How Do We Get the Pieces to Talk? An Architecture to Support Interoperability Between Educational Tools

Andreas Harrer[a], Niels Pinkwart[b], Bruce M. McLaren[c,d], and Oliver Scheuer[c]

[a]Catholic Univ. Eichstätt-Ingolstadt, [b]Clausthal Univ. of Technology, [c]DFKI (Germany), [d]Carnegie Mellon University (USA)

Abstract. For many practical learning scenarios, the integrated use of more than one learning tool is educationally beneficial. In these cases, interoperability between learning tools – getting the pieces to talk – is a crucial requirement that is often hard to achieve. This paper describes an architecture that aims at the integration of independent learning tools into one collaborative learning scenario.

1 Introduction

In the field of educational technology, there have been numerous attempts in recent years to connect differently targeted learning tools to one another. For example, a teacher may want to start her course with students' individual intelligent tutoring sessions, followed by plenum discussions about the topic, then some small-group work with simulation tools, and finally followed by individual essay writing. For each of these sessions, there is likely to be a different tool that is suitable to support the activity, such as Cognitive Tutors, discussion support tools, and educational simulations, yet these individual tools do not usually interoperate or exchange data, resulting in scattered artifacts and functionality that is hard to integrate.

Especially in the field of collaborative learning, tool interoperability and data exchange between heterogeneous learning tools is a crucial requirement. This is so because the data flows in group learning tend to be more complex and require data exchange between a greater variety of tools (such as discussion and graphical mapping tools) and more instances of such tools (e.g., one per student) than in individual learning scenarios. Many collaborative software tools have the advantage that they need to externalize their data anyhow to allow users to exchange data with peers (and thus transfer this data between applications). This characteristic should facilitate inter-tool data exchange – yet, reality has shown that in the field of educational technology, not many collaborative learning tools are interoperable.

In our efforts to implement longer-term learning flows, we have frequently encountered the need to make our tools – such as Cool Modes [1] or FreeStyler – interoperable with other tools. A recurring approach we have adopted is to employ a generic data storage and exchange mechanism, and use this to achieve seamless communication with learning components through the use of *adapters*. This solution principle, which we call the "Scalable Adapter," has proven successful in a number of different scenarios and projects. Thus, we propose this principle as a general software design pattern [2], i.e. a re-usable solution to the named problem,and discuss it together with its implementation in the remainder of the paper.

2 The "Scalable Adapter" Design Pattern

This section describes the "Scalable Adapter" design pattern which constitutes a software architecture that can be used to create interoperability between different educational tools, in particular between collaborative learning tools and intelligent tutors.

The *problem context* is that there exist learning tools (e.g. discussion tools, simulation tools, or ITS systems) that each provide specific functionality and data. Part of this data can be used to inform other tools within an integrated learning scenario. The recurring *problem* to solve is that the different preexisting learning tools need to interoperate with each other through data exchange. Potentially, each environment is interested in only specific portions of the data of other applications. Since it cannot be foreseen which applications need which parts of the data, a flexible design solution is required. Another *force* to be considered is that the existing learning tools should not need to be altered (or at least not much) in order to facilitate their use in their original context. Yet, the interoperation and data exchange between the systems must be supported in a flexible way to allow for arbitrary learning data to be exchanged.

The *solution* we propose is to extend existing learning tools with specific adapter components [2] that provide the interoperability with other components. The granularity of the information to be exchanged between components is tailorable in a scale-free way through the use of a composite data structure. The **Adapters** leave the original learning tools unaltered for the most part. They provide the interface for interaction between the learning tool that the adapter is attached to and the other components used within an integrated learning scenario. The **Composite Data Structure** provides access to arbitrary parts of the data to be shared between the learning applications. Additionally, a subscription mechanism for parts of this data structure is provided. This mechanism uses notifications to inform registered learning tools about changes in shared data (parts), thus avoiding inefficient communication via active polling processes. The **learning tools** use the functionality of the adapter to gain access to the data elements of interest. The processing of the data (i.e., the interpretation of the exchanged learning data) is fully encapsulated within this component.

The learning tools and the composite data structure are completely decoupled (i.e., the shared data is separated from the specific tools), with the adapter assuming a mediating function between these two. The composite data structure provides access to arbitrary data elements using a tree structure (de-)composition. Note that there is a 1-to-n relation between the data structure and the adapters and a 1-to-1 relation between an adapter and a learning tool. This implementation requires both minimal changes to the learning tools (only the communication with the adapter has to be developed) and allows multiple learning tools to access the shared data. The composite data structure allows different learning tools to use different (or the same) parts of the shared data.

The typical component interaction in this micro-architecture is that a learning tool lt_1 sends out data changes (e.g. learner actions) via an adapter a_1 to the composite data structure. This notifies all registered adapters about changes to the respective components of the structure, enabling the adapters to process relevant information only and thus to communicate efficiently. Each adapter a_i updates its learning tool lt_i. This way, application lt_i can be informed about the relevant changes caused by students or system actions.

3 Using the "Scalable Adapter" in the CoChemEx Project

The Scalable Adapter pattern was used within the CoChemEx project [3], where the virtual chemistry experimentation laboratory "VLab" [4] is used within different collaborative inquiry learning scenarios. Here, the features for collaboration and communication of the shared workspace systems FreeStyler / Cool Modes environments [1], such as chat and graphical argumentation, are used in conjunction with the sophisticated experiments students can conduct within the VLab.

In the CoChemEx project, the educational scenario was defined by researchers and practitioners from educational psychology and chemistry education. The scenario is implemented by a collaboration script [3] consisting of several phases of activities. These are represented by separate tabs in the FreeStyler environment.

In this learning scenario, interoperability and data exchange between the different pre-existing tools – in particular, between VLab and FreeStyler – was a crucial requirement. Following the Scalable Adapter design pattern principle, this interoperability – and thus the integration of the VLab into a collaborative context – was achieved via a newly developed *VLab adapter* that creates a communication channel to FreeStyler through the jointly-used data stored in a *composite data structure* (MatchMaker). The use of the Scalable Adapter pattern to connect the applications has two immediate advantages. First, it enables the VLab to interact with other VLab instances of the collaborators, thus supporting collaborative experimentation, and, second, it enables data exchange between the experimentation functions in VLab and the FreeStyler tools which are valuable for experimentation, such as hypothesis generation and documentation of experiments. Both features are important contributions towards richer learning experiences that only integrated solutions combining various learning tools can achieve.

One conceptual challenge we still must tackle is that for every learning tool, a specific adapter must be defined. Also, the shared composite data structure is currently scenario-specific. Mechanisms for a more flexible configuration of the data structure by explicit specification of structure levels seem feasible and preferrable to hard-coding all dependencies.

Acknowledgements: The Pittsburgh Science of Learning Center, NSF Grant 0354420, supported this research. We are grateful to Nikol Rummel for her contributions.

References

1. Niels Pinkwart. *Collaborative Modeling in Graph Based Environments*. PhD thesis, Universität Duisburg-Essen, 2005.
2. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns. Elements of reusable Object-Oriented Software.* Addison-Wesley Professional, Boston, MA (USA), 1995.
3. Bruce M. McLaren, Nikol Rummel, Dimitra Tsovaltzi, Isabel Braun, Oliver Scheuer, Andreas Harrer, and Niels Pinkwart. The CoChemEx project: Conceptual chemistry learning through experimentation and adaptive collaboration. In *Proceedings of the Workshop on Emerging Technologies for Inquiry Based Learning in Science at AIED 2007*, Los Angeles, CA, 2007.
4. D. Yaron, K. Evans, and M. Karabinos. Scenes and labs supporting online chemistry. In *Proc. of 83rd Annual AERA National Conference*, 2003.