.	Scenarios, in which students benefit from the description of another student, e.g., the <i>Peer Discussion of Peer Instruction</i> .
<i>similar</i>	Forms groups of students based on their average correctness in one or more previously answered <i>LearningQuestions</i> – students with similar prior knowledge are grouped. As above, the usage of multiple <i>LearningQuestions</i> is recommended.	It is suitable for analytic scenarios, in which the lecturer target to evaluate the impact of students' prior knowledge on their ability to solve a specific task.

<i>sameAnswer</i>	Forms groups of students that have given the same answer on one specific prior <i>Learning-</i> or <i>SurveyQuestion</i> .	Divide the students into groups that have, for example, the same interests or opinions. In this way, they can work together on a task and present the results of their teamwork later in the lecture.
<i>differentAnswer</i>	Forms groups of students that have given different answers on one specific prior <i>Learning-</i> or <i>SurveyQuestion</i> .	Works best for each situation, in which students with different answers should be grouped, and only one question should be taken into account, e.g., the <i>Peer Discussion</i> of <i>Peer Instruction</i> .
<i>groupShuffle</i>	Refers to another group builder and builds groups that contain one student of each previously created group.	This build schema can be used to realize the <i>Jigsaw Classroom</i> , in which students of different expert groups should be shuffled into learning groups, containing one student of each expert group.
<i>groupMerge</i>	Refers to another group builder and merges two previously created groups into one group.	A popular scenario is <i>Think-Pair-Share</i> , in which, for instance, groups of two students are merged into groups of four students. This allows to continue the discussion beyond the groups of two students.

Table B.1: The different build schemes of the *GroupBuilder* with a description and an example.

B.2 The Build Schemes of the Peer Builder

Build Schema	Description	Example
<i>random</i>	Feedback receiver and feedback provider are randomly assigned.	Allows implementing peer feedback without having prior questions or group builders.
<i>bestToWorst</i>	The best students receive feedback from the worst students and vice versa. This is analogously handled for providing feedback.	Can be used to give students with low prior knowledge feedback from students with high prior knowledge and allows students with high prior knowledge to also consider feedback of students with potentially lower prior knowledge.
<i>similar</i>	Students with similar prior knowledge receive and provide feedback from each other.	Allows to let students consider feedback from another student with similar prior knowledge.
<i>sameAnswer</i>	Students that gave the same answer on one specific prior <i>Learning-</i> or <i>SurveyQuestion</i> receive and provide feedback from/to each other.	For example, students with the same study course receive and provide feedback.
<i>differentAnswer</i>	Students that gave different answers on one specific prior <i>Learning-</i> or <i>SurveyQuestion</i> receive and provide feedback from each other.	For example, students with different study courses receive and provide feedback.
<i>sameGroup</i>	Refers to a group builder and allows students to receive and provide feedback from/to other students of the same group.	Students from the same learning group should receive and provide feedback.

<i>differentGroup</i>	Refers to a group builder and allows students to receive and provide feedback from/to students of another group.	Students from different learning groups should receive and provide feedback.
-----------------------	--	--

Table B.2: The different build schemes of the *PeerBuilder* with a description and an example.

Appendix C User Study on Usability (Mid 2020)

C.1 Slides of the User Study (German)

Evaluation von stARS: scenario-tailored Audience Response System

- Ansatz zur **Aktivierung von Studierenden** durch Benutzung von mobilen Endgeräten
- **Anpassung an das aktuelle Szenario** (Art der Lehrveranstaltung sowie Lehrstrategie des Lehrenden):
 - Abbildung der **Lehrveranstaltung** und **aller stattfindenden Aktivitäten** (reine Präsentation, Anzeige von Medien, Fragerunden oder online Gruppeninteraktionen) mittels **Workflow**
 - **Parallel-** sowie **bedingt** ausgeführte Pfade
 - **Anpassung** von Funktionen **durch Parameter**

Ablauf der Evaluation: Verschiedene Teilbereiche

Teil 1	Einschätzung Ihrer persönlichen Lehrstrategie und -erfahrung
Teil 2	Benutzung des stARS-Editors zur Modellierung von zwei Lehrszenarien sowie Einschätzung der Benutzerfreundlichkeit des Editors (<i>vorgestellt durch Tommy Kubica</i>)
Teil 3 und 4	Präsentation von zwei verschiedenen Gruppen von Vorschlägen zur Optimierung des Editors (<i>vorgestellt durch Lidia Roszko und Sinthujan Thanabalasingam</i>)

Bitte zögern Sie nicht, uns bei Unklarheiten zu fragen.

Vielen Dank für Ihre Bereitschaft zur Teilnahme an dieser Evaluation!

C.2 Questionnaire of the User Study (German)

Umfrage für die Evaluation von stARS

Sehr geehrte Damen und Herren,
vielen Dank, dass Sie sich die Zeit nehmen, um an der Evaluation von stARS (scenario-tailored Audience Response System) teilzunehmen.

In den folgenden 60 Minuten sollen Sie den aktuellen Stand sowie Vorschläge zur Optimierung des stARS-Editors zur Modellierung von eigenen Lehrszenarien bewerten.

Wir möchten Sie darauf hinweisen, dass es bei der Evaluation keine richtigen oder falschen Antworten gibt und bitten Sie die einzelnen Fragen möglichst ehrlich zu beantworten. Sofern Unklarheiten bei der Evaluation auftreten, fragen Sie bitte jederzeit bei uns nach.

1) Fragen zur Lehrstrategie und -erfahrung

Bitte beantworten Sie die folgenden Fragen.

Wie lange betreiben Sie bereits Lehre bzw. haben Sie Lehre betrieben? *

- ☐ Bis zu 1 Jahr
- ☐ Zwischen 1 und 3 Jahre
- ☐ Zwischen 3 und 5 Jahre
- ☐ Zwischen 5 und 10 Jahre
- ☐ Über 10 Jahre
- ☐ Sonstiges: _____

Welche Arten von Lehrveranstaltungen haben Sie bereits gehalten? *

- ☐ < 20 Teilnehmer (z.B. Übung, Seminar)
- ☐ 20 - 49 Teilnehmer (z.B. große Übung, kleine Vorlesung)
- ☐ 50 - 150 Teilnehmer (z.B. Vorlesung)
- ☐ > 150 Teilnehmer (z.B. große Vorlesung)
- ☐ Sonstiges: _____

Halten Sie während der Corona-Krise Lehrveranstaltungen? Wenn ja, welche Strategie(n) benutzen Sie hierbei? *

Meine Antwort _____

Mit welchen technischen Werkzeugen unterstützen Sie allgemein Ihre Lehrveranstaltungen? *

- ☐ Präsentations-Programm (z.B. PowerPoint oder Keynote)
- ☐ Handschriftliche Notizen (z.B. über Tablet)
- ☐ Werkzeuge zur Aktivierung von Studierenden (z.B. Audience Response Systeme, Backchannel Systeme)
- ☐ Vorlesungsvideos
- ☐ Aufgaben zur Vor- / Nachbereitung
- ☐ Sonstiges: _____

Sofern bereits benutzt: Wie oft haben Sie bereits Werkzeuge zur Aktivierung von Studierenden eingesetzt? Welche Werkzeuge haben Sie hierfür eingesetzt und welche Erfahrung haben Sie dabei gemacht? *

Meine Antwort _____

Haben Sie vor dieser Evaluation bereits schon einmal mit dem stARS-Editor gearbeitet oder das stARS-Metamodell evaluiert? *

- ☐ Editor
- ☐ Metamodell
- ☐ Nein

2) Bewertung der Benutzerfreundlichkeit

Im diesem Teil der Evaluation sollen Sie den stARS-Editor benutzen und sowohl vorgegebene Szenarien als auch ein eigenes Szenario modellieren. Zur besseren Einschätzung der Benutzbarkeit bitten wir Sie Ihren Bildschirm mit uns zu teilen.

Bitte navigieren Sie zu nachfolgender URL und loggen Sie sich als neuer Lehrender ein. Öffnen Sie anschließend das Fenster zur Modellierung eines neuen Szenarios.

<https://stars-project.com/>

Aufgabe 1

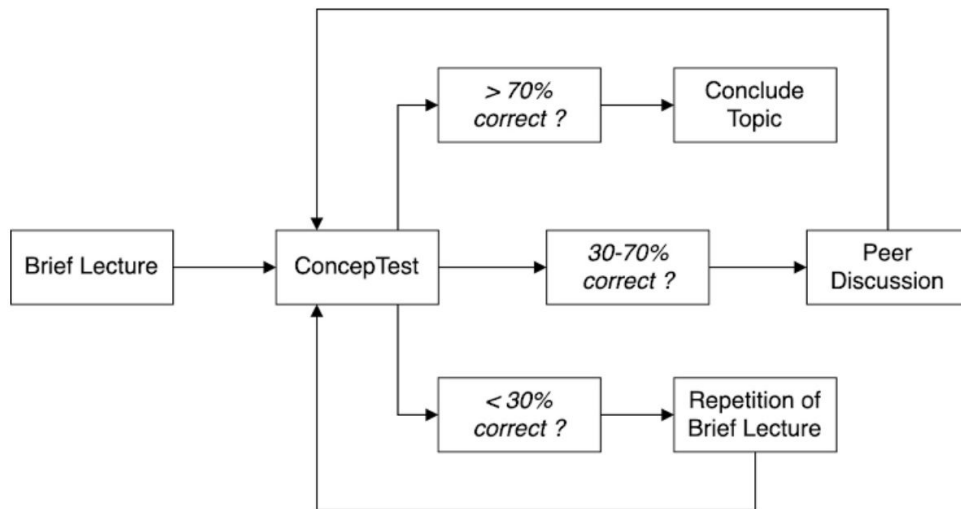
In der ersten Aufgabe sollen Sie eine einfache Lehrveranstaltung mit drei Inhaltsblöcken modellieren, deren Präsentation durch zwei Fragerunden unterbrochen wird.

Überlegen Sie sich hierzu im ersten Schritt durch welche Funktionalität Ihre Inhaltsblöcke unterstützt werden sollen. Fügen Sie diesen Block ein. Anschließend erweitern Sie Ihr Szenario um eine Fragerunde mit zwei parallelen Lernfragen mit einem Sub-Typ Ihrer Wahl. Erweitern Sie das Szenario um einen zweiten Inhaltsblock, eine zweite Fragerunde mit zwei parallelen Umfragen beliebigen Sub-Typs sowie um einen abschließenden Inhaltsblock.

Aufgabe 2)

In der zweiten Aufgabe sollen Sie einen Workflow modellieren, um die Lehrmethode "Peer Instruction" (siehe unten) umzusetzen.

Versuchen Sie diese Lehrmethode mittels geeigneter Blöcke in stARS abzubilden. Der Versuchsleiter wird Sie hierbei durch das Szenario leiten und steht Ihnen bei Fragen jederzeit unterstützend zur Verfügung.

Beispielhafter Ablauf von Peer Instruction**Bewertung der Benutzerfreundlichkeit**

Im letzten Abschnitt haben Sie den stARS-Editor kennengelernt. Nun bitten wir Sie diesen anhand einiger Fragen auf dessen Benutzerfreundlichkeit zu bewerten.

Bitte beantworten Sie folgende Fragen: *

	Stimme gar nicht zu	Stimme eher nicht zu	Weder noch	Stimme eher zu	Stimme voll zu
Ich kann mir sehr gut vorstellen, das System regelmäßig zu nutzen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Ich empfinde das System als unnötig komplex.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich empfinde das System als einfach zu nutzen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich denke, dass ich technischen Support brauchen würde, um das System zu nutzen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich finde, dass die verschiedenen Funktionen des Systems gut integriert sind.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich finde, dass es im System zu viele Inkonsistenzen gibt.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich kann mir vorstellen, dass die meisten Leute das System schnell zu beherrschen lernen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ich empfinde die Bedienung als sehr umständlich.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Ich habe mich
bei der Nutzung
des Systems
sehr sicher
gefühlt.

☐☐☐☐☐

Ich musste eine
Menge Dinge
lernen, bevor ich
mit dem System
arbeiten konnte.

☐☐☐☐☐

Was hat Ihnen besonders gut gefallen?

Meine Antwort

Was hat Ihnen nicht gefallen?

Meine Antwort

Note on the Remaining Sections of this User Study

The rest of the user study was omitted, as it includes questions regarding the supporting concepts of the graphical editor, which are not important for this specific evaluation of the thesis.

Appendix D User Study on Teaching Scenarios (Early 2021)

D.1 Questionnaire of the User Study (German)

stARS-Umfrage zur Lehrstrategie von Dozierenden

Sehr geehrte Damen und Herren,

vielen Dank, dass Sie sich die Zeit nehmen, um an dieser Umfrage des Forschungsprojekts stARS (scenario-tailored Audience Response System) teilzunehmen. stARS beschreibt einen Ansatz, der einen an die Lehrstrategie des Dozierenden angepassten Einsatz technischer Werkzeuge in beliebigen Anwendungsszenarien ermöglicht. Ihre Antworten helfen uns, den Ansatz noch weiter auf Dozierende anzupassen.

Die Umfrage dauert etwa 20 – 25 Minuten und findet anonym statt. Sie ist in folgende Teilbereiche gegliedert:

1. Fragen zu Ihrer Lehrerfahrung und Erfahrung mit digitalen Lernumgebungen,
2. Fragen zu Ihrer/n persönlichen Lehrstrategie/n,
3. Bewertung von didaktischen Szenarien, die mittels stARS abgebildet wurden,
4. (optional) Feedback zu stARS.

Die Umfrage können Sie jederzeit mittels Klick auf „Später fortfahren“ zwischenspeichern und zu einem späteren Zeitpunkt fortsetzen.

Bereits im Voraus bedanken wir uns vielmals für Ihre Teilnahme!

In dieser Umfrage sind 40 Fragen enthalten.

Teil 1: Fragen zur Lehrerfahrung sowie Erfahrung mit digitalen Lernumgebungen

Wie lange betreiben Sie bereits Lehre bzw. haben Sie
Lehre betrieben? *

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ ich habe noch nie Lehre betrieben
- ☐ bis zu 1 Jahr
- ☐ zwischen 1 und 3 Jahre
- ☐ zwischen 3 und 5 Jahre
- ☐ zwischen 5 und 10 Jahre
- ☐ über 10 Jahre

Welche Arten von Lehrveranstaltungen haben Sie bereits gehalten? *

❶ Bitte wählen Sie die zutreffenden Antworten aus:

Bitte wählen Sie alle zutreffenden Antworten aus:

- ☐ < 20 Teilnehmer (z.B. Übung, Seminar)
- ☐ 20 - 49 Teilnehmer (z.B. große Übung, kleine Vorlesung)
- ☐ 50 - 150 Teilnehmer (z.B. Vorlesung)
- ☐ > 150 Teilnehmer (z.B. große Vorlesung)
- ☐ ich habe noch keine Lehrveranstaltung gehalten

Ich bin mit dem Konzept von stARS vertraut. *

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Hinweis:

Im Kontext dieser Umfrage definieren wir eine **digitale Lernumgebung** als Unterstützungswerkzeug für Lehrveranstaltungen, um die Interaktion zwischen dem Dozierenden und den Studierenden oder auch zwischen Studierenden selbst zu erhöhen.

Bei der Benutzung digitaler Lernumgebungen fühle ich mich unsicher.

*

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Welche Funktionalität(en) erwarten Sie, wenn Sie Ihre Lehrveranstaltung durch eine digitale Lernumgebung unterstützen? *

❶ Bitte wählen Sie die zutreffenden Antworten aus:

Bitte wählen Sie alle zutreffenden Antworten aus:

- ☐ Bereitstellung von Materialien (Präsentationsfolien, Texte, Bilder, Videos, ...)
- ☐ Wissensabfragen mit unterschiedlichen Feedback-Stufen
- ☐ Allgemeine Abfragen (z.B. der Grund für den Besuch der Lehrveranstaltung)
- ☐ Datei-Uploads durch Studierende
- ☐ Erstellung von Grafiken (durch den Dozierenden oder die Studierenden) mittels Whiteboard
- ☐ Live-Feedback an den Dozierenden (qualitatives oder quantitatives Feedback)
- ☐ Öffentlicher Live-Chat
- ☐ Forumsfunktionalität für studentische Fragen
- ☐ Zuordnung von Studierenden in Gruppen
- ☐ Virtuelle Gruppenaktivitäten
- ☐ Peer Feedback
- ☐ Weitere Funktionen::

Optional: Welche digitalen Lernumgebungen benutzen Sie in Ihrer/n Lehrveranstaltung(en)?

Bitte geben Sie Ihre Antwort hier ein:

Teil 2: Fragen zur persönlichen Lehrstrategie

Das folgende didaktische Szenario ... *

Bitte wählen Sie die zutreffende Antwort für jeden Punkt aus:

	... kenne ich nicht.	... ist mir bekannt, aber benutze ich nicht.	... habe ich schon verwendet	... verwende ich regelmäßig.
Frontalvortrag	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PowerPoint Präsentation	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Studierendenvorträge	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Interaktive Lernaufgaben / Aufgabenbasiertes Lernen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Peer Instruction	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Gruppen-Puzzle	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Think-Pair-Share	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Stationsarbeit	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Offene Diskussion / Classwide Discussion	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
World-Café	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Blitzlicht / Brainstorming	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fallbeispiel / Problem based learning	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Rollenspiel	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

	... kenne ich nicht.	... ist mir bekannt, aber benutze ich nicht.	... habe ich schon verwendet	... verwende ich regelmäßig.
Cognitive Apprenticeship (CA) / Modelllernen	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Verwenden Sie über die oben aufgeführten didaktischen Szenarien noch weitere, so ergänzen Sie diese bitte hier.

Bitte geben Sie Ihre Antwort hier ein:

Die folgenden didaktischen Szenarien würde ich gerne öfter verwenden.

❶ Bitte wählen Sie die zutreffenden Antworten aus:

Bitte wählen Sie alle zutreffenden Antworten aus:

- ☐ Frontalvortrag
- ☐ PowerPoint Präsentation
- ☐ Studierendenvorträge
- ☐ Interaktive Lernaufgaben / Aufgabenbasiertes Lernen
- ☐ Peer Instruction
- ☐ Gruppen-Puzzle
- ☐ Think-Pair-Share
- ☐ Stationsarbeit
- ☐ Offene Diskussion / Classwide Discussion
- ☐ World-Café
- ☐ Blitzlicht / Brainstorming
- ☐ Fallbeispiel / Problem based learning
- ☐ Rollenspiel
- ☐ Cognitive Apprenticeship (CA) / Modelllernen

☐ Weitere::

Hinweis (Wiederholung):

Im Kontext dieser Umfrage definieren wir eine **digitale Lernumgebung** als Unterstützungswerkzeug für Lehrveranstaltungen, um die Interaktion zwischen dem Dozierenden und den Studierenden oder auch zwischen Studierenden selbst zu erhöhen.

Die folgenden didaktischen Szenarien würde ich gerne mittels einer digitalen Lernumgebung unterstützen.

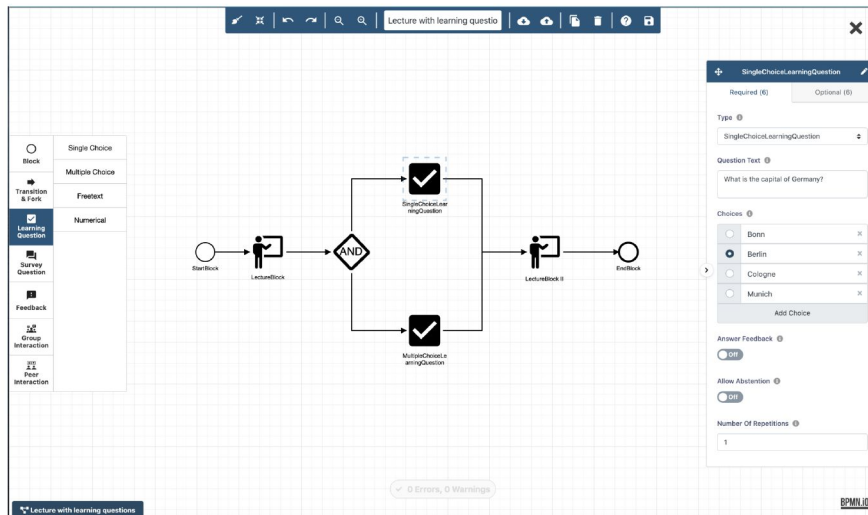
❶ Bitte wählen Sie die zutreffenden Antworten aus:

Bitte wählen Sie alle zutreffenden Antworten aus:

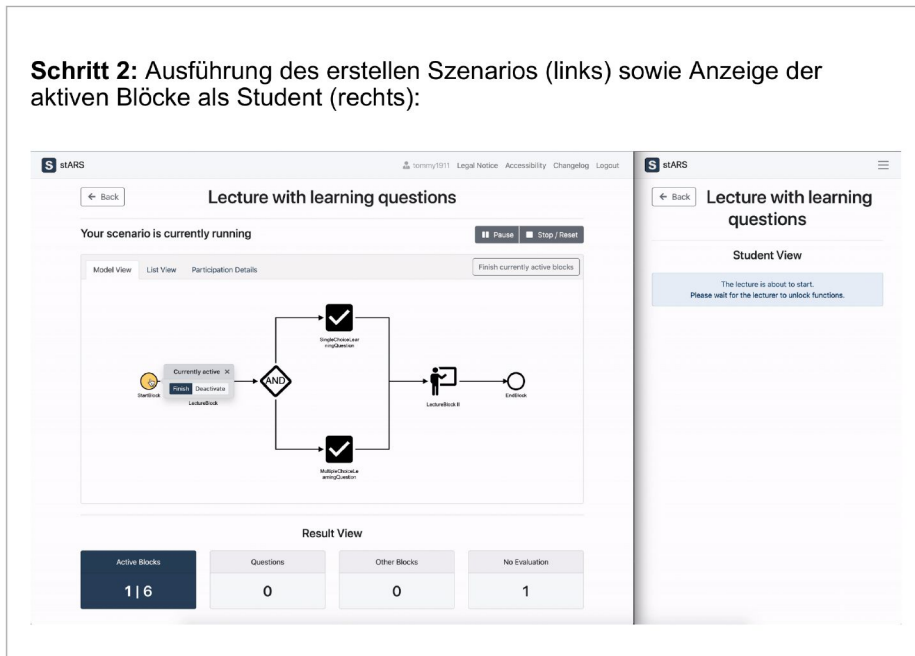
- ☐ Frontalvortrag
- ☐ PowerPoint Präsentation
- ☐ Studierendenvorträge
- ☐ Interaktive Lernaufgaben / Aufgabenbasiertes Lernen
- ☐ Peer Instruction
- ☐ Gruppen-Puzzle
- ☐ Think-Pair-Share
- ☐ Stationsarbeit
- ☐ Offene Diskussion / Classwide Discussion
- ☐ World-Café
- ☐ Blitzlicht / Brainstorming
- ☐ Fallbeispiel / Problem based learning
- ☐ Rollenspiel
- ☐ Cognitive Apprenticeship (CA) / Modelllernen
- ☐ Weitere::

Teil 3: Bewertung unterschiedlicher Szenarien, die in stARS modelliert wurden

Hinweis: Nachfolgend wird Ihnen die Funktionsweise von stARS beispielhaft anhand der Modellierung sowie Ausführung eines einfachen Szenarios demonstriert.

Schritt 1: Erstellung eines individuellen Lehrszenarios mit Hilfe eines grafischen Editors:

Schritt 2: Ausführung des erstellten Szenarios (links) sowie Anzeige der aktiven Blöcke als Student (rechts):

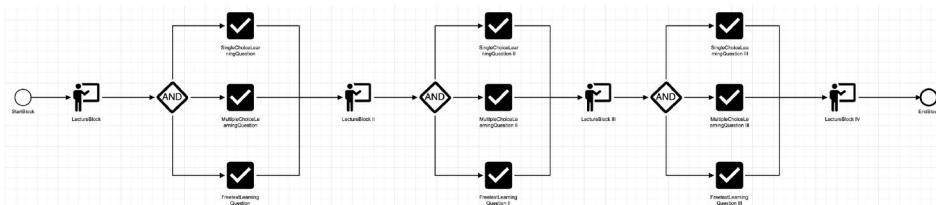


Hinweis zum weiteren Ablauf der Evaluation:

Nachfolgend werden Ihnen einige Lehrszenarien, die mittels stARS modelliert wurden, angezeigt. Ihre Aufgabe wird es sein, diese Modelle auf Verständlichkeit zu überprüfen.

Lehrveranstaltung mit interaktiven Lernfragen

In stARS wurde eine Lehrveranstaltung mit interaktiven Lernfragen wie folgt modelliert (auf die Eingabe von eigenen Namen für Blöcke, die Lösungshinweise geben könnten, wurde gezielt verzichtet):



Modellieren Sie den Ablauf des oben gezeigten Modells mit Hilfe der links angezeigten Veranstaltungsblöcke.

*

❶ Alle Ihre Antworten müssen unterschiedlich sein, und müssen zugeordnet sein.

❷ Bitte wählen Sie maximal 9 Antworten.

Bitte nummerieren Sie jede Box in der Reihenfolge Ihrer Präferenz, beginnen mit 1 bis 9

Start der Lehrveranstaltung

Einleitung in die Lehrveranstaltung sowie Wiederholung der vorherigen Lehrveranstaltung

Wissensabfrage

Vorstellung des ersten Themas

Wissensabfrage

Vorstellung des zweiten Themas

Wissensabfrage

Zusammenfassung der Lehrveranstaltung

Ende der Lehrveranstaltung

Sofern Sie die Aufgabe nicht lösen können, wählen Sie bitte eine beliebige Reihenfolge der Blöcke.

Die Darstellung des Szenarios in stARS empfinde ich als intuitiv. *

❗ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Die Darstellung bezieht sich auf das oben abgebildete Szenario.

Die Ausführung des oben gezeigten Szenarios empfinde ich als umständlich. *

❗ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

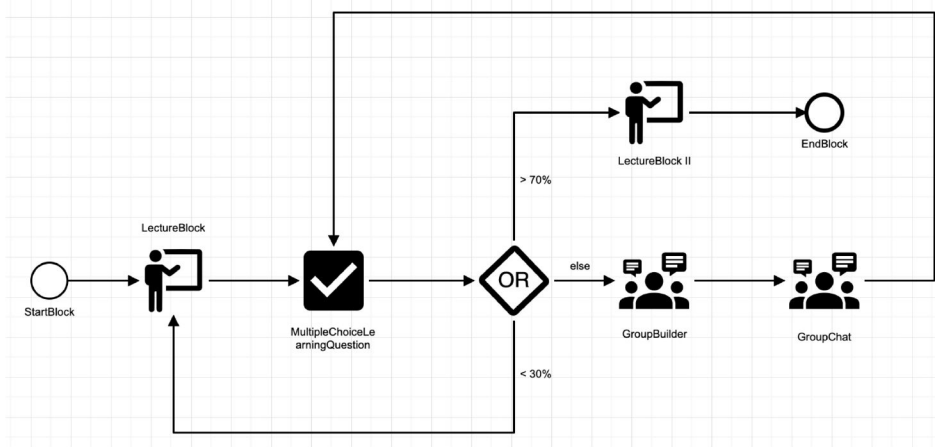
- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

An der aktuellen Visualisierung stört mich ...

Bitte geben Sie Ihre Antwort hier ein:

Lehrveranstaltung mit einer Iteration von Peer Instruction

In stARS wurde eine Iteration von Peer Instruction wie folgt modelliert (auf die Eingabe von eigenen Namen für Blöcke, die Lösungshinweise geben könnten, wurde gezielt verzichtet):



Modellieren Sie den Ablauf des oben gezeigten Modells mit Hilfe der links angezeigten Veranstaltungsblöcke.

Hinweis: In der ersten Iteration beantwortet die Hälfte aller Studierenden die MultipleChoiceLearningQuestion richtig, während die andere Hälfte der Studierenden falsch antwortet. In der zweiten Iteration beantworten nahezu alle Studierenden die Aufgabe richtig.

*

❶ Alle Ihre Antworten müssen unterschiedlich sein, und müssen zugeordnet sein.

❷ Bitte wählen Sie maximal 8 Antworten.

Bitte nummerieren Sie jede Box in der Reihenfolge Ihrer Präferenz, beginnen mit 1 bis 8

Start der Lehrveranstaltung

Impulsvortrag (Brief Lecture)

Wissensabfrage (ConcepTest)

Gruppenbildung von Peers

Diskussion mit den Nachbarn (Peer Discussion)

Wissensabfrage (ConcepTest)

Vertiefte Erklärung zur Zusammenfassung

Ende der Lehrveranstaltung

Sofern Sie die Aufgabe nicht lösen können, wählen Sie bitte eine beliebige Reihenfolge der Blöcke.

Die Darstellung des Szenarios in stARS empfinde ich als intuitiv. *

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Die Darstellung bezieht sich auf das oben abgebildete Szenario.

Die Ausführung des oben gezeigten Szenarios empfinde ich als umständlich. *

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

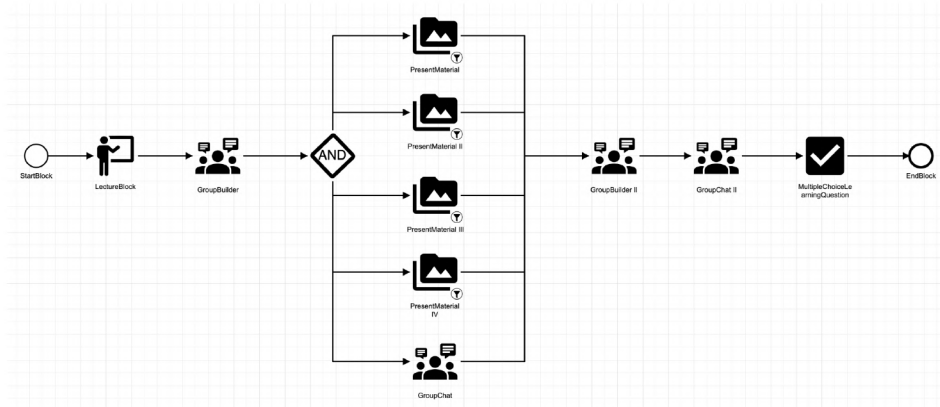
- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

An der aktuellen Visualisierung stört mich ...

Bitte geben Sie Ihre Antwort hier ein:

Lehrveranstaltung, die mittels Gruppen-Puzzle unterstützt wird

In stARS wurde eine Lehrveranstaltung mit Umsetzung eines Gruppen Puzzles wie folgt modelliert (auf die Eingabe von eigenen Namen für Blöcke, die Lösungshinweise geben könnten, wurde gezielt verzichtet):



Modellieren Sie den Ablauf des oben gezeigten Modells mit Hilfe der links angezeigten Veranstaltungsblöcke.

*

❶ Alle Ihre Antworten müssen unterschiedlich sein, und müssen zugeordnet sein.

❷ Bitte wählen Sie maximal 8 Antworten.

Bitte nummerieren Sie jede Box in der Reihenfolge Ihrer Präferenz, beginnen mit 1 bis 8

Start der Lehrveranstaltung

Einleitender Vortrag

Bilden der Expertengruppen

Anzeige von Materialien für die zugewiesene Expertengruppe sowie

Möglichkeit des Austauschs

Bilden der Lerngruppen

Austausch in der Lerngruppe und Diskussion der Teilthemen

Wissensabfrage zur Überprüfung des Verständnisses aller Teilthemen

Ende der Lehrveranstaltung

Sofern Sie die Aufgabe nicht lösen können, wählen Sie bitte eine beliebige Reihenfolge der Blöcke.

Die Darstellung des Szenarios in stARS empfinde ich als intuitiv. *

❗ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Die Darstellung bezieht sich auf das oben abgebildete Szenario.

Die Ausführung des oben gezeigten Szenarios empfinde ich als umständlich. *

❗ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

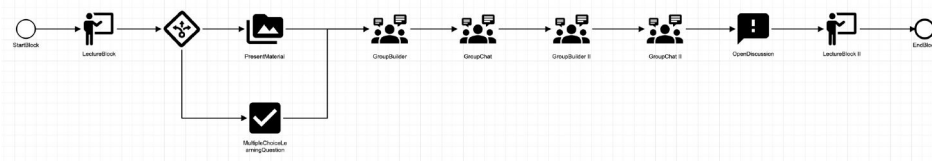
- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

An der aktuellen Visualisierung stört mich ...

Bitte geben Sie Ihre Antwort hier ein:

Lehrveranstaltung, die mittels Think-Pair-Share unterstützt wird

In stARS wurde eine Lehrveranstaltung, in der Think-Pair-Share mit zwei Austauschphasen umgesetzt wird, wie folgt modelliert (auf die Eingabe von eigenen Namen für Blöcke, die Lösungshinweise geben könnten, wurde gezielt verzichtet):



Modellieren Sie den Ablauf des oben gezeigten Modells mit Hilfe der links angezeigten Veranstaltungsblöcke.

❶ Alle Ihre Antworten müssen unterschiedlich sein, und müssen zugeordnet sein.

❷ Bitte wählen Sie maximal 10 Antworten.

Bitte nummerieren Sie jede Box in der Reihenfolge Ihrer Präferenz, beginnen mit 1 bis 10

Start der Lehrveranstaltung

Einleitender Vortrag

Manuelle Auswahl zwischen Fallbeispiel oder komplexer Lernfrage

Bildung von Zweiergruppen

Austausch in Zweiergruppen

Bildung von Vierergruppen

Austausch in Vierergruppen

Offene Diskussion aller Studierenden

Zusammenfassung der Ergebnisse

Ende der Lehrveranstaltung

Sofern Sie die Aufgabe nicht lösen können, wählen Sie bitte eine beliebige Reihenfolge der Blöcke.

Die Darstellung des Szenarios in stARS empfinde ich als intuitiv. *

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Die Darstellung bezieht sich auf das oben abgebildete Szenario.

Die Ausführung des oben gezeigten Szenarios empfinde ich als umständlich. *

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

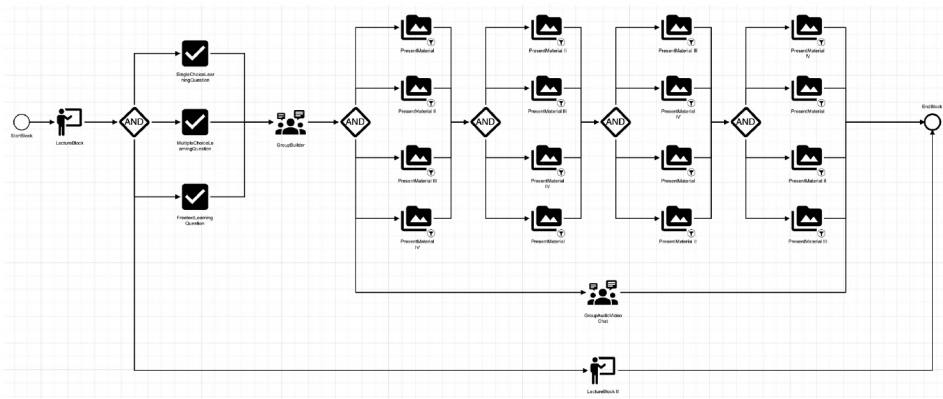
- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

An der aktuellen Visualisierung stört mich ...

Bitte geben Sie Ihre Antwort hier ein:

Lehrveranstaltung mit Stationsarbeit

In stARS wurde eine Lehrveranstaltung mit Stationsarbeit wie folgt modelliert (auf die Eingabe von eigenen Namen für Blöcke, die Lösungshinweise geben könnten, wurde gezielt verzichtet):



Modellieren Sie den Ablauf des oben gezeigten Modells mit Hilfe der links angezeigten Veranstaltungsblöcke.

Hinweis: Versetzen Sie sich bei diesem Ablauf in die Rolle der ersten Gruppe.

*

❗ Alle Ihre Antworten müssen unterschiedlich sein, und müssen zugeordnet sein.

❗ Bitte wählen Sie maximal 9 Antworten.

Bitte nummerieren Sie jede Box in der Reihenfolge Ihrer Präferenz, beginnen mit 1 bis 9

Start der Lehrveranstaltung

Initiale Erklärung des Ablaufs der Lehrveranstaltung

Wissensabfrage mit parallelen Vortragsunterbrechungen

Bildung von Gruppen auf Grundlage des Vorwissens

Anzeige der Materialien für die erste Station mit parallelem Videochat

Anzeige der Materialien für die zweite Station

Anzeige der Materialien für die dritte Station

Anzeige der Materialien für die vierte Station

Ende der Lehrveranstaltung

Sofern Sie die Aufgabe nicht lösen können, wählen Sie bitte eine beliebige Reihenfolge der Blöcke.

Die Darstellung des Szenarios in stARS empfinde ich als intuitiv. *

❗ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Die Darstellung bezieht sich auf das oben abgebildete Szenario.

Die Ausführung des oben gezeigten Szenarios empfinde ich als umständlich. *

❗ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

An der aktuellen Visualisierung stört mich ...

Bitte geben Sie Ihre Antwort hier ein:

Teil 4: Optionales Feedback zu stARS

Das nachträgliche Ändern von didaktischen Szenarien in stARS (z.B. das Hinzufügen einer Frage während der laufenden Lehrveranstaltung) erachte ich als wichtig.

Hinweis: In der aktuellen Implementierung sind die Szenarien nach deren Start statisch und können nicht verändert werden.

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Ich würde stARS gerne in meiner Lehrveranstaltung einsetzen.

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ trifft nicht zu
- ☐ trifft eher nicht zu
- ☐ teils-teils
- ☐ trifft eher zu
- ☐ trifft zu

Sofern Sie Interesse am Einsatz von stARS haben, können Sie sich gerne per Email an tommy.kubica@tu-dresden.de (<mailto:tommy.kubica@tu-dresden.de?subject=stARS-Integration>) wenden oder auch einen Prototypen des Systems selbstständig auf <https://stars-project.com> (<https://stars-project.com>) ausprobieren.



Sofern Sie keine weiteren Updates von stARS verpassen wollen, können Sie uns ebenfalls gerne Ihre Email-Adresse hinterlassen.

Bitte geben Sie Ihre Antwort hier ein:

Hier wäre zudem Platz für weiteres Feedback, dass Sie uns mit auf den Weg geben möchten.

Bitte geben Sie Ihre Antwort hier ein:

Wir bedanken uns vielmals für Ihre Teilnahme an dieser Evaluation und wünschen Ihnen weiterhin einen schönen Tag.

D.2 Detailed Results of the Sorting Questions

Interactive Learning Questions (20 participants)

Table D.1: The results of the sorting question for *Interactive Learning Questions*. As a rating, 5 is used when participants agreed with the intuitiveness and 1 if they do not. Participants without prior knowledge on stARS (i.e., disagree or partly disagree with being familiar with the concept) are marked bold.

Participant	Sorting Task	Intuitive Visualization	Intuitive Execution
P1	full	5	5
P2	incorrect	4	5
P3	full	5	5
P4	partly (1)	5	5
P5	full	5	5
P6	full	5	3
P7	full	4	4
P8	full	4	4
P9	full	5	5
P10	full	2	3
P11	full	5	4
P12	full	5	2
P13	incorrect	4	2
P14	full	4	4
P15	full	4	4
P16	full	4	4
P17	full	5	5
P18	full	4	4
P19	full	5	5
P20	full	5	5

Peer Instruction (19 participants)

Table D.2: The results of the sorting question for *Peer Instruction*. As a rating, 5 is used when participants agreed with the intuitiveness and 1 if they do not. Participants without prior knowledge on stARS (i.e., disagree or partly disagree with being familiar with the concept) are marked bold.

Participant	Sorting Task	Intuitive Visualization	Intuitive Execution
P1	full	5	5
P2	incorrect	4	5
P3	full	5	5
P4	full	4	3
P5	full	5	5
P6	full	4	2
P7	partly (1)	4	4
P8	full	4	5
P9	full	5	5
P10	full	4	4
P11	full	4	4
P12	full	4	4
P13	–	–	–
P14	full	4	4
P15	full	4	4
P16	nearly	4	4
P17	full	5	5
P18	full	4	3
P19	full	5	5
P20	full	5	5

Jigsaw Classroom (19 participants)

Table D.3: The results of the sorting question for *Jigsaw Classroom*. As a rating, 5 is used when participants agreed with the intuitiveness and 1 if they do not. Participants without prior knowledge on stARS (i.e., disagree or partly disagree with being familiar with the concept) are marked bold.

Participant	Sorting Task	Intuitive Visualization	Intuitive Execution
P1	full	5	5
P2	nearly	4	5
P3	full	4	4
P4	full	5	5
P5	nearly	5	5
P6	full	3	4
P7	nearly	5	5
P8	partly (2)	4	4
P9	partly (1)	5	5
P10	full	2	3
P11	full	3	4
P12	full	3	3
P13	–	–	–
P14	full	4	4
P15	partly (2)	4	4
P16	nearly	4	4
P17	full	5	5
P18	nearly	3	3
P19	nearly	5	5
P20	full	5	5

Think-Pair-Share (19 participants)

Table D.4: The results of the sorting question for *Think-Pair-Share*. As a rating, 5 is used when participants agreed with the intuitiveness and 1 if they do not. Participants without prior knowledge on stARS (i.e., disagree or partly disagree with being familiar with the concept) are marked bold.

Participant	Sorting Task	Intuitive Visualization	Intuitive Execution
P1	full	5	5
P2	partly (3)	4	5
P3	partly (2)	3	3
P4	full	5	5
P5	full	4	5
P6	full	2	2
P7	full	5	5
P8	full	4	5
P9	full	4	5
P10	full	2	3
P11	partly (2)	4	4
P12	full	3	3
P13	–	–	–
P14	partly (2)	3	2
P15	full	3	4
P16	partly (3)	3	3
P17	full	5	5
P18	full	2	3
P19	full	4	4
P20	full	5	4

Learning Stations (19 participants)

Table D.5: The results of the sorting question for *Learning Stations*. As a rating, 5 is used when participants agreed with the intuitiveness and 1 if they do not. Participants without prior knowledge on stARS (i.e., disagree or partly disagree with being familiar with the concept) are marked bold.

Participant	Sorting Task	Intuitive Visualization	Intuitive Execution
P1	full	5	5
P2	partly (1)	4	5
P3	full	2	2
P4	partly (1)	5	5
P5	full	5	5
P6	full	3	2
P7	full	3	3
P8	full	5	4
P9	full	4	5
P10	full	2	2
P11	full	3	4
P12	nearly	2	3
P13	–	–	–
P14	full	2	3
P15	full	2	2
P16	full	3	3
P17	full	5	5
P18	full	3	3
P19	full	4	4
P20	nearly	4	4

Appendix E User Study on Runtime Adaptation (Mid 2021)

E.1 Questionnaire of the User Study (German)

Evaluation eines Prototyps zur Erweiterung laufender Szenarien in stARS

Sehr geehrte Damen und Herren,

vielen Dank, dass Sie sich die Zeit nehmen, um an dieser Umfrage des Forschungsprojekts stARS teilzunehmen. Diese nimmt etwa 30 Minuten in Anspruch. Ziel ist die Benutzung eines Prototyps, der es erlaubt, in stARS ausgeführte Szenarien während der Laufzeit zu erweitern, um beispielsweise auf Echtzeit-Ergebnisse reagieren zu können.

Hierzu sollen Sie nachfolgend vier kurze Aufgaben lösen und jeweils Bewertungen vornehmen. Weiterhin haben Sie die Möglichkeit, offenes Feedback zu geben.

Um einen Einblick in die kognitiven Prozesse während der Bearbeitung der Aufgaben zu erhalten, bitte ich Sie parallel zur Ausführung laut mitzudenken. Sprechen Sie hierbei all das aus, was Sie tun und denken, worauf Sie schauen und was Sie während der Bearbeitung der Aufgaben fühlen. Bitte denken Sie zudem daran, dass es bei der Bearbeitung der Aufgaben keine richtigen oder falschen Verhaltensweisen oder Antworten gibt. Im Fokus der Umfrage steht vielmehr Ihre individuelle Einschätzung.

Sofern Unklarheiten während der Bearbeitung der Aufgaben auftreten, können Sie jederzeit nachfragen.

Bereits im Voraus bedanke ich mich vielmals für Ihre Teilnahme!

Teil 1: Fragen zu Ihrer Erfahrung mit stARS

Wie vertraut sind Sie mit dem Konzept von stARS? *

1 2 3 4 5

Das Konzept ist mir unbekannt

☐ ☐ ☐ ☐ ☐

Das Konzept ist mir vertraut

Haben Sie in einer vorherigen Evaluation oder einem vorherigen Einsatz die Funktionalität zum Laden bzw. Importieren existierender Szenarien benutzt? *

- ☐ Ja
- ☐ Nein
- ☐ Ich bin mir unsicher

Haben Sie bereits schon einmal ein Szenario in stARS ausgeführt? *

- ☐ Ja
- ☐ Nein
- ☐ Ich bin mir unsicher

Die Erweiterung von laufenden Szenarien in stARS empfinde ich als relevant. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

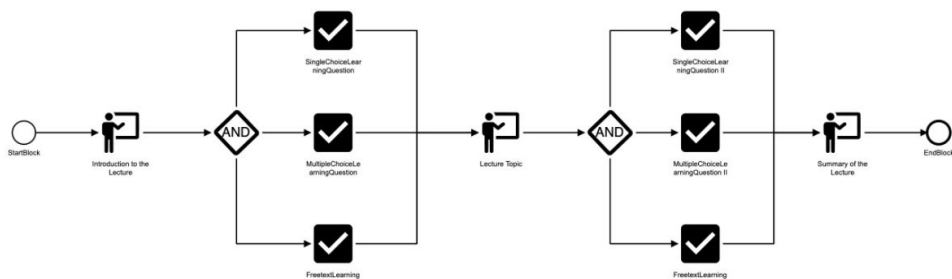
(Optional) Was erwarten Sie von einer Funktion zur Erweiterung laufender Lehrszenarien? Welche Herausforderungen sehen Sie bei einer solchen Funktionalität?

Meine Antwort _____

Teil 2: Vorbereitung des Prototyps

Bitte teilen Sie Ihren Bildschirm und navigieren auf <https://stars-project.com/>. Loggen Sie sich anschließend mit den nachfolgenden Credentials im System ein: pseudonym "evaluationtommy" - passwort "passwort".

Öffnen Sie den Szenarien-Editor, laden das private Template "Interactive Learning Questions" und erstellen dies als neues Szenario. Navigieren Sie anschließend in dieses erstellte Szenario und starten Sie es.



Teil 3: Aufgabe 1

Versetzen Sie sich in die Lage einer Lehrperson, die das modellierte Szenario ausführt. Nachdem Sie die "Introduction" Ihrer Lehrveranstaltung gehalten haben, wollen Sie spontan eine Umfrage durchführen, die im vorhandenen Szenario nicht hinterlegt ist: Konkret sollen die Studierenden eines von drei Themen auswählen, welches im Verlauf der Lehrveranstaltung genauer betrachtet wird.

Versuchen Sie nun, das Szenario ausgehend vom "Introduction to the Lecture"-Block um diese Frage zu ergänzen. Verwenden Sie hierfür lediglich die Funktion zur Erweiterung eines laufenden Szenarios.

Navigieren Sie anschließend auf die erstellte Frage und stellen Sie sicher, dass diese korrekt ausgeführt wird.

Ich hatte Schwierigkeiten beim Auffinden der Funktion zur Erweiterung des Szenarios. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Das Lösen der Aufgabe habe ich als einfach empfunden. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Ich bin mit dem Ergebnis der Aufgabe zufrieden. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

(Optional) Weiteres Feedback zur Aufgabe

Meine Antwort

Teil 4: Aufgabe 2

Nach Beendigung der Umfrage navigieren Sie zum ersten Fragenkomplex, wodurch drei Lernfragen aktiv werden. Da das Thema der "SingleChoiceLearningQuestion" anschließend vertieft in der Lehrveranstaltung aufgegriffen werden soll, wollen Sie den Studierenden die Ergebnisse dieser Aufgabe auf ihren Geräten präsentieren. Fügen Sie hierzu einen "PresentResult"-Block an einer geeigneten Stelle ein und referenzieren Sie diesen auf die "SingleChoiceLearningQuestion". Stellen Sie nach der Erweiterung Ihres Szenarios sicher, dass der Block korrekt ausgeführt wird.

Die Referenzierung von neu hinzugefügten Blöcken auf Blöcke des vorhandenen Szenarios verwirrt mich. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Das Lösen der Aufgabe habe ich als einfach empfunden. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Ich bin mit dem Ergebnis der Aufgabe zufrieden. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

(Optional) Weiteres Feedback zur Aufgabe

Meine Antwort

Teil 5: Aufgabe 3

Navigieren Sie anschließend zum zweiten Fragenkomplex. Ziel der dritten Aufgabe ist es, das private Template "Group Interaction" zur Diskussion der "FreetextLearningQuestion II" hinzuzufügen. Finden Sie hierfür einen geeigneten Ort, überprüfen Sie die Parameter und erweitern das Szenario. Bitte bleiben Sie im aktuellen Zustand des Modells und schalten keinen der Blöcke weiter.

Die Erweiterung von Szenarien um private oder öffentliche Templates erachte ich als sinnvoll. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Das Lösen der Aufgabe habe ich als einfach empfunden. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Ich bin mit dem Ergebnis der Aufgabe zufrieden. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

(Optional) Weiteres Feedback zur Aufgabe

Meine Antwort

Teil 6: Aufgabe 4

Nach der Erweiterung des Szenarios fällt Ihnen auf, dass Sie vergessen haben den "GroupBuilder" an Ihr aktuelles Szenario anzupassen. Ziel dieser Aufgabe wird es sein, die letzte vorgenommene Erweiterung rückgängig zu machen und diese erneut mit dem Parameter "GroupSize = 5" für den "GroupBuilder" durchzuführen.

Das Rückgängigmachen der letzten Erweiterung erachte ich als sinnvoll. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Das Lösen der Aufgabe habe ich als einfach empfunden. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Ich bin mit dem Ergebnis der Aufgabe zufrieden. *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

(Optional) Weiteres Feedback zur Aufgabe

Meine Antwort

Zusammenfassende Fragen

Die Erweiterung des Szenarios um Blöcke und Teilszenarien empfinde ich für die Anpassung zur Laufzeit als ausreichend (eine freie Anpassung des Szenarios ist nicht notwendig). *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Ich würde mir wünschen, dass auf Grundlage von Echtzeit-Ergebnissen automatisch sinnvolle Erweiterungen vorgeschlagen werden (z.B. könnte eine Gruppendiskussion vorgeschlagen werden, wenn ein Großteil der Studierenden falsch geantwortet hat). *

- ☐ Trifft zu
- ☐ Trifft eher zu
- ☐ Weder noch
- ☐ Trifft eher nicht zu
- ☐ Trifft nicht zu

Platz für weitere Anmerkungen und Feedback

Meine Antwort

Vielen Dank für Ihre Teilnahme!

Bitte vergessen Sie nicht die Umfrage abzusenden.

Appendix F Questionnaire for the Lecture Experiments (German)

Nachbefragung zum stARS-Realtest

Ich möchte mich oftmals dafür bedanken, dass Sie stARS in Ihrer Lehrveranstaltung eingesetzt haben. Nachfolgend werden Sie gebeten, diesen Realtest zu beschreiben sowie anhand verschiedener Fragen zu bewerten. Zudem wird Ihnen die Möglichkeit gegeben, weiteres Feedback zu geben.

Bereits im Voraus bedanke ich mich für Ihre Teilnahme.

In dieser Umfrage sind 22 Fragen enthalten.

Einsatz-Szenario sowie Zielgruppe

Bitte beschreiben Sie kurz die Lehrveranstaltung, in der Sie stARS eingesetzt haben (Art und Länge der Lehrveranstaltung, Thematik, Anzahl der Studierenden). An welche Zielgruppe ist diese gerichtet? Haben die Studierenden einen technischen Hintergrund? *

Bitte geben Sie Ihre Antwort hier ein:

Bitte beschreiben Sie zudem kurz das Szenario, welches in stARS erstellt wurde. Welchen Grundgedanken (welchen angedachten Mehrwert) haben Sie bei der Erstellung verfolgt? (ggf. Aufschlüsselung nach der Art der Funktionalitäten) *

Bitte geben Sie Ihre Antwort hier ein:

Wie viel Zeit haben Sie in etwa für die Vorbereitung der Inhalte in stARS benötigt? *

Bitte geben Sie Ihre Antwort hier ein:

Traten bei der Erstellung / Bearbeitung des Szenarios technische Schwierigkeiten auf? Sofern ja, welche? *

Bitte geben Sie Ihre Antwort hier ein:

Wie vertraut sind Sie mit der Benutzung ähnlicher Systeme, die es Studierenden erlauben aktiv an der Lehrveranstaltung teilzuhaben? (z.B. AMCS, ARSnova, TUD.vote, ...) *

● Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ Nicht vertraut
- ☐ Eher nicht vertraut
- ☐ Weder noch
- ☐ Eher vertraut
- ☐ Vertraut

Fragen zum Real-Einsatz

Haben Sie in der gleichen Lehrveranstaltung zuvor bereits schon einmal ein ähnliches Tool (z.B. für Umfragen) eingesetzt? Sofern ja, welche/s? *

Bitte geben Sie Ihre Antwort hier ein:

Wie viele der anwesenden Studierenden haben am Real-Einsatz teilgenommen? (ungefähre Prozentanzahl) *

Bitte geben Sie Ihre Antwort hier ein:

Bitte beschreiben Sie kurz wie Sie das System benutzt haben (Ankündigung im Voraus, Ankündigung in der Lehrveranstaltung selbst, Entscheidung wann jeweils mit der Auswertung gegonnen wurde, Vorgehen bei der Auswertung). *

Bitte geben Sie Ihre Antwort hier ein:

Wie viel Zeit hat der stARS-Einsatz in etwa in Ihrer Lehrveranstaltung beansprucht? *

Bitte geben Sie Ihre Antwort hier ein:

Gab es einen Zeitpunkt während der Ausführung, wo Sie gerne "live" etwas angepasst oder zu Ihrem Szenario hinzugefügt hätten (z.B. eine weitere Fragestellung)? Sofern ja, beschreiben Sie diese Situation bitte kurz. *

Bitte geben Sie Ihre Antwort hier ein:

Traten bei der Ausführung des Szenarios technische Schwierigkeiten auf Lehrendenseite auf? Sofern ja, welche? *

Bitte geben Sie Ihre Antwort hier ein:

Traten bei der Ausführung des Szenarios technische Schwierigkeiten auf Studierendenseite auf? Sofern ja, welche? *

Bitte geben Sie Ihre Antwort hier ein:

Gab es sonstige Schwierigkeiten, die aufgetreten sind (z.B. Verständnisprobleme oder Probleme mit der Zeitplanung)? Sofern ja, welche? *

Bitte geben Sie Ihre Antwort hier ein:

Würden Sie nachblickend betrachtet etwas an Ihrem erstellten Szenario ändern? Sofern ja, was würden Sie ändern? *

Bitte geben Sie Ihre Antwort hier ein:

Wünschen Sie sich weiterführende Funktionen, die durch stARS nicht umgesetzt werden konnten? Sofern ja, welche? *

Bitte geben Sie Ihre Antwort hier ein:

Auswertung des Real-Einsatzes

Wie zufrieden sind Sie mit der Benutzung von stARS? *

❶ Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ Nicht zufrieden
- ☐ Eher nicht zufrieden
- ☐ Weder zufrieden noch unzufrieden
- ☐ Eher zufrieden
- ☐ Zufrieden

Haben die verwendeten Funktionen Ihrer Meinung nach den gewünschten Mehrwert erzielt? (ggf. Aufschlüsselung nach der Art der Funktionalitäten) *

Bitte geben Sie Ihre Antwort hier ein:

Ist Ihnen etwas besonders Positives bei der Erstellung oder Ausführung des Funktionsumfangs in stARS aufgefallen? Sofern ja, bitte beschreiben Sie dies kurz. *

Bitte geben Sie Ihre Antwort hier ein:

Wie wahrscheinlich ist es, dass Sie stARS erneut einsetzen? *

● Bitte wählen Sie eine der folgenden Antworten:

Bitte wählen Sie nur eine der folgenden Antworten aus:

- ☐ Sehr unwahrscheinlich
- ☐ Eher unwahrscheinlich
- ☐ Unklar
- ☐ Eher wahrscheinlich
- ☐ Sehr wahrscheinlich

Welche Anpassungen würden Sie in zukünftigen Lehrveranstaltungen vornehmen? (bezogen auf das Szenario selbst sowie die Art des Einsatzes) *

Bitte geben Sie Ihre Antwort hier ein:

Platz für weiteres Feedback sowie Vorschläge jeglicher Art.

Bitte geben Sie Ihre Antwort hier ein:

A large, empty rectangular box with a thin black border, intended for providing feedback or suggestions.

Vielen Dank für Ihre Mühe und Zeit zur Beantwortung dieser Umfrage!
Bitte vergessen Sie nicht diese abzusenden.

Übermittlung Ihres ausgefüllten Fragebogens:
Vielen Dank für die Beantwortung des Fragebogens.

Acknowledgments

During the creation of this thesis, I was supported by many persons to whom I would like to express my heartfelt gratitude. Without your continuous support, the success of this thesis would not have been possible.

First, I would like to thank Prof. Dr. Alexander Schill for providing me the opportunity to work within the Research Training Group RoSI, as well as the freedom and trust he gave me in my research. I also want to thank him for his constant support and constructive feedback throughout my thesis. His ability to offer solutions for differently complex problems in “real-time” has not only always impressed me but has also changed my way of thinking when solving a problem.

In this context, I would also like to thank several other persons who have supervised me in different ways. First and foremost, I would like to mention Prof. Dr. Tenshi Hara and Dr. Iris Braun, who have supported me through (nearly) my entire studies, and also encouraged me to write this thesis, whose possibility I had never imagined five years ago. I would also like to thank Dr. Thomas Springer and Dr. Markus Wutzler for their guidance in the weekly stand-up meetings, as well as Dr. Felix Kapp and Dr. Gregor Damnik for supporting me in the vastness of psychology. Moreover, I want to thank Dr. Marius Feldmann, who, despite an approximately 80-hour week, has managed to use stARS twice in his lectures.

Furthermore, I would also like to thank Prof. Dr. Thorsten Strufe for acting as the advisor of my thesis, and Univ.-Prof. DI Dr. Andreas Bollin, who supported my thesis as the external reviewer and provided me the opportunity to present in front of his group. In this context, I also want to thank Prof. Dr. Nadine Bergner for the possibility of presenting stARS to her group.

Additionally, I want to thank all colleagues and students that have supported me differently. In this context, I want to specifically mention Ilja Shmelkin, Robert Peine, and Lidia Roszko, who only made it possible to realize stARS to the extent and quality that it is today. My deep gratitude goes to Robert Peine, who did not only labor-intensive

work on the prototype but always impressed me with the way of solving problems, already looking five steps into the future and adjusting the solution accordingly. I also want to thank all colleagues of the Chair of Computer Networks as well as the Research Training Group RoSI, who always provided me with help if necessary. Particular thanks to José Irigon de Irigon and Jannis Lilienthal.

A huge thanks also applies to the numerous participants of the evaluations that provided feedback from different directions. Without your support, it would not have been possible to realize this entire project.

Last but not least, I would like to express my deepest gratitude to my father Gerd, my grandparents Renate and Günter, my brother Danny, as well as my lifelong friend Rico, who always supported me and believed in whatever I did, even if I did not myself. At this point, I would also like to thank my partner Nathalie for creating an environment, in which I always felt comfortable when writing my thesis.

Thank you.

Tommy Kubica
Dresden, December 2021

“I look above, and I know I’ll always be blessed with love.”
– Robert Peter “Robbie” Williams