Tommy Kubica, Ilja Shmelkin, Alexander Schill

Towards a Development Methodology for Adaptable Collaborative Audience Response Systems

Erstveröffentlichung in / First published in:
DOI: https://doi.org/10.1109/ITHET46829.2019.8937354

Diese Version ist verfügbar / This version is available on:
https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-791715

© 20XX IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Towards a Development Methodology for Adaptable Collaborative Audience Response Systems

Tommy Kubica  
Department of Computer Science  
Technische Universität Dresden  
Dresden, Germany  
tommy.kubica@tu-dresden.de

Ilja Shmelkin  
Department of Computer Science  
Technische Universität Dresden  
Dresden, Germany  
ilja.shmelkin@tu-dresden.de

Alexander Schill  
Department of Computer Science  
Technische Universität Dresden  
Dresden, Germany  
alexander.schill@tu-dresden.de

Abstract—The use of Audience Response Systems (ARSs) for tech-enhanced learning scenarios has proven to address issues occurring within higher education, e.g. the missing interaction between the lecturer and the students. Since the majority of these systems relies on a single supported didactic concept and therefore has a limited set of provided functions, ARSs are currently restricted to support classic content-based as well as enquiry-based learning. The support of more advanced didactic concepts in order to investigate studio-based learning is currently not possible due to the lacking collaborative and cooperative functionality. This paper presents a unified (meta-)model which is able to express various scenarios, targeting the holistic support of content-based, enquiry-based and studio-based learning. The created model is evaluated within a user study to reason about the applicability of its underlying concept as well as the defined function blocks. In addition, this paper purposes ideas for a future graphical editor, which will support the modeling process, and provides concrete details for a possible implementation of a system on top of the (meta-)model.

Index Terms—audience response systems, collaborative learning, adaptability, meta-model, domain-specific language, higher education, technology-enhanced learning

I. INTRODUCTION

Higher education still relies on traditional teaching strategies: The lecturer stands in front of the students and gives a presentation while the students act as passive listeners. This results in missing interaction and leads to well-known problems: The students do not feel integrated and are unsure if they understood a certain topic. Furthermore, the lecturer cannot assess the students’ progression to align upcoming lectures accordingly.

In order to solve these issues, audience response systems (ARSs) provide a promising opportunity: Using their own mobile devices, which is called “bring your own device” (BYOD), students get the possibility to be actively involved during lecture [1], [2]. Depending on the chosen system, they can answer questions of different types or even ask their own questions which are rated by other students according to their importance. Using such systems can support classic content-based learning, in which the lecturer chooses both the questions and the answers the students should know, as well as enquiry-based learning, in which the questions are given by the lecturer and the students have to create the answers by themselves. Studio-based learning builds up on those and defines an approach in which students have to discover the questions for themselves and also find the correct answers to those questions. This should enable students to articulate their knowledge and skills into capability. There are plenty of observations showing the benefits of this approach [3]–[5] since it promotes several human-centric aspects of learning, e.g. collaboration, mentoring and peer-learning [6].

While different ARSs exist, which target to support content-based or enquiry-based learning, or even the combination of both, there is, for the best of our knowledge, currently no system targeting the explicit support of studio-based learning or even further a holistic support of content-based, enquiry-based and studio-based learning. This is caused by the problem that such systems often rely on a single didactic concept, e.g. peer instruction or self-regulated learning, and their functional scope and settings are tailored accordingly. Especially collaborative and cooperative functionality, which are an integral part of the studio-based learning approach, are currently rarely supported within classroom scenarios and often limited to school settings or offline scenarios.

In order to provide a solution, this paper presents a unified (meta-)model for supporting various scenarios occurring within ARSs. In addition to classic scenarios targeting content-based and enquiry-based learning approaches, further collaborative and cooperative scenarios will be supported to investigate studio-based learning in these environments.

The remainder of this paper is structured as follows. In section 2, we give background information on the technical aspect and on the didactic aspect of ARSs. In section 3, we present related work. In section 4, we introduce our concept before we describe the implementation of it in section 5. In section 6, we discuss the results of our evaluation and finally conclude this paper in section 7.

II. BACKGROUND

Within this section, the reader will gain a common understanding of audience response systems (ARSs) and their basic functional scope on one side and didactic concepts on the other.
A. Audience Response Systems

“Audience response systems (ARSs) or clickers, as they are commonly called, offer a management tool for engaging students in the large classroom.” [7] While prior hardware-based systems only allowed for answering single choice questions by pressing the specific button on the device (e.g. A, B, C or D), modern web-based ARSs provide advanced functionalities. Systems, such as Polleverywhere1, Socrative2 or GoSoapBox3, allow for answering questions of different types (e.g. single choice, multiple choice or short answer questions) or even further to ask own questions of interest. A comprehensive overview of systems can be retrieved from [8].

[9] summarizes the functional scope of ARSs and distinguishes them in front channel and back channel systems, whereby the former provide functionality that is visible for every student and require an active break during the lecture in order to get executed and presented properly, and the latter functionality that can run in the background of the lecture and does not require an active break or presentation. Furthermore, they distinguish between qualitative and quantitative systems for each group. The classification is depicted in Figure 1.

<table>
<thead>
<tr>
<th>Audience Response Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Front Channel</td>
</tr>
<tr>
<td>Qualitative Systems</td>
</tr>
<tr>
<td>Digital Back Channel</td>
</tr>
<tr>
<td>Quantitative Systems</td>
</tr>
<tr>
<td>Qualitative Systems</td>
</tr>
<tr>
<td>Quantitative Systems</td>
</tr>
</tbody>
</table>

Fig. 1. Classification of audience response systems (ARSs) that distinguishes between front- and back channel systems by [9], translated from German.

1) Qualitative Front Channel: Qualitative front channel systems allow for answering open-ended questions, e.g. short answer questions or questions that require a graphical answer. Examples are evaluation questions that ask for the students’ opinions or drawing tasks to create diagrams or sequences.

2) Quantitative Front Channel: Quantitative front channel systems are used to ask students to answer closed-ended questions, i.e. questions with predefined answers. Examples are multiple choice questions allowing students to check their gained knowledge during lecture or questions regarding the prior knowledge of those students.

3) Qualitative Back Channel: Qualitative back channel systems allow for feedback on closed-ended dimensions. An example is called instant feedback and gives students the ability to rate the speed or volume of the lecturer using buttons represented by icons.

Currently, there exist a lot of ARSs combining functionality of multiple groups. Thus, it is not always possible to assign one system to one group. Moreover, it is often not even possible to assign one functionality to one single group (e.g. closed-ended questions asking for an additional textual answer when choosing “another answer” option).

B. Didactic Concepts

ARSs are able to support specific didactic concepts occurring in different learning styles; e.g., there exist systems providing support of self-regulated learning or peer instruction, or even combinations of them. To establish a common understanding of the didactic concepts mentioned later, they will be introduced in the following. It is only a limited selection of concepts occurring in content-based, enquiry-based and studio-based learning. A lot more didactic concepts and terminologies can be found in related literature, e.g. [10]–[12].

1) Self-Regulated Learning: Self-Regulated Learning (SRL) is a domain occurring in self-regulation. Although there are various definitions, three components seem to be especially important for classroom performance. First, SRL includes students’ meta-cognition, the thinking about one’s thinking. Second, the students’ strategic actions, which include planning, monitoring, and evaluating of the personal progress. Third, the motivation to learn is included in SRL. [13], [14] In order to support the students in their SRL, the teacher has to limit the subject area to relevant topics, so that the students do not get lost in the vastness of potentially extensive topics. [15] In ARSs, this is possible by providing questions of interest that can be executed multiple times. It is important to note that other didactic concepts, such as peer instruction, use components and principles of SRL – they build on top of SRL.

2) Peer Instruction: Peer Instruction (PI) is a student-centered approach to teaching that has proven itself to be beneficial within higher education. [16] The concept consists of different stages that are executed depending on the students’ group performance. In a first step, a brief lecture of 10 to 15 minutes is held to briefly introduce a topic. In the second step, a ConcepTest is executed (often supported by an ARS), which assesses the knowledge of the students about the previously explained topic. The progression of the lecture is determined depending on the results of the ConcepTest.

A small percentage of correct answers will result in the repetition of the brief lecture, which is to be held in another version or with focus on further information regarding certain aspects. Afterwards, the ConcepTest is repeated.

A medium percentage of correct answers results in a peer discussion. Students normally talk to their neighbor and try to convince each other of their respective understanding of the topic. That way, students have to repeat the topic and make
use of suitable arguments to explain their point of view. In return, this may reveal misunderstandings, misconceptions or plain knowledge gaps. Afterwards, the ConceptTest is repeated.

A large percentage of correct answers is needed to end the process with a short explanation and/or summary, and the introduction of the next topic. PI can be repeated multiple times within a single lecture and each procedure of PI can have unlimited stages, depending on the students’ performance(s).

3) Collaborative / Cooperative Learning: Collaborative and Cooperative Learning are student-centered approaches that describe classroom techniques, in which students work in small groups on learning activities and receive rewards and/or recognition based on the group’s performance. While in Cooperative Learning the students divide the task into several sub-tasks and merge their results in a final output, they work together on the task in Collaborative Learning. Both approaches enable their usage in lectures with the help of didactic concepts, however there are multiple documented ways of how to incorporate them. [17], [18]

III. RELATED WORK

The value of ARSs in large lectures has already been investigated a lot [1], [19], [20] – especially by supporting content-based learning using quantitative front channel functionality. Current research tends to investigate ARSs in different types of courses [15], which require different learning styles, e.g. qualitative front channel functionality to support enquiry-based learning. In order to support various scenarios and learning styles, the functional scope needs to be adjusted. This is normally done by simply selecting the range of functions, e.g. in GoSoapBox 4 or Tweedback 5. ARSnova 6 supports this selection by providing predefined scenarios, e.g. peer instruction or simple polls from the audience. [21] focuses on establishing a connection between current scenario and functional scope and investigated a guided selection that supports the lecturer in selecting an appropriate range of functions. Depending on parameters of interest, e.g. an approximate number of students, a suggestion is provided. The developed prototype lacks at its predefined settings, e.g. after an answer to a single-best choice question has been given, a feedback whether it is correct or not is displayed, which is not appropriate in every scenario. [22] investigated the connection between the functional scope of ARSs and didactic concepts and found that, in addition to the functional scope, specific settings are necessary to support a didactic concept.

[23] introduces a generic model that allows for defining individual and configurable scenarios. It is based on a set of objects with attributes and rules with conditions and actions that can be combined to a scenario. In MobileQuizz 7, the model was implemented and evaluated: The modeling turned out to be very complex and difficult to understand and more complicated scenarios produced deep-nested objects, which led to an overall low performance. In addition, highly collaborative or cooperative functions that are an integral part of some learning styles, e.g. studio-based learning, cannot be expressed.

In general, collaborative and cooperative functions within ARSs are currently only investigated as qualitative back channel functionality: They are running in the background and do not require an active break during the lecture, however they are often not addressed in more detail at all. For example, Tweedback or Pigeonhole Live 8 allow students for anonymous, open questions that can be rated and, depending on the chosen setting, answered by other students. [24] investigates the qualitative back channel functionality in more detail and proposes an approach which is provided in Backstage 9, that allows for asking open questions at any specific position of the slides. This enables a connection between the cause of the problem and the question, which could be necessary to answer it. Similar to the aforementioned approaches, these questions can be rated and discussed with other students. Using text markings, the discussions can be executed anonymously or continued by sending private messages to specific students. Nevertheless, the generation of groups and associated discussions, which are described by [22] as essential requirement to support cooperative learning within ARSs, is currently only investigated in school scenarios, e.g. ClassDojo 10 or ONCOO 11 which are limited in their number of participants, or offline scenarios, e.g. the generation of learning groups based on the data received by Learning Management Systems (LMSs) [25]. An investigation of using collaborative and cooperative scenarios within environments providing an anonymous participation during the ongoing lecture, namely ARSs, was not conducted yet.

In the current research, two directions can be recognized. On one side, there exist systems with predefined functional scopes and settings in order to support predefined didactic concepts and on the other side, there exist systems that are based on a generic approach and allow for supporting various scenarios. While the first group of systems lacks at supporting customized scenarios and concepts, the second group often lacks in its generics, which results in low comprehensibility or deep nesting. This current state of technology motivates us to create a unified (meta-)model that uses the best of both groups of systems in order to support various scenarios occurring in ARSs without lack of comprehensibility or performance issues. In addition to the support of content-based and enquiry-based learning, the system targets support of studio-based learning, e.g., by allowing for group formations with associated discussions and voting.

IV. CONCEPT

Within this section, the concept of a unified (meta-)model for scenarios occurring in ARSs is developed. First, the requirements will be defined based on the results of section III.
Next, approaches for creating a unified model will be introduced and checked if they meet the requirements. Requirements that cannot be checked immediately are evaluated within a user study. An approach will be chosen that is used as foundation of the solution. Finally, the functions of the (meta-)model are explained.

A. Requirement Definition

[23] has shown that using a generic model is in some point a suitable solution but will not be the best solution for our approach since it lacks at comprehensibility as well as a bad performance due to deep nesting. In the following, we will summarize existing evaluation results and define requirements for our solution from functional- and from non-functional side.

1) Functional Requirements: These requirements describe concrete functional aspects of our provided solution.

a) Real-Time Capability: During a lecture, decisions have to be made, which create the necessity for adaptations of the created application model. Examples are the addition of new questions or the adjustment of existing questions. This has to be possible with a justifiable number of modeling steps and especially without recreation of the whole application model.

b) Parameterizability: [22] has shown that different didactic concepts require different functional settings, e.g., whether correct answers are shown or hidden after answering. The model has to be able to express these different settings, e.g., by using parameters that adapt the system’s behavior.

2) Non-Functional Requirements: These requirements describe behavior of the model apart from concrete functional aspects, mainly focused on organizational and usability aspects.

a) Simplicity: The investigations of [23] have shown that simplicity has to be one of the main requirements when creating a unified model that is used by a variety of lecturers with different skills in computer science. The model in its plain form has to be understood easily – a scenario editor should only assist the user in creating valid scenarios.

b) Validatability: User models generated from the (meta-)model have to be validatable automatically. This allows for a seamless integration of application models within the execution environment. A scenario editor aids modeling for valid scenarios.

c) Expandability: Since the functional scope of ARSs is not fixed, an extension of the model by further functionality has to be possible.

B. Investigation of Approaches

In the following, approaches that could be suitable to fulfill the defined requirements are introduced and investigated accordingly.

1) Generic Model of MobileQuiz2 [23]: The model was already described in section III as a generic model that is able to express various scenarios. The approach is lightweight and does not require a large variety of different elements: a scenario consists out of a set of objects with attributes and rules with conditions and actions. Nevertheless, critical issues became obvious during evaluation, namely the lacking comprehensibility as well as a low performance during the execution of more complex scenarios.

In order to define a baseline for current state of technology, this approach is investigated accordingly to the defined requirement criteria. The focus of the model was primarily set on modeling individual scenarios. Due to the chosen generic model, even small changes, e.g. changing the question type, result in extensive changes in the application model. Accordingly, the model is not real-time-capable in the definition of the requirement. The use of parameters to allow for different functional settings is not possible with this model. The way in which objects interact with each other is described by the rules associated with objects. These rules determine the circumstances under which certain actions are performed on the objects. Adapting the ARS to a specific setting requires changing the rules accordingly. Next, the model lacks in its simplicity. According to [23], it is often necessary to consult a didactic expert in order to model custom scenarios. Anyway, there exists a graphical editor containing a validation algorithm that will ensure to model valid scenarios. Hence, the requirement of validatability is fulfilled for the model. Last, the model is expandable since it is designed to describe various scenarios and is not limited to specific scenarios.

2) Autonomously Re-Configurable Workflows: The processes taking place during a lecture, which is supported by an ARS, are similar to the concept of workflows. According to the DIN institute, a workflow is a structured, computer-supported process. [26] Moreover, a process is a set of activities being in interrelationship or interaction, which transforms inputs into results. [27] investigated, how workflows are reconfigured automatically. Especially, the concept of spontaneously re-configurable workflows can be transferred to the domain of ARSs.

Workflows have similar structural elements known from common modeling languages. By default, they always start with a start node. The order of different activities (called tasks) is determined by transitions. Under certain circumstances, a transition can lead to a decision node (called OR-fork), which allows the branching into several nodes based on a condition similar to an if-statement of common high-level programming languages. The opposite of the decision node is the AND-fork, which splits the control-flow into several sub-flows. These sub-flows are executed in parallel. A join connects several sub-flows again. An extended language construct is provided by iterations that describe looping behaviors.

If unforeseen behavior occurs in the process of a workflow, it has to be reconfigured. In order to allow this, the dynamic adaptation of instantiated workflows has to be supported. Within the literature, multiple approaches were introduced [28], [29] that are divided into three categories: Re-configuration by evolution, reconfiguration by over- & under-specification and re-configuration by dynamic changes. In summary, workflows are able to be real-time capable. In addition, since workflows are
commonly described by domain-specific languages\textsuperscript{12} (DSLs), which allow, depending on the chosen language, for assigning attributes that could be used by an interpreter, workflows are able to be parametrized. The decision whether workflows fulfill the requirement of simplicity was evaluated within a user study, which will be summarized in section IV-B4. Next, the requirement of validatability is fulfilled since modern frameworks for creating DSLs allow for the creation of an interpreter for the described language that is used to validate application models. Finally, workflows can be defined to be expandable since it is possible to extend application models by new processes.

3) $\Sigma$-Automata: The behavior of event-discreet systems can be described by a sequence of events which cause a system to transfer from one state to another. Systems like ARSs work in a similar way. In computer science, automata theory describes the problem of whether a given word belongs to a previously defined language. The language is defined by an implicit or explicit set of strings and is generated from a grammar. Each string of a language is called a word. Strings usually consist out of several characters that are read by an automaton, where the set of all accepted strings is called the language of the automaton. An automaton changes its internal state after each string read in a previously defined manner. The following state can be unique (deterministic automaton) or ambiguous (non-deterministic automaton). Anyway, since these state transitions are unique within ARSs, it lends itself to investigate the representation of processes within ARSs by deterministic finite automata (DEA), or their extension $\Sigma$-automata that are described by [31] in more detail.

One state represents one specific function within the ARS. Transitions between those states are triggered by implicit (e.g. an expiring timer) or explicit (e.g. button press) events. However, the underlying grammar determines the transitions that are allowed. Validation mechanisms can use the same feature by automatically parsing all possible executions designed within a particular application model against a predefined grammar. This results in $\Sigma$-automata being real-time capable. However, this approach only allows for modeling states and their transitions without incorporating adaptable functionality via parameters which limits the extensibility and parameterizability. The decision, whether $\Sigma$-automata fulfill the requirement of simplicity was evaluated within a user study and is summarized in the following section.

4) Evaluation of Simplicity: Since simplicity is a subjective perception of individual users, it needs to be examined in a user study. This study was structured as follows: First, a short presentation introduces the participants to the topics ARS, foundations of modeling, workflows and $\Sigma$-automata in order to communicate a common understanding of the evaluated topic. Next, a questionnaire for the evaluation of simplicity of workflows and automata was executed. It was structured into 3 parts: In the beginning, questions regarding the participants and their educational background were asked. This was continued by asking more detailed questions about the understanding of both approaches. Finally, the participants had to decide for one of the both approaches. Following the questionnaire, the participants were invited for an optional task to model with automata and workflows, which was executed as a thinking-aloud test\textsuperscript{13}. During this test, we targeted to evaluate first experiences using the previously chosen approach.

The evaluation was executed by 11 participants, with 7 having an IT background. The participants rated both approaches as intuitive in their usage but decided unanimously for workflows. During the modeling tasks, which were executed as thinking-aloud tests, the participants modeled their scenarios using the chosen approach. The test has shown that there are two kinds of participants: One group modeled the scenarios in a simple manner to complete the task and another group modeled them using complex ideas. In addition, we found that the naming of functional elements and their parameters as well as the implementation of elements occurring in workflows, namely transitions, OR-forks, AND-forks, joins and iterations, will become a crucial aspect regarding the usability of the model.

C. Summary and Selection of a Solution

The usage of automata would provide a suitable and easy to understand solution for some requirements, but would still suffer from the non-supported requirements, similar to the approach of MobileQuiz2. Since Workflows provide a possible support for each requirement and the intended ARS processes are very similar to their behavior, they will be chosen as underlying solution for the further investigations. In Table I, a summary of the fulfilled requirements for the previously described approaches is provided.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Requirement & MobileQuiz2 & Workflows & Automata \\
\hline
Functional requirements & & & \\
Real-time capability & \xmark & \cmark & \cmark \\
Parameterizability & \xmark & \cmark & \xmark \\
Non-functional requirements & & & \\
Simplicity & \xmark & \cmark & \cmark \\
Validatability & \cmark & \cmark & \cmark \\
Expandability & \cmark & \cmark & \xmark \\
\hline
\end{tabular}
\caption{Fulfilled requirements of different approaches. \cmark = approach meets requirement, \xmark = approach does not meet requirement.}
\end{table}

D. Functions of the (Meta-)Model

Function blocks are abstract elements of the (meta-)model that can be concatenated to create application models. A function block describes one functionality of an ARS. The application model as a whole forms an abstract description of the ARS. The purpose of this section is to describe functions that will be implemented as function blocks in the next chapter. In addition, non-functional blocks as well as functions of workflows such as transitions have to be investigated.

\textsuperscript{12} A small, usually declarative, language offering expressive power focused on a particular problem domain. [30]

\textsuperscript{13} The participants have to share their ideas and opinion during the test.
1) **Functions of Workflows**: Every workflow starts with a **Start Node**. In the targeted scenarios, this node will be used to open an authentication dialog for participants, in which they can register themselves for an account or login. This function can be adapted by several parameters; e.g., whether the participants will be anonymously or identifiable, or if there is an access control activated, e.g., pin or password.

Workflows exactly determine in one **End Node**. However, if the workflows makes use of conditional branching, it is possible that multiple **End Nodes** exist. In our (meta-)model, this node will be used to model the end of the lecture.

**AND-forks** allow for parallel execution of multiple function blocks, e.g., showing multiple questions at the same time.

In order to merge several parallel control-flows, a **Join** is used.

**OR-forks** allow for conditional execution of different control-flows. In ARSs, this will be used to execute different function blocks depending on teacher-defined conditions.

**Iterations** can be used to model repetitions within control-flows, which allows for multiple execution of the same control-flow until a abort condition is met.

2) **Functional Blocks**: Functional blocks of ARSs represent their functionalities. The most popular functionality provided by ARSs is front channel functionality. It is used for several purposes:

- Request personal information and interests (e.g., the course of study or the reason for visiting the lecture),
- Poll pre-knowledge of the students,
- Repeat of knowledge at the beginning, middle and end of the lecture (self-testing),
- Evaluation of the lecture or course or preparation and post-processing of lectures.

This is just a reduced list considering well-known purposes for which ARSs are used. In order to express such functionalities, it is possible to distinguish between survey- and learning questions that are described in more detail in the following.

**Survey Questions** are used for requesting personal information and interests, polling the level of knowledge or doing evaluations. The answer options can be predefined or open.

**Learning Questions** allow for checking the knowledge of the audience within the lecture or prepare or post-process the lecture. These questions are checked automatically for being answered correct or incorrect. The answer options can be predefined or open.

**Results** are used to present the evaluation of survey- or learning questions in a suitable chart or overview.

**Closed Feedback** allows for providing a feedback on pre-defined feedback dimensions, e.g., the rating of the speed or volume of the lecturer.

**Open Discussions** provides a discussion opportunity, in which students can provide suggestions or vote on others suggestions.

**Attachments** allow for presentation of media during the lecture, e.g., text, programming code, images or videos.

**Formation of Groups** is used to create group participants and allow for in-group discussions and voting.

3) **Non-Functional Blocks**: Non-functional blocks allow for situations, where ARS functionality should not be inactive. **Pause Nodes** allow for hiding functionality; e.g., during the presentation of a specific topic, no ARS functionality will be visible during the **Pause Node**. **Pause Nodes** can be used to model the complete 90 minutes of a lecture.

**V. IMPLEMENTATION**

Within this section, the chosen (meta-)model will be adapted to our defined application scenario of modeling processes using ARS functions. First, the structure of the (meta-)model is presented. Next, the function blocks and parameters are summarized in tabular form before the used techniques for the realization of a DSL as well as their validation and transformation into a text format are presented. Finally, an example of a resulting application model is displayed.

**A. Structure of the (Meta-)Model**

The (meta-)model is based on the concept of workflows. For this reason, it is mandatory to integrate basic concepts such as **Transitions**, **OR-forks**, **AND-forks**, **Joins** and **Iterations** as fundamental elements using the **TransitionBlock**. In addition, function blocks occurring in ARSs have to be integrated. Since all function blocks share common parameters, e.g., a timeout until there will be a transition to the next element (if not done manually), a parent element called **FunctionBlock** exist. Furthermore, a **PauseNode** is added, which allows for modeling situations, in which no ARS functionality is active. Figure 2 summarizes the elements in their first level of inheritance.

**B. Function Blocks and Transitions**

The function blocks, which were described in section IV-D2, are extended by sub-function blocks to express the current functionality provided by ARSs. The parental function blocks can be used in the same manner as the sub-function blocks, which provide a more accurate definition of a specific function block. Table II summarizes every defined function blocks.

The non-functional block **PauseNode** as well as the **StartNode** and **EndNode** were implemented as seen in Figure 2. The transition blocks, which were introduced in section IV-D1, were implemented as heredity of a fundamental **TransitionBlock**, similar to the above shown types **LearningQuestions** or **SurveyQuestions**.
C. Techniques Used

There exist 3 groups of languages that can be used to describe a DSL: text-based-, graphical- and projectional languages. Since workflows are usually described as a graph with vertices and edges, we will focus on graphical modeling languages. For the current state of technology, Sirius\(^4\) is a popular and still maintained choice and was chosen for our project. Sirius allows for the creation of DSLs in the self-describing (meta-)language Ecore, which is part of the Eclipse Modeling Framework (EMF). Furthermore, Sirius provides extensive support for the creation of graphical interfaces based on Sirius Viewpoints.

Models based on Ecore are validated upon creation by the EMF. However, since the (meta-)model heavily uses Ecore’s EAttributes to describe parameters of function blocks, it is necessary to add further validation mechanisms by using the Object Constraint Language.

(Meta-)models based on Ecore are written in XMI/XML. To allow an easy integration into web-applications, we decided that it is beneficial to provide the same (meta-)model in the JSON format. We created a Model2Text transformation-template for the Acceleo code-generation engine to transform our (meta-)model into JSON.

D. Example of a Resulting Application Model

In the following example, the didactic concept of peer instruction, which was described in section II-B2, is modeled using the created (meta-)model. For visualization a graphical presentation of the resulting application model is displayed in Figure 3.

VI. Evaluation

In order to evaluate the created (meta-)model, it has to be checked, whether it is easy to understand and to model with. Since a graphical editor is not present at the moment, but will be added in the future, a paper-based approach simulating the editor was chosen. For each function block, including start- and end node, a paper-based representation has been created. Also, the different types of transitions had to be considered. The result is a construction kit able to build own scenarios within an ARS – similar to a future graphical editor.

A. Method

The evaluation itself was aligned to represent the execution order of the planned editor as accurately as possible. In order

---

\(^4\)https://www.eclipse.org/sirius/ – last successful access on August 23, 2019

<table>
<thead>
<tr>
<th>FunctionBlock</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleChoiceLearningQuestion</td>
<td>Choose one correct answer</td>
<td>anonymity, answerFeedback, showAggregatedResults, allowAbstention, numberOfRepetitions, questionText, recordConfidence</td>
</tr>
<tr>
<td>MultipleChoiceLearningQuestion</td>
<td>Choose multiple correct answers</td>
<td>–</td>
</tr>
<tr>
<td>FreetextLearningQuestion</td>
<td>Enter the correct textual answer</td>
<td>shortAnswer, characterLimit</td>
</tr>
<tr>
<td>NumericalLearningQuestion</td>
<td>Enter the correct numerical answer</td>
<td>enableEstimation, correctEstimationValue, correctEstimationRange</td>
</tr>
<tr>
<td>HotspotLearningQuestion</td>
<td>Select the correct point on an image</td>
<td>image, imageURL</td>
</tr>
<tr>
<td>MatchingLearningQuestion</td>
<td>Match choices together</td>
<td>–</td>
</tr>
<tr>
<td>OrderLearningQuestion</td>
<td>Bring choices in the correct order</td>
<td>–</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FunctionBlock</th>
<th>SurveyQuestion</th>
<th>anonymity, allowAbstention, questionText, voteCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>SingleChoiceSurveyQuestion</td>
<td>Choose one answer</td>
<td>showAggregatedResults</td>
</tr>
<tr>
<td>MultipleChoiceSurveyQuestion</td>
<td>Choose one or multiple answers</td>
<td>showAggregatedResults</td>
</tr>
<tr>
<td>FreetextSurveyQuestion</td>
<td>Enter a textual answer</td>
<td>shortAnswer, characterLimit</td>
</tr>
<tr>
<td>NumericalSurveyQuestion</td>
<td>Enter a numerical answer</td>
<td>–</td>
</tr>
<tr>
<td>HotspotSurveyQuestion</td>
<td>Select a point on an image</td>
<td>image, imageURL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FunctionBlock</th>
<th>ResultPresentation</th>
<th>displayType</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResultFreetext</td>
<td>Displays a wordcloud or textwall</td>
<td>swearwordFilter</td>
</tr>
<tr>
<td>ResultHotspot</td>
<td>Presents results of a hotspot question</td>
<td>–</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FunctionBlock</th>
<th>GroupBuilder</th>
<th>anonymity, groupSize, buildSchema, numberOfGroups</th>
</tr>
</thead>
<tbody>
<tr>
<td>GroupDiscussion</td>
<td>Creates a group discussion</td>
<td>allowVoting, discussionTopic, selectGroupAnswer</td>
</tr>
<tr>
<td>GroupChoosing</td>
<td>Allows students to choose a group</td>
<td>numberOfGroups, groupDescriptions, groupCreators</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FunctionBlock</th>
<th>MediaPresent</th>
<th>–</th>
</tr>
</thead>
<tbody>
<tr>
<td>PresentCode</td>
<td>Displays text or programming code</td>
<td>programCode, codeType</td>
</tr>
<tr>
<td>PresentCountdown</td>
<td>Displays a countdown</td>
<td>–</td>
</tr>
<tr>
<td>PresentImage</td>
<td>Displays an image</td>
<td>image, imageURL</td>
</tr>
<tr>
<td>PresentVideo</td>
<td>Displays a video</td>
<td>video, videoURL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FunctionBlock</th>
<th>OpenDiscussion</th>
<th>anonymity, visibility, allowVoting, allowAnswering, markCorrectAnswer</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunctionBlock</td>
<td>ClosedFeedback</td>
<td>anonymity, cooldown, showAggregatedResults, feedbackText</td>
</tr>
</tbody>
</table>
As a third and final task, the participants were asked to explain given scenarios modeled by the system that were depicted in a graphical notation. This task was used to evaluate the comprehensibility of the (meta-)model. The evaluation was executed using thinking aloud. In addition, after completion of each task, the participants were asked to fill out a 5-item Likert question, whether they understood the task or not. The evaluation ends with a two-sided questionnaire rating the overall (meta-)model.

B. Results

The evaluation was executed by 20 participants in individual sessions of around 20 to 30 minutes. 11 of the participants had an IT background and 14 already used models, e.g. UML.

The first goal of the evaluation was to check if the participants were able to understand the workflow-based modeling of ARS functionality. The results show that all participants had little to no trouble at all solving the task, resulting in 15 ratings for agreement and 5 ratings for a partial agreement that the given task was easy to execute. Since there was no lower rating, the 9 participants without an IT background were also able to model the given scenarios. Some participants mentioned that the naming of the (meta-)models’ elements are somewhat confusing – this needs to be investigated in the future.

The second goal was to check, whether the participants were able to combine specific problems with the appropriate function blocks. The description differed from the first task by having far less details about the (meta-)models characteristics solely describing the didactic needs of a potential lecturer. The results differ from the first tasks by having only 6 participants that agree with the simplicity of the task. Anyway, 13 participants partly agreed that the task was rather easy to them to execute. Only 1 participant rated the task as neither easy nor hard. Confusions occurred with the correct use of AND and OR and Join. A future graphical editor needs to support the correct usage of these elements.

Finally, the participants had to interpret an application model, which is displayed in Figure 3, showing the commonly referred didactic concept peer instruction by verbally interpreting every shown function block and the parameters used. This task should show that the participants are able to understand even more complex scenarios. 12 participants could explain the model without any difficulties, 6 participants partly agreed that they could easily solve this task, 1 participant rated it neutral and another participant rated it as partly hard to solve. The results of the 3 tasks are summarized in Figure 5.

In addition to these 3 tasks, a questionnaire about the functional behavior of the (meta-)model was included. 12 out of 20 participants think the (meta-)model is easy to understand, the remaining 8 participants partly agreed. Furthermore, 18 participants agreed that they would like to use an ARS based on the (meta-)model in the future.

The results confirmed our investigations for an intuitive (meta-)model. Anyway, it is important to note that the success of the (meta-)model and its future editor relies a lot on naming conventions for the function blocks as well as the parameters.
Optimization have to be conducted. Moreover, the planned editor needs to provide support during the modeling of parallel or conditional executions for allowing an intuitive, valid usage.

VII. TOWARDS THE CREATION OF AN ADAPTABLE ARS INFRASTRUCTURE

Providing platform independent access to software systems is crucial to ensure its availability to a broad audience. Since modern software for tech-enhanced classrooms (e.g. ARS) incorporate the BYOD paragon, it is important to support a variety of different operating systems for mobile devices such as Android, iOS and Windows Mobile. There exist multiple approaches to overcome this barrier, such as providing applications which run in a web-browser independently of the devices underlying operating system, feeling like a native application. Those web-applications use interfaces (e.g. REST) to communicate with a backend or runtime which offers the concrete functionality of an ARS. In [23], an approach was presented, in which the developed ARS (MobileQuiz) and its model were embedded in a LMS (ILIAS). However, this leads to the problem that universities, which did not use the same LMS with the identical configuration, were incapable of using this approach. To overcome this limitation, we purpose an independent infrastructure to run our (meta-)model-based approach in order to provide the maximum amount of flexibility and usability.

In the following, a concept for a fully scalable 3-layer architecture consisting of a frontend, a backend and a runtime-cloud, as depicted in Figure 6, is introduced. A prototype based on this concept is currently in development.

1) Frontend: The frontend provides the entry-point for the lecturers as well as the administrators of the system. It can be accessed through native or web-based applications. The main functions of the frontend are:
   - Secure authentication based on username and password,
   - Creation, modification and administration of application models based on our (meta-)model,
   - Import, export and sharing of application-models with other lecturers as pre-defined templates,
   - Capability to start the instantiation of application-models as a dedicated container in the runtime-cloud.

2) Backend: The backend is mainly used as a gateway for several types of communication (e.g. transmitting data or control signals) between the runtime, the database and the frontend. Crucial functions, which the backend provides, are:
   - Providing a scalable database that is synchronized between multiple instances of the backend,
   - Websocket-interface to pull new data from runtime-containers
   - REST-API to provide a management interface for lecturers and administrators,
   - REST-API for the registration of new runtime-servers,
   - Management of run-time containers and instances of application-models.

3) Runtime-Cloud: The runtime-cloud provides a scalable amount of servers which register themselves at the backend. They are capable of running virtual containers (e.g. Docker-containers) from which each one represents an instance of an application-model. Features, which the containers provide are:
   - Interpreting an application-model and executing ARS-functions based on it,
   - Temporary storage for runtime-data such as user-data or answer-data,
   - Providing a websocket-interface to push volatile runtime-data into the persistend backend database,
   - Providing a (web-)interface for students to answer questions and use the functions of a particular ARS instance.

As described earlier, our (meta-)model is based on the approach of workflow-modeling, which enables runtime-adaptions of instantiated application models. [27] showed that systems, which incorporate the modeling and execution of workflows, are commonly split in Workflow Management-Systems (WfMS) and a runtime (e.g. Process-aware information system) (PAIS), whereby the former is used to create, modify, share and delete workflows and the latter is used to run them as an instance as well as adding adaptation and interaction mechanisms (as seen in Figure 6). We plan to use this approach in the future.
VIII. CONCLUSION AND OUTLOOK

The goal of this paper was to present and validate a new approach for designing and using ARSs. We have shown the necessity of redefining the life-cycle of ARSs by describing the didactic requirements before implementing a concrete software, which enables a wider usage of didactic concepts and approaches like studio-based learning. We defined requirements for a generic (meta-)model, which allowed us to analyze multiple promising models and modeling-languages, from which we have chosen workflows for further research. Based on the domain of workflows, we created a new DSL to precisely describe the didactic processes within the domain of ARSs. Our approach focuses therefore on being automatically validatable, extensible, real-time capable and adaptable while simultaneously staying as simple as possible. We evaluated that our (meta-)model can be used to successfully model rather complex scenarios without having prior modeling knowledge nor experience with ARSs at all, just by receiving a short just-in-time introduction into the modeling concept. The evaluation showed that participants which are experienced in using ARSs could detect benefits in using our new approach compared to traditional ones and that participants which are inexperienced in using ARSs still understood the modeling process. Since we want to make the new approach available to as many people as possible, we proposed a concept for a scalable 3-layer based infrastructure which provides platform independent access. In the future, we will put additional effort in adding further function blocks, parameters, constraints and the implementation of a prototype to provide a complete and user-friendly experience in the future.

ACKNOWLEDGMENT

Special thanks to Robert Peine and Lidia Roszko for their participation in the research project as well as to Tenshi Hara, Iris Braun, Markus Wutzler and José Irigone for any inspiring discussion, feedback and proof reading.

Disclosure: This paper summarizes the core research contributions of Ilja Shmelkin’s master’s thesis [32].

REFERENCES