Towards an Adaptive Audience Response System Using Role-Concepts

Tommy Kubica

Technische Universität Dresden
Department of Computer Science
Computer Networks Group
tommy.kubica@tu-dresden.de

1st Thesis Advisory Board Report

Abstract. Universities still offer some courses as traditional lectures, tutorials or seminars, in which the lecturer gives his presentation while the students act as passive listeners, who try to absorb as much information as they can.
Audience Response Systems provide the opportunity to actively involve students during these courses by means of technical tools. Since most of these systems rely on specific didactic concepts, our goal is to use role concepts in order to support different didactic concepts and implement an audience response system adapting its behaviour and view accordingly.

1 Introduction

Since a long period of time, teaching in universities is accomplished using classic course types like lectures, tutorials or seminars. For each of them different problems occur. While large courses suffer from the missing interaction between the lecturer and the students, smaller courses struggle with low participation rates due to the anxiety of students to ask irrelevant questions or provide wrong answers.

With the increasing availability of wireless campus networks, the usage of mobile phones during courses arose. While most lecturers only observe the disadvantages of these devices, there are also promising ways for the course integration. Following "Mobile Phones in the Classroom: If you can't beat them, join them", *Audience Response Systems* are able to provide a separate communication channel between the lecturer and the students [1]. Using their own devices, students for example are able to answer multiple choice questions in order to check their gained knowledge or to ask own questions in order to solve open problems.

Each of these systems relies on specific didactic concepts. Depending on these concepts, the functional scope is tailored and different user roles are defined. While the role idea is clearly visible, there were, to the best of our knowledge, up to now only few investigations how role concepts can help within similar scenarios [2,3]. Using roles, our main goal will be the investigation of an audience response

system that is able to support different didactic concepts and adapt its behaviour and view according to changing concept stages. In addition, the simplification of the modeling and extensibility should be also part of our research.

The remainder of this paper is structured as follows: In section 2, we briefly present background information about audience response systems, didactic concepts and role concepts. In section 3, we focus on related work and motivate our approach before describing it in section 4. In section 5, our current research is presented and section 6 concludes this paper with a short summary and view on the next steps.

2 Background

2.1 Audience Response Systems

Audience Response Systems (ARS) are based on the concept of "bring your own device" (BYOD). Students use their own devices to execute functions during the ongoing course [4].

Due to the fact that these systems rely on specific didactic concepts, the range of functions is adjusted accordingly. In order to describe the functional scope of these systems, a basic classification [5] by Ebner et. al, which distinguishes between front- and back-channel functions, is presented.

Front-channel functions require an active break during the course to ensure that each student has the ability to execute a certain function. An example for this kind of functions is the answering of questions, which need time to think. Front-channel functions are further distinguished between *qualitative* and *quantitative functions*, whereas qualitative functions require an open-ended answer, and quantitative functions provide given choices that can be chosen.

Back-channel functions are running in the background of the course and do not need an active break during the course in order to be executed properly. An example for this kind of functions is instant feedback, which can be either given in *qualitative*, open-ended form or in *quantitative*, closed-ended form, in which specific answers are given.

It is important to note that also combinations of different variants exist, e.g., in some question types it is possible to select an answer option that requires additional open-ended feedback. Furthermore, it is also important to note that not every type of function is appropriate in every single scenario. Moreover, it is necessary to select a suitable functional scope [6].

In the domain of audience response systems, there exists a variety of systems. Our paper will focus on *Auditorium Mobile Classroom Service (AMCS)*[1] in order to motivate modifications that need to be done.

[1] `https://amcs.website/` - last successful access: 20.11.2018

Auditorium Mobile Classroom Service (AMCS) is an audience response system that primarily concentrates on front-channel functions by providing a variety of different question types in order to execute classic surveys or learning questions with individual feedback [7]. Additionally, extensive evaluation options are available to give the lecturer the opportunity to post-process his events in order to get information about misconceptions or general problematic topics [8,9].

The most popular back-channel functions, like instant feedback or a question and answer function, are currently disabled due to the results of various experiments showing the distractibility of students, but they are further investigated. However, contrary to other audience response systems, AMCS provides a functionality to send push notifications that depend on predefined conditions, e.g. answers to learning- or survey questions. This enables the option to send further material or hints to specific user groups [7].

Written in Ruby on Rails, AMCS is implemented in an object-oriented manner. Since AMCS relies on a postgres database, it is pretty fast at writing entries, but slows down in reading tasks.

2.2 Didactic Concepts

In this paper, we want to concentrate on didactic concepts that are used within audience response systems. That is why we need to define a few limitations:

- The concept needs to enable tool support
- and also provide a separate communication channel.

If we have a look at different audience response systems, a variety of didactic concepts occurs, e.g. *Peer Instruction*, *Classwide Discussions*, *Self-Regulated Learning* or *Just-in-Time Teaching*. Our paper will focus on *Peer Instruction* in order to visualize one possible use case.

Peer Instruction consists of different stages, namely a *brief lecture*, a *ConcepTest* and depending on the results a *peer discussion*. This overall procedure is repeated multiple times within a course (called lecture in the following). Figure 1 shows the execution in more detail.

After the brief introduction lecture, which takes normally 10-15 minutes, a ConcepTest is executed. Depending on the results, the progression is changed. A small number of correct answers results in the repetition of the introduction lecture and the repetition of the ConcepTest. A medium number of correct answers results in the peer discussion stage, in which students try to convince each other and afterwards the ConcepTest is repeated. A large number of correct answers is needed to continue with a short explanation and the next topic.

The usage of audience response systems can support multiple stages of this procedure: The ConcepTest for instance can be done by using Multiple Choice Questions, the peer discussion by using some kind of question and answer functionality.

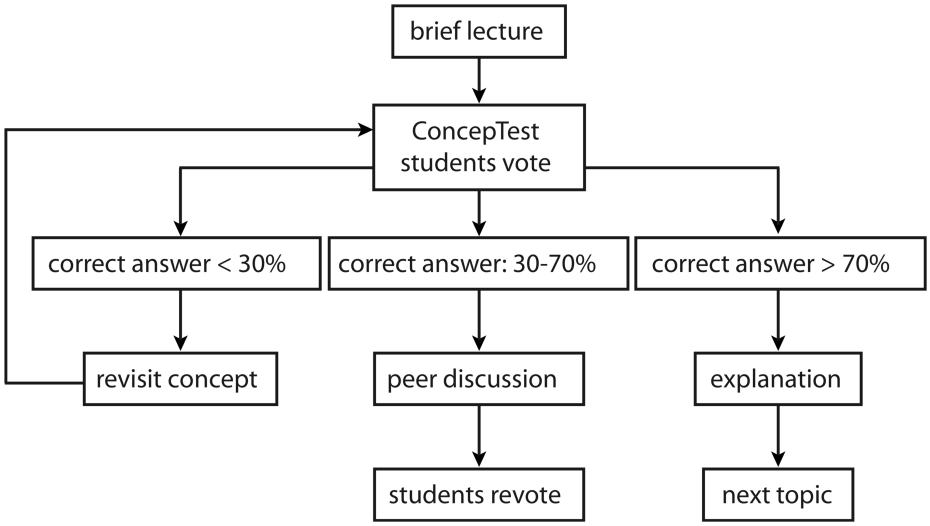


Fig. 1. The overall procedure of Peer Instruction by [10] that is executed multiple times within a course. It consists out of different stages that are called depending on the results.

2.3 Role Concept

The concept of roles in computer science was first described in 1977 by Bachmann and Daya [11]. While their work concentrated on a modeling approach for databases, the term *role* was used in many contexts afterwards. In 2000, Steimann investigated role concepts within a literature survey. He extracted a list of 15 features that roles should support but also had to admit "that there is no single definition of roles integrating all of them." [12]. Kühn et al. extended this feature list, which until then mostly concentrated on the relational nature of roles, by another 11 items in order to cover the context-dependent nature of roles, too [13].

Based on this feature list, the formal model for *Compartment Role Object Models* (CROM) with the corresponding *Compartment Role Object Instances* (CROI) and *Constraint Models* were introduced [14]. These formal models distinguish between four different types, namely *Natural Types*, *Role Types*, *Compartment Types* and *Relationship Types*. Natural types act similar to objects in an object-oriented manner. Role types are named places that can be played by natural or compartment types. Playing a role alters the behaviour and the properties of their players. Compartments are defined as "objectified collaboration with a limited number of participating roles and a fixed scope" [13]. They differ from the role type by the possibility to contain roles. Relationship Types represent a binary relation between two distinct role types.

2.4 Role-based Programming Languages

In the past, a variety of role-based programming languages has been developed that realizes the role concept using different approaches [13,15]. In the following, we will concentrate on LyRT and SCROLL that realize most features of the CROM role model.

LyRT introduces variability on the granularity of single objects, called *dynamic binding mechanism*, by providing a Java API [15,16]. The core of this runtime is a global registry that handles all states in a central lookup table. Within this registry, new cores, like player objects, can be requested. When binding roles, the registry stores the relation between compartments, players and roles, as well as the type, the level (that increases when roles play roles) and the sequence (that increases when multiple roles are bound at the same level) of this relation. In order to find the appropriate role implementing the targeted method, the role with the highest level and sequence will be chosen and executed.

SCROLL is a domain-specific language (DSL) written in Scala enabling role-based programming without the need for a specific runtime [15,17]. It uses a single underlying model (SUM) and provides *views* on that model. Role types are embedded within reified contexts, so-called *compartments*. By entering such contexts within a running system, the compartment got activated together with all its related roles. For each compartment the role-playing states of its roles are stored in a directed acyclic graph (DAG). The result of a role-playing object is a compound type, which is the intersection of all role types the object is playing. When a function is called that is not available on the role-playing object, the compiler rewrites the function call.

3 Related Work

The functional scope as single criterion is insufficient for choosing an appropriate audience response system. In [18], we investigated further selection criteria that are displayed in an *Index Card*[2]. Additionally, we applied this index card to a majority of 50 audience response systems to provide a filtered starting set of systems. To further support the selection process, a web-based filter tool was created that is able to combine multiple selection criteria and output links to the remaining, appropriate systems.

While our investigated list of systems is not complete and also further selection criteria can be defined, it is already recognizable that there will be no single best system that is appropriate in every single use case. Furthermore, we recognized that most systems rely on specific didactic concepts only, which affects the resulting functional scope. Often, the support is also limited to single stages of

[2] A suitable metaphor to comprehend the process and an easy-to-use method for filtering systems depending on a variety of attributes.

the didactic concept, e.g., in peer instruction, the peer discussion stage is often still done offline and the audience response system only provides a solution to support the ConcepTest.

Due to the fact that different situations require different didactic concepts and the used didactic concept depends on a certain lecturer, there is a need for an audience response system that is able to support different didactic concepts and that is easy to be extended. Can role concepts help to fulfill this belief?

4 Role-based Audience Response System

Audience Response Systems (ARS) define inherently different user roles, like the *Lecturer*, which is able to create and unlock questions or the *Student*, which can give answers to questions or provide instant feedback to the *Lecturer*. If we have a deeper look at these systems with regard to the application of role concepts, we recognize a lot more possible roles. For example, the aforementioned questions are defined in different types whereas some parts, e.g. *choices*, can be detected in multiple types. Using role concepts, it would be possible to model such types as combination of different role types. Nevertheless, within this section, we want to concentrate on roles for adding different didactic concepts.

4.1 General role idea

The main container of ARS is the representation of the current event, which is often called *session*, or *lecture* in more university-oriented systems. A *lecture* can be any of the aforementioned course types, like a classical lecture, tutorial or seminar. Within these *lectures*, different functions can be defined, as already observed in subsection 2.1. In order to model this in a role-based manner, these *lectures* would define a compartment, in which different roles are specified. An example of the above-mentioned minimal example is displayed in Figure 2. The depicted roles can then be played by any natural object, e.g. the *user*.

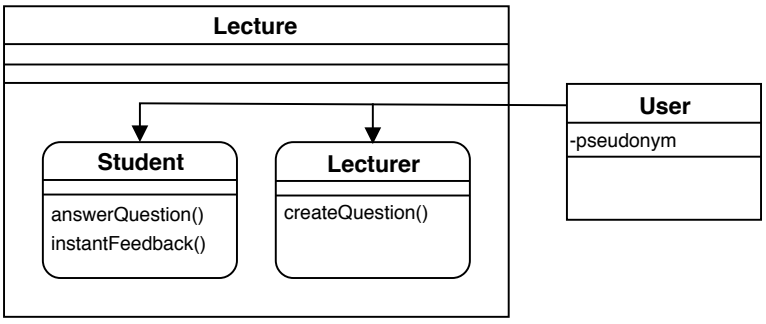


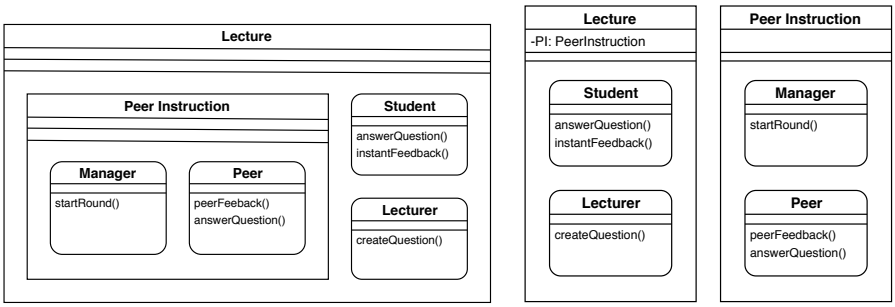
Fig. 2. Role Model of a minimal ARS example.

4.2 Use Case: Model didactic concepts using roles

Didactic concepts consist of multiple stages in which different roles are specified. The stage and the roles can change during runtime. If we have a look at peer instruction, we recognize two up to three stages per iteration. Depending on the *lecture*, there will be four up to six iterations per 90 minutes. This results in at least eight up to eighteen role changes only for switching the current stage of our *lecture*. Depending on the total number of roles inside these stages, the number of role changes could increase drastically.

In order to integrate didactic concepts into the minimal ARS example from Figure 2, it is needed to add multiple new roles, one for every single functionality, to our already existing *lecture* compartment. The most intuitive way to do this is by defining didactic concepts as compartments that are integrated in or linked to our *lecture* compartment.

While *Inner Compartments* benefit from the easy access on the inner roles of our *lecture* compartment, the re-use of them in other components is pretty limited. *Separate Compartments* provide a more valuable re-use, but have difficulties during the access of roles defined in our *lecture* compartment. A simplified example for each variant using peer instruction is depicted in Figure 3.



(a) Inner Compartment (b) Separate Compartment

Fig. 3. Different variants to combine *lecture* and didactic concept compartment on the example of peer instruction.

In order to find the most appropriate solution, both variants need to be further investigated. It could be also interesting to investigate combinations of different concepts in order to detect the feasibility, or possible dependencies or constraints between them.

5 Current Research

Based on the concepts of LyRT, which was introduced in section 2.4, a minimal role runtime written in Ruby was created, which supports basic role functions like *newPlayer()*, *newCompartment()*, *bind()*, *unbind()* or *invokeRole()*.

Derived from the knowledge about our application case, some assumptions of our role runtime already were proofed to be bad in performance. If we think about modeling each *lecture* as compartment containing multiple role changes inside, the size of our global registry as well as the amount of instances in the global instance pool would grow up tremendously. As a consequence, this would result in long request times in combination with an increasing server load, which is not targeted in audience response systems.

In order to motivate a possible solution, having a look at the role runtime SCROLL is promising. As introduced in section 2.4, SCROLL holds its role instances / objects per compartment, which avoids the tremendous blowup that occurs when adding new compartments. Nevertheless, it is also important to mention that this will not be the overall solution. Imagine a *lecture* containing 200 participants using our target system. Every single role binding would result in adding another step that needs to be traversed in order to find the appropriate method. The implementation of a real application needs to show the feasibility and serve as basis for experimenting on further solutions for providing a role-based adaptive audience response system that does not lack tremendously in performance.

6 Conclusion

Audience Response Systems provide a promising opportunity to solve well-known problems in classic course types, like the missing interaction between the lecturer and the students, by means of technical tools. In order to overcome the limitation of single underlying didactic concepts within these systems, which results in a static system behaviour and view, the introduction of roles can help to adapt to different didactic concepts and stages. This does not only ease the modeling of such systems, it actually also helps to understand the functionality of these systems.

In a next step, we want to further improve our role runtime, as already mentioned in section 5, and model and implement the aforementioned didactic concepts in order to show its feasibility. Therefore, we want to concentrate on modeling the didactic concept of Peer Instruction first, before creating a general solution being able to add different didactic concepts. Depending on the changing didactic concepts, the adaptation of the view also needs to be investigated.

References

1. E. Scornavacca, S. Huff, and S. Marshall, "Mobile phones in the classroom: if you can't beat them, join them," *Communications of the ACM*, vol. 52, no. 4, pp. 142–146, 2009.
2. H. Zhu, "Role mechanisms in collaborative systems," *International Journal of Production Research*, vol. 44, no. 1, pp. 181–193, 2006.

3. H. Zhu and M. Zhou, "Role-based collaboration and its kernel mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 4, pp. 578–589, 2006.
4. J. Imazeki, "Bring-your-own-device: Turning cell phones into forces for good," *The Journal of Economic Education*, vol. 45, no. 3, pp. 240–250, 2014.
5. M. Ebner, C. Haintz, K. Pichler, and S. Schön, *Technologiegestützte Echtzeit-Interaktion in Massenvorlesungen im Hörsaal. Entwicklung und Erprobung eines digitalen Backchannels während der Vorlesung.* Waxmann, 2014.
6. T. Kubica, T. Hara, I. Braun, F. Kapp, and A. Schill, "Guided selection of it-based education tools," in *Frontiers in Education Conference (FIE)*, pp. 1–9, IEEE, 2017.
7. I. Braun, F. Kapp, T. Hara, T. Kubica, and A. Schill, "AMCS (Auditorium Mobile Classroom Service) - ein ARS mit Lernaufgaben, Push-Notifications und umfangreichen Evaluationsmöglichkeiten.," in *DeLFI/GMW Workshops*, 2017.
8. I. Braun, T. Hara, F. Kapp, L. Braeschke, and A. Schill, "Technology-enhanced self-regulated learning: Assessment support through an evaluation centre," in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 1032–1037, IEEE, 2018.
9. F. Kapp, I. Braun, and T. Hara, "Evaluating lectures through the use of mobile devices," in *Proceedings of the 8th International Conference on Computer Supported Education*, pp. 251–257, SCITEPRESS - Science and Technology Publications, Lda, 2016.
10. N. Lasry, E. Mazur, and J. Watkins, "Peer instruction: From harvard to the two-year college," *American Journal of Physics*, vol. 76, no. 11, pp. 1066–1069, 2008.
11. C. W. Bachman and M. Daya, "The role concept in data models," in *Proceedings of the third international conference on Very large data bases*, vol. 3, pp. 464–476, VLDB Endowment, 1977.
12. F. Steimann, "On the representation of roles in object-oriented and conceptual modelling," *Data & Knowledge Engineering*, vol. 35, no. 1, pp. 83–106, 2000.
13. T. Kühn, M. Leuthäuser, S. Götz, C. Seidl, and U. Aßmann, "A metamodel family for role-based modeling and programming languages," in *International Conference on Software Language Engineering*, pp. 141–160, Springer, 2014.
14. T. Kühn, S. Böhme, S. Götz, and U. Aßmann, "A combined formal model for relational context-dependent roles," in *Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering*, pp. 113–124, ACM, 2015.
15. L. Schütze and J. Castrillon, "Analyzing state-of-the-art role-based programming languages," in *Companion to the first International Conference on the Art, Science and Engineering of Programming*, p. 9, ACM, 2017.
16. N. Taing, T. Springer, N. Cardozo, and A. Schill, "A dynamic instance binding mechanism supporting run-time variability of role-based software systems," in *Companion Proceedings of the 15th International Conference on Modularity*, pp. 137–142, ACM, 2016.
17. M. Leuthäuser and U. Aßmann, "Enabling view-based programming with scroll: Using roles and dynamic dispatch for etablishing view-based programming," in *Proceedings of the 2015 Joint MORSE/VAO Workshop on Model-Driven Robot Software Engineering and View-based Software-Engineering*, pp. 25–33, ACM, 2015.
18. T. Kubica, T. Hara, I. Braun, F. Kapp, and A. Schill, "Choosing the appropriate audience response system in different use-cases," in *International Conference on Education, Training and Informatics (ICETI)*, IIIS, 2019.