Towards a Pattern Catalogue for E-Assessment System Integration

Michael Striewe
paluno – The Ruhr Institute for Software Technology
University of Duisburg-Essen
Essen, Germany
michael.striewe@paluno.uni-due.de

***Abstract*— This paper presents preliminary results of an extensive literature study on software components commonly used in e-assessment systems. The purpose of the study is to prepare the creation of a pattern catalogue for design patterns, which can be used for integrating e-assessment features into larger systems.**

Keywords— E-learning and E-assessment systems, Design patterns, System integration

I. INTRODUCTION

Following a general tendency in system design and system architectures in recent decades, educational systems transformed in three generations from monolithic blocks via modular systems to service oriented frameworks [5]. This is a comprehensible development due to the many similarities between educational systems and other software products. Consequently, there is also a tendency in very recent years to move forward to cloud based solutions in e-learning and e-assessment, which is considered a fourth generation by some authors [11].

These trends were not only driven by purely technical innovations, but also by actual requirements in the context of these systems. For example, service oriented architectures were in particular introduced due to the need for sharing materials or assessments across courses and teachers or even institutions [1, 4]. A similar need for sharing expert systems and knowledge modules also led to modularization in the area of Intelligent Tutoring Systems (ITS) [8], which usually also include some kind of assessment features. Learning management systems (LMS) also included a rising amount of e-assessment features. Especially those systems that are developed (as open source projects) by a distributed community (such as MOODLE or ILIAS) benefit from modularization. With rising numbers of students and in particular rising numbers of electronic assessments, scalability became a crucial issue for e-assessment systems in particular and thus put arguments in favor of cloud solutions to the front [25].

While the notion of different generations of systems according to their architecture refers to the internal structure of these systems in the first place, modularization also is a prerequisite for constructing integrated systems. Integrated e-assessment can be understood in two ways: In the sense of technical integration of e-assessment features into other tools and in the sense of assessment activities integrated into larger educational contexts. Both cases are not possible from the software engineering perspective without understanding educational systems as a composition of components and services. Although situations might exist in which a system offering only e-assessment features is appropriate to use, ITS or learning management systems (LMS) can be expected to integrate e-assessment capabilities either as own components or as external services. This is the most favorable view in particular when using a broad notion of assessment that includes any kind of non-formal (self-)assessment that might occur during learning and training. Consequently, there will be no strict definition on how to tell an LMS with e-assessment features from an e-assessment system with LMS features and alike.

The remainder of this paper hence reports in chapter II on different kinds of components found in the literature, that typically appear in the context of educational systems. The assumption is that these components may be integrated with other components in a system offering e-assessment features. The goal of this chapter is hence to compile an overview including a rough description of component interfaces. The intention is to use this overview as a baseline for subsequent considerations on architectural patterns. Chapter III provides a first and preliminary sketch for these considerations. It takes some of these bits and provides abstract descriptions of some reoccurring patters found in the components and systems mentioned above. The goal is not yet to provide a full pattern catalogue, but to present and discuss abstractions on various granularity levels by example. Chapter IV reviews these results in order to name future work towards a more complete pattern catalogue.

II. A LITERATURE STUDY ON COMPONENTS

The following sections provide an overview on typical components related to e-assessment features that can be found in literature. The study includes publications from major conferences and journals in the computer-aided assessment and intelligent tutoring systems community as well as documentation for commercial tools. The literature study particularly includes (amongst other sources) a systematic review of papers from the *International Conference on Technology Enhanced Assessment* (TEA) (formally known as *International Conference on Computer Assisted Assessment*

(CAA)), the *IEEE Global Engineering Education Conference* (EDUCON), the *International Conference on Intelligent Tutoring Systems* (ITS) and the *IEEE Transactions on Learning Technology* (TLT). Although the study provides some remarks on the quantity of publications, its focus is on the qualities and characteristics of the components.

A. User Interface Components

The literature review identified three main user interface components, where one of them faces the students and two face the educators or administrators.

A *student frontend* (also called student LMS, student VLE, student CMS, student agent, or learning interface) is most commonly mentioned in literature [1, 2, 3, 8, 10, 11, 12, 14, 18, 25, 26]. It offers features to display assessments to the students and to retrieve their answers. The student frontend is thus typically highly interactive and the amount of different item types supported by an e-assessment system is typically determined by the amount of different types of interactions the student frontend is able to offer. This in turn explains the large amount of papers on student interfaces, as publishing new features in this area appears highly attractive for the community. Systems often employ one student frontend component, which is extensible by plug-ins (see section III.C).

A *teacher frontend* (also called teacher LMS, teacher VLE, teacher CMS, or admin agent) is mentioned less often explicitly in literature [2, 11, 17, 25]. It offers features for administration, authentication, and assessment scheduling. It thus aggregates the features related to the organizational aspects of assessments. As these are in the focus of research more rarely, publication counts for these interfaces are low, which does not imply that these interfaces are offered more rarely by e-assessment systems.

More often, an *authoring tool* is discussed explicitly in literature [3, 12, 17, 19, 20, 21, 22]. It offers features required to create contents, which in particular refers to assessment items, item pools, and grading schemas. It thus aggregates the features related to the educational aspects of assessment and is related more closely to the student interface and its features. Thus it is more in the interest of research and thus mentioned more often in literature, but also remarkably often by commercial tools.

B. Educational Components

The core of e-assessment systems are their educational qualities and thus the algorithmic power they offer for generating contents, providing advice, and evaluate answers. The literature study identified four components that relate to this area. They are discussed here in the order of appearance during an assessment.

An *assessment generator* (also called instructional manager, curriculum agent, task selector, tutoring component, or steering component) is mentioned very often in the literature [6, 8, 10, 15, 18, 20, 23, 25, 26]. It is concerned with preparing an assessment for delivery to the student. This often includes selecting appropriate items from an item pool in case of adaptive system behavior in order to individualize training or assessment. However, it can also appear in non-adaptive context in which nevertheless a particular exam needs to be retrieved from a database to be delivered to a student. As the former case attracts a lot of research, it is highly present in the literature.

An additional *problem generator* (also called item constructor) is mentioned sometimes in the literature as well [1, 20, 21, 26]. It is concerned with filling item templates with actual content, for example by creating random numbers. Consequently, it is not used in context in with fixed items are used and in which any adaptations are performed by the assessment generator mentioned above. This explains the lower number of occurrences in the literature.

A *pedagogical module* (also called hint generator) is mentioned sometimes in the literature [6, 10, 18, 26]. It is concerned with providing hints to students while they work on an assessment item. Consequently, these components primarily occur in assessments that focus on learning, training, or tutoring instead of formal evaluation of student performance. Notably, a literature review from 2009 [24] explicitly makes a distinction between plain feedback on correctness (which would refer to an evaluator component discussed in the next paragraph) and more intelligent analysis as required by a pedagogical module. Although one would expect the latter to be a crucial part of intelligent tutoring systems, the literature review reports a low occurrence rate of components for intelligent analysis of student solutions in intelligent tutoring systems (3 out of 34).

An *evaluator component* (also called checker, diagnose module, assessor, or expert module) is mentioned very often in the literature [1, 2, 3, 8, 10, 14, 18, 21, 26]. It is concerned with analyzing submissions from students and identifying mistakes that may occur in these submissions. As part of that, it is also concerned with the generation of feedback that is presented to the student. It is hence somewhat similar to the pedagogical module mentioned above and may be used by these modules. However, it also may be much more simpler in that it basically just applies a grading schema to a solution but is not able to provide any hint on how to improve a wrong solution. As this seems to be sufficient in several situations, an evaluator component is mentioned much more often than a pedagogical module. Large e-assessment systems often employ a large amount of different evaluator components, where each one is specialized to process a specific type of input or create a specific type of feedback.

C. Knowledge Representation and Storing Components

Virtually any e-assessment system contains a component for general data storage for users, assessment items, and solutions. These very basic features are common to almost every information processing systems and are thus out of scope for this literature study. However, there are also components for storing more specific data, which are often mentioned in the context of intelligent tutoring systems or adaptive assessment systems.

A *domain knowledge model* (also called knowledge base) is mentioned often in the literature [3, 6, 7, 8, 10, 18, 22, 26]. It is responsible for storing information on the domain of the

assessment, which are not specific to a certain assessment item, but reflect facts or competencies of the particular domain. Domain knowledge models are mentioned most often in conjunction with expert modules that are able to evaluate a submission by using domain knowledge, but without knowing the correct answer to the particular assessment item explicitly. The same goes for connections to pedagogical modules that use domain knowledge to generate hints.

A *student model* is mentioned often in the literature as well [3, 6, 7, 8, 14, 15, 18, 22, 23, 26]. It is responsible for storing information on a particular student, which again is not specific to a particular assessment item. Instead, a student model reflects competencies or similar properties that relate to the person and his or her capabilities or performance. These may be designed as records referring to an underlying competency model, which in turn is stored in a domain knowledge model as mentioned above. Student models are mentioned most often in conjunction with adaptive system behavior, where adaptation is based on the information stored in the student model.

Additional *domain-specific data storage* is mentioned only rarely in the literature [16]. It is relevant only in domains in which submissions to assessment items are large or complex objects, such as program code in the domain of programming assessment. Consequently, specific components for this purpose are explored only in conjunction with these domains and almost never as part of general assessment systems.

D. Management Components

The core features and requirements of e-assessment systems motivate the components discussed so far. However, additional requirements may introduce some more components. Some more components may exist primarily for the sake of better software architectures. In general, these components are far less present in the literature.

A *reservation service* realizes an additional feature of e-assessment systems reported sometimes in the literature and by commercial tools [16, 17, 20]. It is responsible for registering students for assessments and thus covers an additional part of the organizational process around assessments, which is not necessarily covered by the teacher frontend discussed in section II.A above.

A *service broker* (also called spooler or middleware) is mentioned in discussions of system architectures only [2, 9]. It connects some frontend or steering components to evaluator components that may run in parallel on separate systems for performance or security reasons.

An *infrastructure agent* is reported for cloud-based solutions only [25]. It is responsible for starting and shutting down instances of other components to adjust the size of the running system to the current needs. It is only necessary in systems which are aware of being a cloud system. Different to that, components can also be deployed as services in a cloud based or container based environment in which the underlying cloud or container infrastructure is responsible for starting and shutting down additional instances.

III. Architectural Patterns for E-Assessment Systems

The previous chapter reported on typical building blocks for e-assessment systems that have been found in recent literature. Based on these findings, this chapter now reports on patterns that can be considered useful when designing and engineering e-assessment systems using some of these components. A particular focus of these considerations is on questions regarding integration and thus also on well-defined interfaces that describe suitable connections. The idea of this chapter is to some extent inspired by the similar idea of architectural patterns for intelligent tutoring systems (ITS) explored by Andreas Harrer et al. 10 to 15 years ago [7, 13]. Unlike in that work, this chapter does not focus on the decomposition of a complete system into parts. Instead, it discusses system parts that can be integrated with each other or with other systems in order to create meaningful e-assessment features. To ensure a broader exploration of the design space, it is not limited to patterns found directly in the literature. Considering the limited space of this paper, the following sections primarily look at static aspects of system architectures, interfaces, and general data handling. Behavioral aspects (including adaptive behavior) are not discussed in this paper.

A. Component Types

As a general observation, one can identify two types of components: Passive services are waiting for requests that are directly or indirectly cause by user interactions. They perform some actions upon these requests and then wait for the next request to process. They can be considered a standard way of designing business information systems. Some literature mentions them as a general principle of system design [2, 4, 5]. In contrast to that, active agents have their own agenda on what to do and thus they perform their actions potentially even without any user input. They are used both for educational components (such as agents that generate hints or exercise suggestions without explicit request from the user) and management components (such as agents adjusting the cloud infrastructure to the current load). They are particularly common in the domain of intelligent tutoring systems [3, 23]

B. Data Storage

Regardless of the number and design of components, many systems employ the pattern of a central data storage, which accumulates data for all components. This is particularly useful when using several agents that are supposed to work on the same data. Moreover, data storage is centralized in cases in which most components are realized as stateless services. An alternative pattern is that of a distributed data storage, which is used when components typically process specific data that is of no meaning to other components, such as domain knowledge in different expert modules. A third and rarely used pattern is that of a duplicate storage, where data is prepared and stored in one place but copied to another place on demand. This is used for example when item pools are stored in one place for authoring and copied to another place when running an actual assessment.

C. Plug-In Types

User interface components offer various ways of how to integrate into larger context. One pattern is that of a native plugin, which implements the full feature set of the component. It is written in the same language as the host system and uses the data storage provided by the host system. This is the standard way of implementing plug-ins in the LMS MOODLE or ILIAS. An alternative pattern is that of a foreign plugin, which only implements a subset of the desired features directly. Besides connecting to the plug-in API of the host system, it also connects to an own backend component which implements the missing part of the feature set and often also offers its own data storage mechanism. The third alternative is that of an external tool. In this pattern, the host system redirects the user to the external tool via some standard API and receives a callback when the user has finished their duties there. This mechanism is also realized in LMS via the IMS-LTI standard.

D. Job Delegation

The connection between the student frontend and an evaluator component can be realized in many different ways. One pattern is that of a synchronous push. In this pattern, user interaction directly triggers the grading process and the user has to wait until the input is processed. Systems in which grading tasks are short running and in which the next step depends on the previous result usually employ this pattern. An alternative is the asynchronous push pattern, which also triggers the grading process directly, but without blocking user interaction by waiting. A third alternative is asynchronous pull, in which user input is stored in a queue and pulled from there by the evaluation module. This pattern often occurs in conjunction with a service broker component or with evaluator components realized as agents.

IV. CONCLUSIONS

The results achieved so far are instrumental in two ways: First, they suggest a structure for a classification of existing components and a pattern catalogue derived from existing systems and components. Second, they provide a preliminary overview on some design alternatives for designing integrated e-assessment systems. However, these results are far from being complete, yet, and hence more detailed research and work on pattern descriptions is still required to provide a more complete picture. In particular, a large body of standards existing in the domain of e-learning systems has not been reviewed so far. Behavioral aspects also need to be included during the next steps.

REFERENCES

[1] Armenski, G.; Gusev, M.: E-testing based on service oriented architecture. Proceedings of the 10th CAA Conference, 2006.

[2] Amelung, M.; Krieger, K.; Rösner, D.: E-assessment as a service. IEEE Transactions on Learning Technologies, 4:162–174, 2011.

[3] Costa, E.; Silva, P.; Silva, M.; Silva, E.; Santos, A.: A multiagent-based ITS using multiple viewpoints for propositional logic. Intelligent Tutoring Systems (ITS 2012), pages 640–641, 2012.

[4] Davies, W.; Howard, Y.; Davis, H.; Millard, D.; Sclater, N.: Aggregating assessment tools in a service oriented architecture. 9th International CAA Conference, 2005.

[5] Dagger, D.; O'Connor, A.; Lawless, S.; Walsh, E.; Wade, V.: Service-oriented e-learning platforms: From monolithic systems to flexible services. IEEE Internet Computing, 11(3):28–35, May 2007.

[6] Devedzic, V.; Radovic, D.; Jerinic, L.: On the notion of components for intelligent tutoring systems. Intelligent Tutoring Systems (ITS 1998), volume 1452 of LNCS, pages 504–513, 1998.

[7] Devedzic, V.; Harrer, A.: Architectural patterns in pedagogical agents. Intelligent Tutoring Systems (ITS 2002), volume 2363 of LNCS, pages 81–90, 2002.

[8] El-Sheikh, E.; Sticklen, J.: Generating intelligent tutoring systems from reusable components and knowledge-based systems. Intelligent Tutoring Systems (ITS 2002), volume 2363 of LNCS, pages 199–207, 2002.

[9] Garmann, R.; Heine, F.; Werner, P.: Grappa - die Spinne im Netz der Autobewerter und Lernmanagementsysteme DeLFI 2015 - Die 13. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI), 2015, 169-181

[10] Gonzalez-Sanchez, J.; Chavez-Echeagaray, M.; Van Lehn, K.; Burleson, W.; Girard, S.; Hidalgo-Pontet, Y.; Zhang, L.: A system architecture for affective meta intelligent tutoring systems. Intelligent Tutoring Systems (ITS 2014), pages 529–534, 2014.

[11] Gusev, M.; Ristov, S.; Armenski, G.; Velkoski, G.; Bozinoski, K.: E-Assessment Cloud Solution: Architecture, Organization and Cost Model. iJET, 8 (Special Issue 2):55–64, 2013.

[12] H5P.org. H5p documentation. https://h5p.org/documentation. Last accessed: 2017-12-01.

[13] Harrer, A.; Martens, A.: Towards a pattern language for intelligent teaching and training systems. Intelligent Tutoring Systems (ITS 2006), volume 4053 of LNCS, pages 298–307, 2006.

[14] Kenfack, C.; Nkambou, R.; Robert, S.; Nyamen Tato, A.; Brisson, J.; Kissok, P.: A brief overview of logic-muse, an intelligent tutoring system for logical reasoning skills. Intelligent Tutoring Systems (ITS 2016), 2016.

[15] Kurup, M.; Greer, J.; McCalla, G.: The fawlty article tutor. Intelligent Tutoring Systems (ITS 1992), 1992.

[16] Küppers, B.; Politze, M.; Schroeder, U.: Reliable e-assessment with GIT - practical considerations and implementation. EUNIS 23rd Annual Congress, 2017.

[17] LPLUS GmbH. Lplus: Portfolio. https://lplus.de/en/lplus-portfolio/. Last accessed: 2017-12-01.

[18] Martens, A.: Time in the adaptive tutoring process model. Intelligent Tutoring Systems (ITS 2006), volume 4053 of LNCS, pages 134–143, 2006.

[19] Martin, B.: Authoring educational games with greenmind. Intelligent Tutoring Systems (ITS 2008), volume 5091 of LNCS, pages 684–686, 2008.

[20] MapleSoft. Features in Maple T.A. https://www.maplesoft.com/products/mapleta/mainfeatures.aspx. Last accessed: 2017-12-01.

[21] Maths for More. Wiris quizzes - technical description. http://www.wiris.com/en/quizzes/docs. Last accessed: 2017-12-01.

[22] Murray, T.: Having it all, maybe: Design tradeoffs in ITS authoring tools. Intelligent Tutoring Systems (ITS 1996), 1996.

[23] Neji, M.; Ben Ammar, M.: Agent-based collaborative affective e-learning framework. Electronic Journal of e-Learning, 5(2):123–134, 2007.

[24] Papadimitriou, A.; Grigoriadou, M.; Gyftodimos, G.: Interactive problem solving support in the adaptive educational hypermedia system MATHEMA. TLT, 2(2):93–106, 2009

[25] Ristov, S.; Gusev, M.; Armenski, G.; Velkoski, G.: Scalable and Elastic e-Assessment Cloud Solution. IEEE Global Engineering Education Conference (EDUCON), 2014.

[26] Rickel, J.: Intelligent computer-aided instruction: A survey organized around system components. IEEE Transactions on Systems, Man, and Cybernetics, 19(1):40–57, 1989.