

The Scalable Adapter Design Pattern: Enabling Interoperability between Educational Software Tools

Andreas Harrer, Niels Pinkwart, Bruce M. McLaren, and Oliver Scheuer

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 to one another in a coherent, well-founded manner—is a crucial requirement that is often hard to achieve. This paper describes a reusable software design that aims at the integration of independent learning tools into one collaborative learning scenario. We motivate the usefulness and expressiveness of combining several learning tools into one integrated learning experience. Based on this, we sketch software design principles that integrate several existing components into a joint technical framework. The feasibility of the approach, which we name the “Scalable Adapter” design pattern, is shown with several implementation examples from different educational technology domains, including Intelligent Tutoring Systems and collaborative learning environments.

Index Terms—Computer applications, group and organization interfaces, computers in education.

1 INTRODUCTION

IN the field of educational technology, there have been numerous attempts in recent years to connect differently targeted learning environments to one another. The desire to do so is understandable—a variety of “micro-level” tools that teach students specific skills—e.g., Intelligent Tutoring Systems (ITSs) that help a student solve algebra equations (e.g., [1]) or collaborative technology that helps students learn to solve problems in a cooperative manner (e.g., [2])—have been developed and shown to be successful in targeted educational scenarios. Nevertheless, the support of longer term educational learning goals, relevant to everyday teaching practice, requires educational technologists to think more broadly, beyond the use of a single tool.

For example, a realistic educational scenario is that a teacher may want to start his course with individual student sessions with an ITS, followed by plenum discussions about the topic (face to face or via a Web forum), then some small-group work with simulation tools

like Netlogo,¹ and finally followed by individual sessions in which students write essays about their work. 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 simulation tools, yet these individual tools do not usually interoperate or exchange data, resulting in scattered artifacts that are hard to integrate with one another.

Another use case where interoperability between heterogeneous educational tools is an essential requirement is monitoring and assessment. If students use single isolated tools in different learning phases (such as described above), this may well be helpful for their learning processes. Nevertheless, in a classroom context, it is also important to inform teachers about the progress of their students. Provided that educational technology systems are able to exchange data, this can be automated by sending data to monitoring and reporting tools for teachers (and researchers), resulting in a unified view and representation of student activities.

The interaction between tools and the exchange of data between them makes a lot of sense also in scientific inquiry learning scenarios [3] where several phases of activities are usually required to allow the students to gain scientific insight into the phenomenon under investigation. The preparation and planning of the research typically requires different tools than the experimentation (in real or in a computer-simulation); the same holds for the documentation and experiment reflection and potentially for the definition and refinement of the initial hypothesis and plans. In order to help the student or student groups during the different steps of this inquiry procedure, it makes sense to enable them to have hypotheses, experimentation data, and reflections at hand to generate reports or essays, i.e., the availability of the outputs of experiment phases as a

- A. Harrer is with the Catholic University Eichstätt-Ingolstadt, Ostenstr. 14, 85072 Eichstätt, Germany. E-mail: andreas.harrer@ku-eichstaett.de.
- N. Pinkwart is with the Clausthal University of Technology, Institut für Informatik, Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany. E-mail: niels.pinkwart@tu-clausthal.de.
- B.M. McLaren is with Carnegie Mellon University, Human Computer Interaction Institute, 2617 Newell-Simon Hall, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891, and also with the Competence Center for e-Learning, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany. E-mail: bmclaren@cs.cmu.edu.
- O. Scheuer is with the Competence Center for e-Learning, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany. E-mail: Oliver.Scheuer@dfki.de.

Manuscript received 24 Jan. 2008; revised 3 Nov. 2008; accepted 4 Nov. 2008; published online 13 Nov. 2008.

For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLT-2008-06-0051. Digital Object Identifier no. 10.1109/TLT.2008.18.

1. <http://ccl.northwestern.edu/netlogo/>.

learning resource in different phases is highly desirable for rich learning scenarios.

Such a smooth flow of learning phases, called “educational interoperability” by Jansen et al. [4], can only be achieved if the underlying technology allows for an exchange of educational data objects between differently-targeted systems. However, this kind of interoperability is not readily available in today’s learning environments. Why is this so? A short answer is that achieving this kind of interoperability is difficult since many of the highly effective learning environments are not designed with interoperability in mind as a primary design concern. Almost a decade ago, challenges for making educational systems more interoperable through good computational design were identified [5], [6]. The vision of “educational software components” that can be connected as easy as copy & paste, expressed in these papers, led to the idea that design patterns and programming conventions should be used to attract developers to re-use systems, and to facilitate this re-use on the code level [5]. Other attempts to achieve such interoperability between educational technology components, but focused primarily on making Cognitive Tutors interoperable with other components, were [7], [8].

Especially in the field of collaborative learning, tool interoperability and data exchange between heterogeneous learning environments is a crucial requirement. This is so because the data flows in group learning tend to be more complex and require a 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 need to externalize their data anyhow to allow users to exchange data with peers (i.e., they transfer this data between applications anyway). This fact should facilitate intertool data exchange. Nevertheless, practice shows that in the field of educational technology, not many collaborative learning tools are interoperable. Recent initiatives toward interoperability for collaborative tools include the Collaborative Inquiry and Experiential Learning (CIEL) project and the Scalable Architecture for Interactive Learning (SAIL) architecture. In CIEL, the conceptual integration of collaborative inquiry scenarios was implemented by means of technical broker architecture [9] and standardized data formats for the exchange between different learning tools. The SAIL approach [10] provides a technical framework that can be used to realize learning scenarios based on different learning tools, including the well-known Web-Based Inquiry Science Environment (WISE) tools [11] that have been refactored into SAIL components.

A key focus of our research and development during the past few years has been the investigation of collaborative modeling tools as learning environments. In this line of work, we have frequently encountered the need to make our tools—such as Cool Modes [2] or FreeStyler [12]—interoperable with other tools, either to investigate new research ideas, to implement the research results in school practice, or to implement longer term learning flows. Usually, the requirements included that 1) the usage of the individual tools in their original contexts should not be affected, 2) the solution has to be inexpensive to implement, and 3) the approach to establish interoperability should be applicable to a wide range of tools.

A recurring approach we have adopted based on these main objectives is to employ generic data storage and exchange mechanism for data and use this to achieve seamless communication at a customizable and freely scalable level among learning components through adapter components. 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 [13], i.e., a reusable solution to the named problem and present it together with its implementation in the remainder of the paper. We also describe several examples of its use, with a focus on our newest integration scenario in a collaborative chemistry education project.

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 differently targeted educational tools, including but not restricted to collaborative learning tools and ITSs. The key idea behind the pattern is to add a small “data adapter” to each learning environment. The adapters can then access arbitrary (scalable) parts of the data of “their” learning environment and can exchange this data with other adapters.

These changes are not costly and usually easy to create since the existing systems do not have to be changed but merely need to be extended. As will be shown in the following, this design pattern provides a solution to a recurring problem in educational technology development: the data exchange between tools in order to allow for longer learning sequences that require differently targeted systems to be used. Using the software design pattern can reduce the development time needed for the construction of learning tools, since functionality from existing systems can be reused.

The presentation of the “Scalable Adapter” design pattern in this paper loosely follows the standard pattern description format [13], [14]. We start by explaining when the pattern can and should be applied—i.e., which specific problem it solves, in which context it does so, and what other requirements (forces) are central for a design solution in which the Scalable Adapter makes sense. Then, we describe how the design pattern solves the problem. In the tradition of design pattern descriptions, these sections (solution and structure) are kept on an abstract reusable level and in a format that describes the arrangement of elements (code classes and objects) that solve the problem. Finally, we illustrate other known software design patterns that are closely related to the Scalable Adapter.

2.1 Problem Context

There are existing learning environments (such as discussion tools, simulation tools, or ITS systems) that each provide specific functionality and data. Parts of this data can be used to enrich one another’s features within an integrated learning scenario. The individual tools offer some kind of API that provides access to their data (e.g., through a publish-subscribe-like mechanism, via network messages, or based on files). If such an API does not exist for the individual tools, then it typically can be added without a large effort: For collaborative systems, a communication API to exchange

data with other tools exists anyway. For noncollaborative systems that store their application data at a central place (e.g., the database), the API can usually be implemented easily since only the data storage component (but not the whole system) has to be extended.

2.2 Problem

The different preexisting learning environments should 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 and scalable design solution is required.

2.3 Forces

The existing learning environments should not need to be altered (or at least not much, see comment about API above) in order to facilitate their use in their original context and to minimize code changes: any changes to existing software are usually costly. Nevertheless, the interoperation and data exchange between the systems must be supported in a flexible way in order to allow for arbitrary learning objects (in the sense of parts of application data) to be exchanged. This property of allowing access to arbitrary portions of the application data will be called *scalable* in the following, because it enables the interoperation of multiple different learning tools at different granularity of data. In addition, the design solution must be applicable also to collaborative systems, since a considerable number of today's learning systems have support for groups of learners.

2.4 Solution

The existing learning environments are extended with specific adapter components [13] that provide the interoperability with other components. The granularity of the information to be exchanged between components is tailorable in a scalable way, i.e., it can be adapted to the specific characteristics of the applications used through the use of a composite data structure [13]. This data structure contains the educational data of the overall system, which may contain data provided by different tool types—including, e.g., learner models, user actions, or discussion logs.

- The **Adapters** leave the original learning environments unaltered for the most part. They provide the interface for interaction between the learning environment that the adapter is attached to (using the API of the learning environment or some other available way to access its internal data) and the other components used within an integrated learning scenario that includes multiple tools.
- The **Composite Data Structure** provides a central access place to arbitrary parts of the data to be shared between the learning applications. The hierarchical structure not only is just a structuring principle from computer science but also allows representation of educationally meaningful information at different abstraction levels and scale. Additionally, a subscription mechanism for parts of this data structure is provided. This mechanism uses notifications to inform registered learning environments about changes in shared data (parts), thus avoiding

inefficient communication via active polling processes. Technology wise, the Composite Data Structure is the key to keep development costs low: without it, adapters for all possible pairs of tools might be needed. For n tools, this would amount to $n(n - 1)$ adapters versus n adapters when using the Composite Data Structure. From an educational point of view, a hierarchical tree data structure is often adequate—e.g., to represent learner models (with different skills), workspace contents, or user actions in a system (indexed by user and system component).

- The **Learning Environments** use the functionality of the adapter to get access to the data elements they are interested in for internal use. The processing of the data (i.e., the educationally meaningful interpretation of the exchanged learning objects) is fully encapsulated within this component.

For the implementation of the pattern, it is critical that both the syntax and semantics of the data represented in the composite data structure and the communication/interoperation interface between the components be explicitly defined and that the data can be handled by all relevant components. This entails technical formats for the messages, such as XML-Schema or a formal grammar, as well as semantic interoperability, i.e., the ability for the learning environment using the adapter to process data coming from the composite data structure.

2.5 Structure

The typical structure of this pattern can be seen in the class diagram in Fig. 1. The diagram shows the three types of components and their relations to one another. Each learning environment 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 one-to- n relation between the data structure and the adapters and a one-to-one relation between an adapter and a learning environment. This implementation requires both minimal changes to the learning environments (only the communication with the adapter has to be developed) and allows multiple learning environments 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 microarchitecture is visualized in Fig. 2 with the learning environment (le1) processing data internally (e.g., through student actions), then sending out data changes via the adapter (a1) to the composite data structure. The notification mechanism informs all registered adapters about changes to the respective components of the composite data structure, enabling the adapters to process relevant information only and thus to communicate efficiently. The composite data structure is responsible for the hierarchic delegation of the messages: update messages received by one entity in the data structure are delegated to its children (recursively), which then notify learning environments interested in them. In the figure, adapter a2 is notified and updates learning environment le2. This way, application le2 can be informed

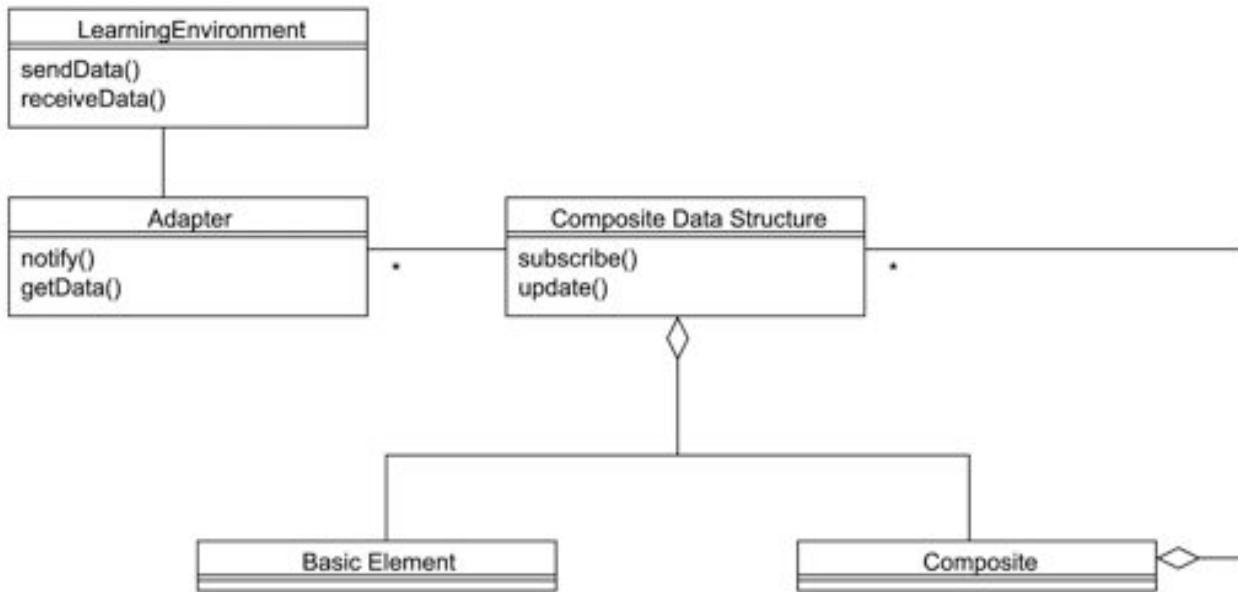


Fig. 1. Diagrammatic representation of the scalable adapter pattern.

about the specific actions in the data (caused by student or system actions in one of the other learning environments).

2.6 Related Patterns

The Scalable Adapter is closely related to some other design patterns. In particular, it is similar to the Blackboard Architecture [14], where components called knowledge sources communicate and interoperate indirectly using a blackboard as a shared communication medium. Given that the blackboard provides means for hierarchical structure in the data, the composite data structure is feasible to be used in this architecture. Current implementations of the blackboard approach frequently refer to the Linda coordination approach and TupleSpace implementations [15]. However, the Blackboard Architecture does not solve the interoperability problem of different preexisting systems, each with their own data structure, with the additional requirement of minimal changes to these systems. The adapter approach and the loose coupling can also be linked to proposals using cooperating agents, which leads to related approaches for multiagent proposals of learning environments and tutoring systems, such as [16]. The Scalable Adapter also uses the established design patterns Adapter [13], Composite [13], and Publisher-Subscriber [14] as subcomponents: The

Adapter is used for the integration of the learning environment into the interoperable system, the Composite for the realization of the hierarchically nested data structure, and the Publisher-Subscriber for the change-propagation mechanism that notifies each subscribed adapter about changes in the data structure.

3 EXAMPLE USES OF THE “SCALABLE ADAPTER” DESIGN PATTERN

This section describes three different example applications where the Scalable Adapter pattern has been used to make existing learning tools interoperable. These examples demonstrate the flexibility of the design pattern: the systems connected via the design pattern within the integrated learning scenarios are different in each case and the individual connected tools are heterogeneous, including argumentation tools, simulation environments, example-tracing tutors, chats, discussion support tools, and virtual experimentation systems that were created by different research groups and software developers. Together, these three example scenarios demonstrate how, using the Scalable Adapter pattern, interconnections between different learning

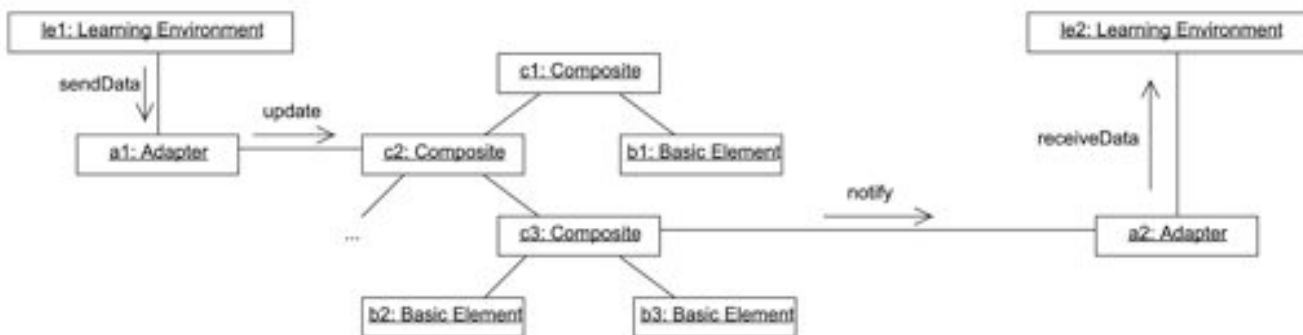


Fig. 2. Diagrammatic representation of the interaction.

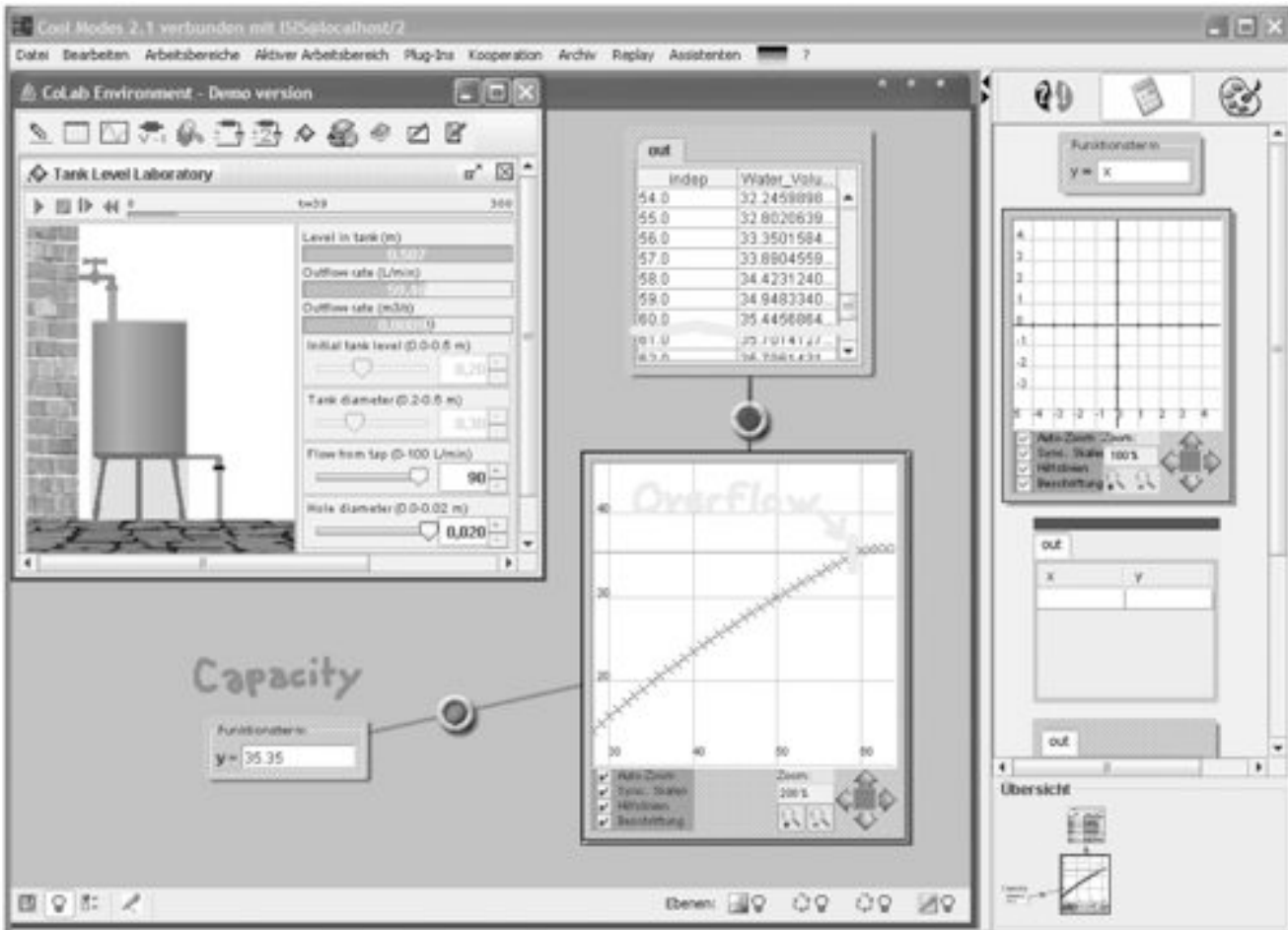


Fig. 3. Screenshot of the integration of a Co-Lab simulation tool with the modeling tool Cool Modes.

systems can be implemented in a (relatively) easy manner that does not require many changes to the existing tools.

3.1 NetCoIL

An integration of two existing learning applications into a joint scenario was achieved in the NetCoIL project with the simulation tool Co-Lab [3] and the collaborative graphical annotation/argumentation tool Cool Modes [2] (Collaborative Open Learning and MODELing System). The Co-Lab system is the outcome of a three-year European research project. The aims of the system are to help learners to develop flexible knowledge in science domains, collect, and synthesize information and to collaborate with others. For these aims, the Co-Lab learning environment offers facilities for experimentation (including remote laboratories), collaboration, and domain modeling. Cool Modes, on the other hand, is a flexible CSCL framework that allows learning groups to synchronously manipulate shared visual representations. These shared representations can be flexibly defined and include brainstorming and discussion maps, handwritten notes, modeling languages like Petri Nets or System Dynamics, and mathematical tools such as function plotters. Cool Modes allows these representations to be mixed, thereby facilitating smooth transitions between phases in group learning processes (where each phase might be associated to a typical representation that is being used). In the Networked Collaborative Inquiry Learning (NetCoIL) project, one of the research aims was to find

ways to interconnect various educational tools that have a value in an inquiry learning process. Co-Lab and Cool Modes were among the considered tools, since they facilitate essential steps in collaborative inquiry learning: hypothesis generation and testing, as well as group discussion. In order to connect the two tools (Co-Lab for simulation and Cool Modes for discussion) in a meaningful way, a requirement was to allow a real data flow between simulation and discussion phases, i.e., learner groups should have their simulation results available once they begin a discussion about experiment outcomes.

To enable this, data produced in Co-Lab simulations was sent to Cool Modes, where the data was collected within a table and updated immediately if new data from the Co-Lab simulation components was received. This data table was visualized in a graphical plotter and was then ready to be discussed and annotated using the tools that the Cool Modes environment offers. A demonstrator of this scenario with a Co-Lab water-tank simulation is shown in Fig. 3. This figure illustrates the educational usefulness of combining Co-Lab with Cool Modes: the Co-Lab system provides educationally targeted simulations and virtual experiments of high quality—which can be used collaboratively through the flexible options offered by Cool Modes (joint handwritten annotations, discussion support, etc.).

In this scenario, the MatchMaker communication server with its “Synchronization Tree” [17] provided the

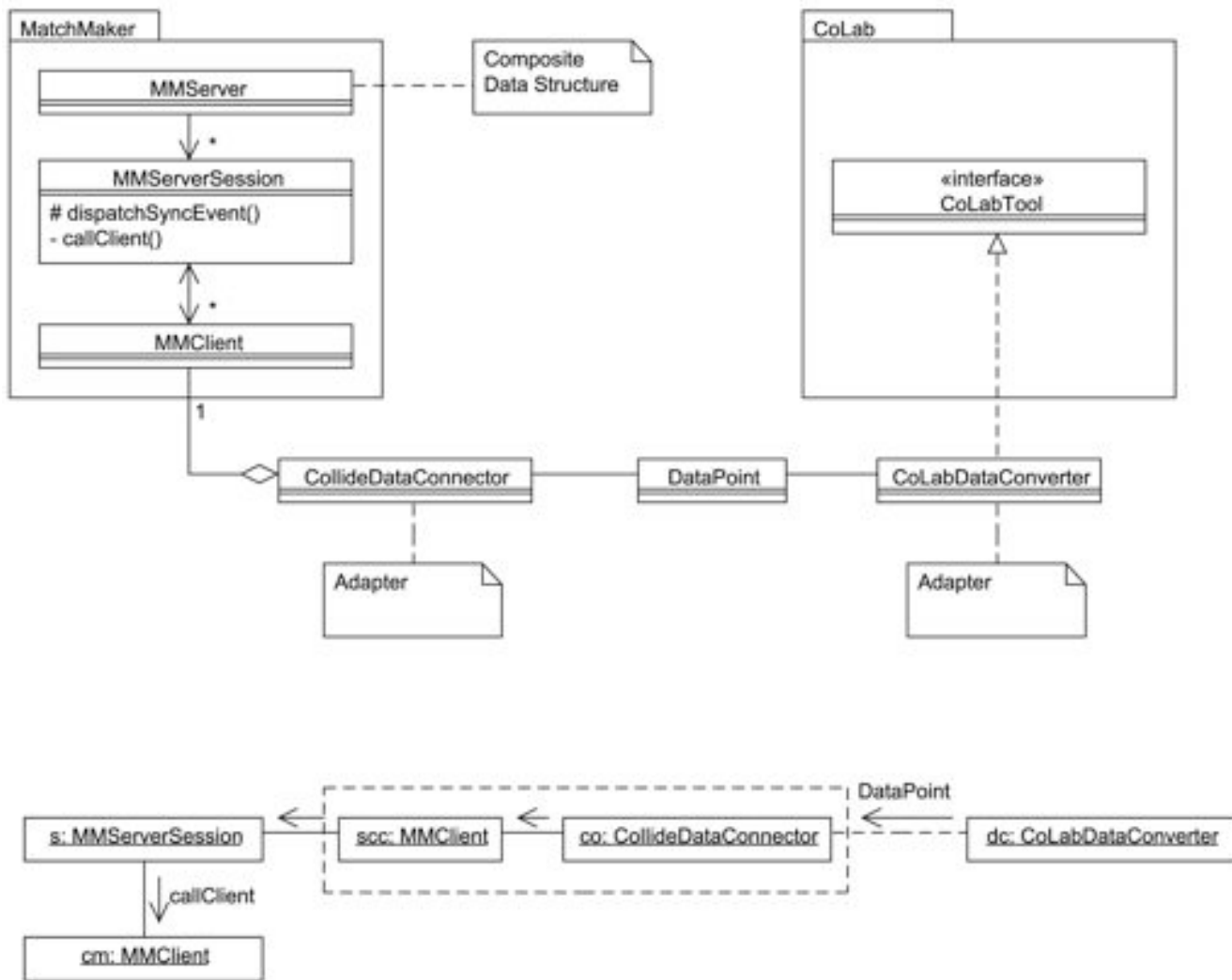


Fig. 4. Combination of the simulation tool Co-Lab with the modeling tool Cool Modes via the MatchMaker composite data structure.

composite data structure. The **adapter** component between the Co-Lab **learning environment** and MatchMaker was implemented as a specific MatchMaker client that uses the API of MatchMaker to update a data table located in the composite data structure whenever new data points are produced in the Co-Lab simulation environment. On the Co-Lab side, only small modifications had to be made in the existing code to relay the data created in Co-Lab to the adapter. The Cool Modes **learning environment** also has a MatchMaker client (the object called `cm` in the lower left of Fig. 4) subscribed to this data table and, consequently, gets informed about any updates on the data; this is then propagated to the Cool Modes application where the data table is visualized graphically and is available for further processing, e.g., as a source for group discussions. Fig. 4 shows the components used and the typical interaction between the components in this integration scenario.

The approach has been implemented with on-the-fly data exchange between the CoLab simulation and Cool Modes without runtime problems, i.e., the Cool Modes data table was filled and displayed directly on each data point produced by CoLab without substantial delays. The first implementation was achieved in a two-day programmers' workshop

including experts in both tools. This first version was refined and code cleaned with minimal effort (less than another day) afterwards to prepare a demonstrator that has been shown and used successfully at several international events.

3.2 Bootstrapping Novice Data

Another example where the Scalable Adapter pattern has been used is the project Bootstrapping Novice Data (BND) [18]. Our goal in this project was to provide tutoring of collaboration within the Cool Modes collaborative software environment [2]. Intelligent tutors have traditionally focused on cognitive, rather than collaborative, skills, so this is a challenging goal. There is currently no method for providing feedback to students in collaborative environments such as Cool Modes, so it has the potential of having high impact.

We devised a process known as *BND*, in which collaborating students' actions are collected and used to develop an example-tracing tutor [19], a special type of tutor developed using the Cognitive Tutor Authoring Tools (CTAT) [20]. Example-tracing tutors provide the same type of support and feedback as the well-known Cognitive Tutors but are driven by example traces rather than production rules. In other words, example-tracing tutors are developed using

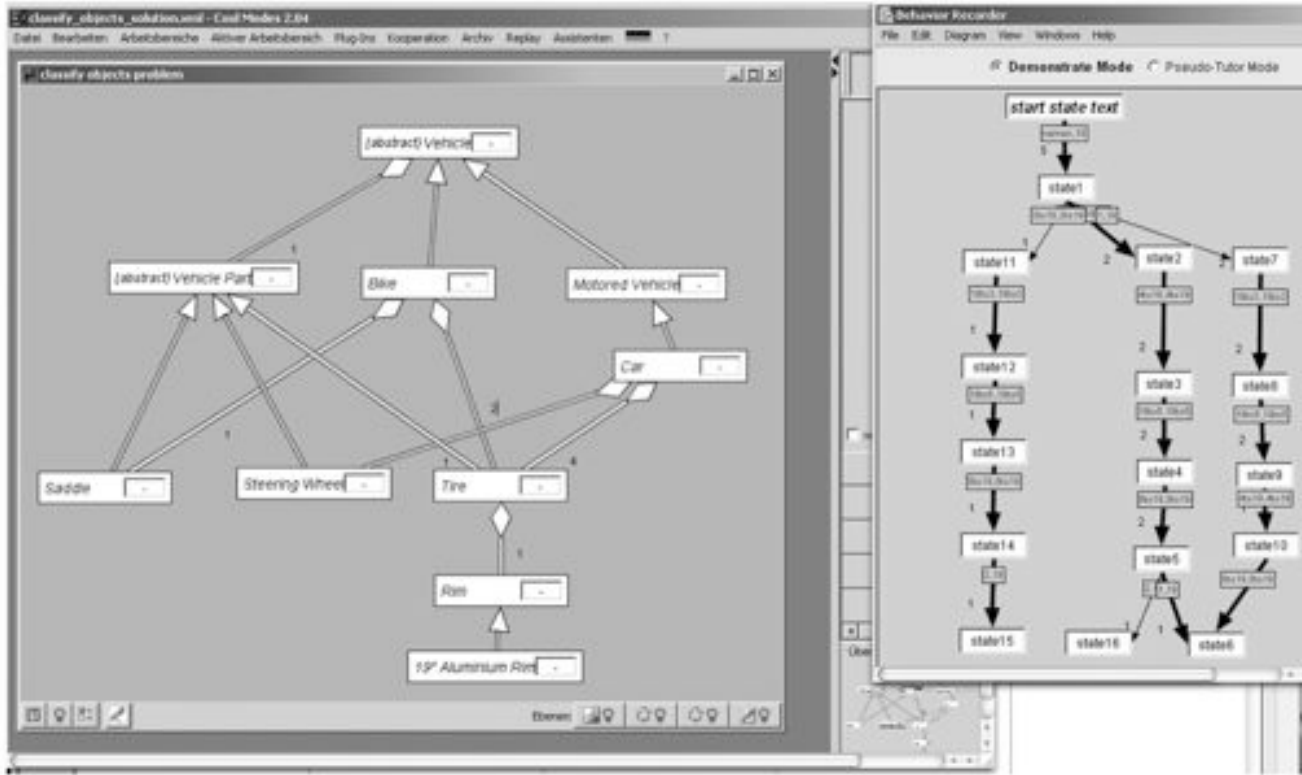


Fig. 5. Screenshot of the collaboration tool Cool Modes with an example-tracing tutor [22].

programming by demonstration [21], an approach that allows authors with no programming skills to build tutors. After example data is provided to CTAT, a tool known as the *Behavior Recorder* (abbreviated BR) records all of these actions and stores them in a structure known as a *behavior graph*. An author, at “author time” then updates the behavior graph with hints and error message feedback, to complete work in developing an example-tracing tutor. Our initial implementation of BND provides a means to directly capture Cool Modes data and feed it into CTAT, as the beginnings of a collaboration example-tracing tutor, but we have not yet created a full-scale tutor to support collaboration.

Fig. 5 shows a collaboration scenario between two students in the domain of UML modeling with Cool Modes. In this problem, the students are given the task of modeling a vehicle and all of its component parts. The actions taken

by the collaborating students (shown on the left side of the figure) are provided to CTAT’s Behavior Recorder (shown on the right side of the figure) first to create the “skeleton” of an example-tracing tutor and later to actually provide real feedback in the context of other students trying to solve the same problem. The states in the behavior graph represent specific sets of actions taken in the Cool Modes tool. After a task is first demonstrated (i.e., at “author time”), an author can provide hints and feedback by annotating the links of the behavior graph with messages. Later, at “student runtime,” these hint and error messages are available to students as they work on the problem.

Fig. 6 shows a diagram of the instantiated runtime architecture (i.e., after the tutor has been fully created) with one tutor component, the MatchMaker providing the **composite data structure** and three clients/students

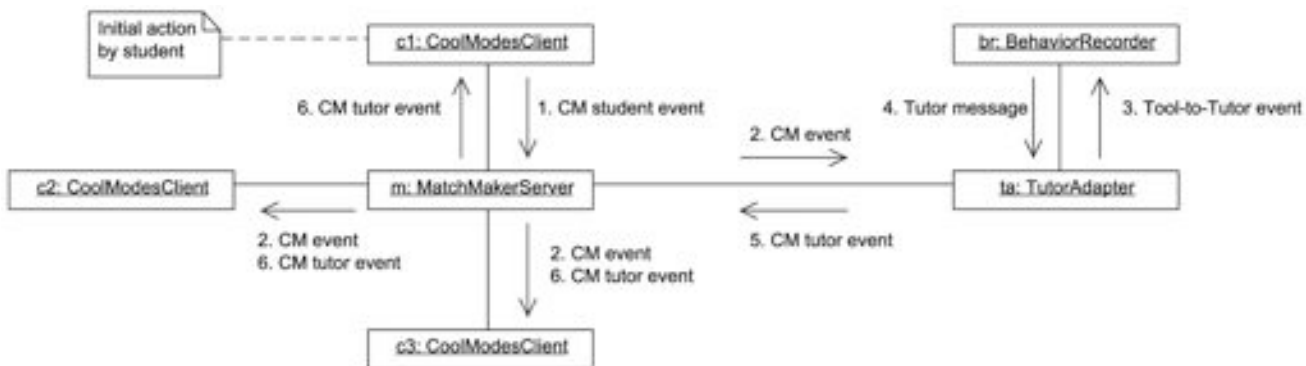


Fig. 6. Combination of the Collaboration Tool Cool Modes with an example-tracing tutor according to the BND approach [22]. The MatchMaker server provides the Composite Data Structure, the Tutor Adapter bridges to the behavior recorder tool.

TABLE 1
XML Fragment of a Cool Modes Logging Event

```
<SyncAction action="objectCreated"
  label="//1/2/5/17"
  objectType="class info.collide.mm.sync.SyncTree"
  time="1100078527389"
  user="Christian">
  <SyncTree label="//1/2/5/17">
    <InheritanceEdgeModel angle="0.7853981633974483">
      <UniqueID>UMLClassModel_002</UniqueID>
      <UniqueID>UMLClassModel_003</UniqueID>
    </InheritanceEdgeModel>
  </SyncTree>
</SyncAction>
```

TABLE 2
XML Fragment of the Event Transformed
into the Behavior Recorder Format

```
<message>
  <properties>
    <MessageType>InterfaceAction</MessageType>
  </properties>
  <Selection>
    <value>InheritanceEdgeModel_
      from_UMLClassModel_002_
      to_UMLClassModel_003</value>
  </Selection>
  <Action><value>objectCreated</value></Action>
  <Input><value>//1/2/5/17</value></Input>
  <User><value>Christian</value></User>
</properties>
</message>
```

collaborating. The component called *Tutor Adapter* is the software **adapter** as of the pattern diagram (Fig. 1). The actions sent to the Behavior Recorder are processed and matched to the behavior graph. Relevant feedback is sent back to the adapter and delivered to the Cool Modes clients, thus providing feedback to the Cool Modes **learning environments** used by the students.

Technically, the user actions in the collaboration tool are mapped to the data format used by the Behavior Recorder, which is also a structured format with well-defined slots: The collaboration actions are represented in an XML format or its corresponding Java object representation containing information about action type, object type, user performing the action, and time stamp. The Behavior Recorder uses the selection-action-input triple [7] extended by information about the user performing the action. The conceptual mapping is straightforward and was realized in different variants, first as an XSL-transformation and later as an on-the-fly integration using our Scalable Adapter pattern.

The two XML fragments (slightly simplified from the exact representation to focus on the essential information transported) given in Tables 1 and 2 show the straightforward mapping of Cool Modes actions to Behavior Recorder actions. The hierarchical structure used in the pattern can be seen in the structured labels (here, //1/2/5/17 for a four-layered hierarchy) that represent the position of the object in the logical hierarchy of the data representation.

This use case demonstrates the expressiveness and flexibility of the Scalable Adapter pattern well: it enables the tutoring component to give feedback based on parts of student actions *in another tool which is completely decoupled from the tutor*. A detailed description of this approach and some details about implementation effort is provided in [22]. While the initial systems used required several person years in development effort, the integration using our scalable adapter design pattern required less than a person week on both ends, i.e., adapting the Behavior Recorder and sending/receiving events from Cool Modes. Changing the domain of tutoring does not produce any additional cost besides the regular effort of providing a domain-specific Cool Modes plug-in, which is needed anyway to learn a new domain with Cool Modes; the adapter is completely generic for XML actions and does not have to be modified at all for new domains.

We performed two exploratory studies in which dyads of students used our integrated BND software to collaborate in solving modeling tasks, such as that in Fig. 5. The indirection of using the adapter did not incur runtime lags, so the capturing of student actions for the example-tracing

tutor was conducted on-the-fly in both collaborative and single user mode. The data collected from these studies led us to identify five dimensions of collaborative and problem-solving behavior that point to the need for abstraction of student actions to better recognize, analyze, and provide feedback on collaboration.

3.3 CoChemEx

A third scenario where the Scalable Adapter pattern was used was in the “CoChemEx” project, a one-year research project within the Pittsburgh Science of Learning Centre. This project addressed a central issue in chemistry education: teaching students to solve problems conceptually rather than by simply applying mathematical equations. Research in chemistry education has shown that students tend to learn and solve problems “algorithmically” but often do not grasp the deeper conceptual aspects of chemistry and reasoning necessary to be more creative and flexible problem solvers [23]. Based on some descriptive evidence in chemistry education research indicating that collaborative activities can improve conceptual learning in chemistry [24], [25], our aim in CoChemEx was to design an educational scenario and collaborative learning system that is able to support students in their acquisition of conceptual knowledge.

The educational scenario in CoChemEx was defined by a team of researchers and practitioners from educational psychology and chemistry education. The scenario consisted of a number of phases that students would go through as they use the system, and a number of tools that are available in these phases [26], [27]. Based on typical steps in scientific discovery learning [28] and additional pilot tests, three phases were finally foreseen for the scenario: 1) experiment planning and design, 2) hypothesis testing, and 3) interpretation and conclusion. The tools required for the phases in the educational scenario included collaborative and individual notepads, chats, argumentation tools, a glossary of chemistry terms, virtual experimentation tools, and functions for checking the correctness of solutions.

None of these single functionalities really requires the development of novel tools: for all the required system functions, there are existing applications available. Thus, in order not to reinvent the wheel (and given the limited funding we had for the one-year project), we decided to implement the CoChemEx system by reusing existing tools as much as possible. In particular, we combined three technologies for the development of the overall system. First, the FreeStyler system [12] was employed for chat and argumentation support, as well as for the glossary. Except

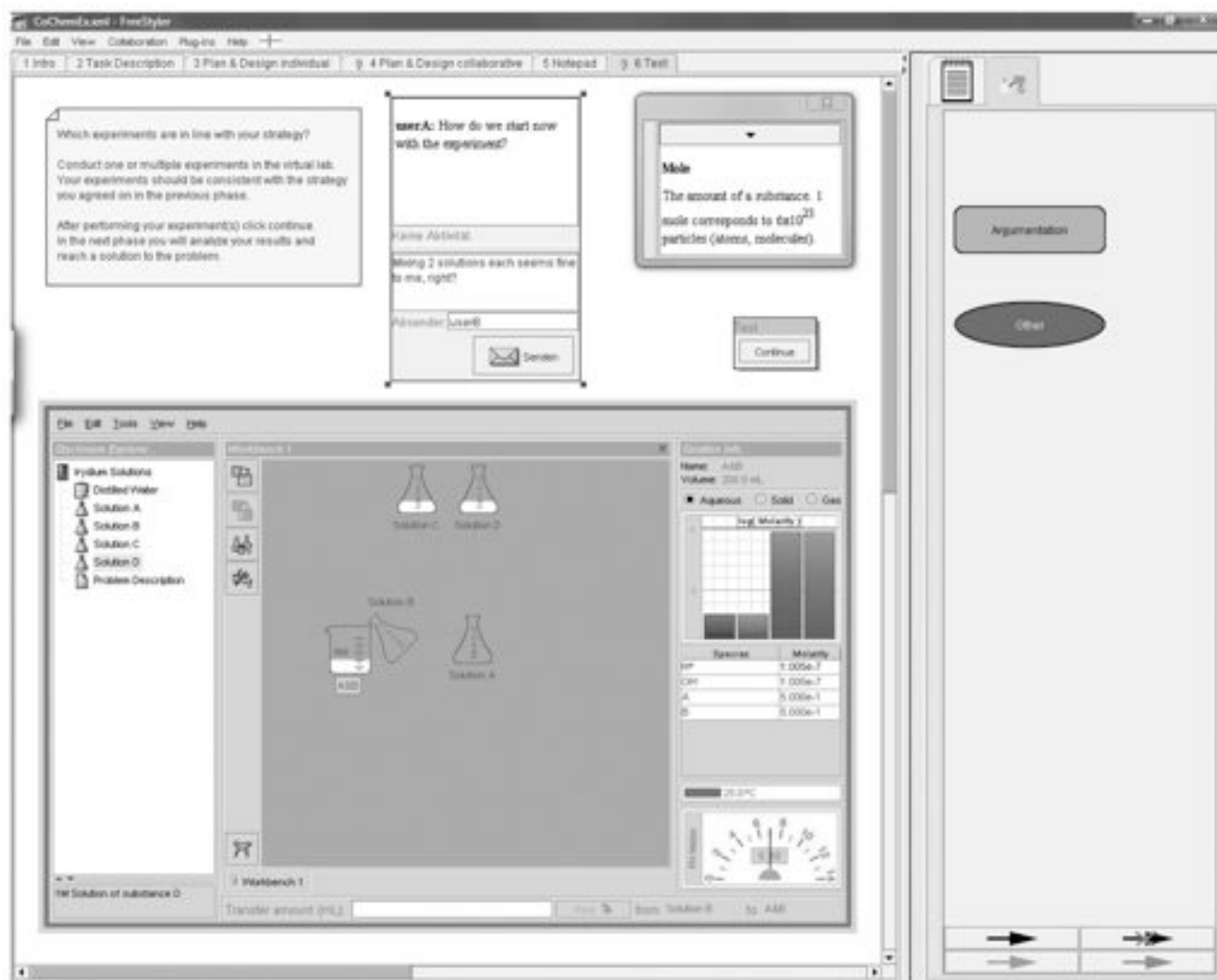


Fig. 7. Collaborative experimentation and chat in the CoChemEx system.

for the user interface, FreeStyler is very similar to the Cool Modes framework described before in this paper: it offers flexible graphical representations that can be used collaboratively over a network. For all but the virtual experimentation and the explicit sequencing of learner actions into phases, the functions of FreeStyler sufficiently covered the requirements of the CoChemEx educational scenario.

The second tool we employed for the construction of the CoChemEx learning system is the virtual laboratory "VLab" [29], a Web-based software tool that emulates a chemistry laboratory and supports chemistry experiments. VLab was developed at Carnegie Mellon University and aims at providing the students with an "authentic" laboratory environment in which they can run experiments to solve chemistry problems much like in a real chemistry lab. The system offers virtual versions of many of the physical items found in a real chemistry laboratory, including chemical solutions, beakers, Bunsen burners, etc. It also includes meters and indicators for real-time feedback on substance characteristics, such as concentration and molarity.

Fig. 7 illustrates the students' view of the full CoChemEx system in the midst of their activity. The single phases of the educational design correspond to the numbered tabs (including *Plan & Design*, *Test/Experiment*, and *Interpret*

Results) on top of the workspace. In the figure, the student *userB* is currently active in the Test/Experiment phase. He is using the VLab (lower left of the workspace) and a chat (top middle) to coordinate his actions with his peer *userA*. Additional tools, such as a glossary (right of the chat) and a graphical argumentation language (at the right) can also be used by the students in specific phases. The workspace also contains the instructions for the students (upper left) and a function to check the correctness of a solution (below glossary).

In this learning scenario, interoperability and data exchange 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 in this scenario. First, it enables the (noncollaborative!) VLab **learning environment** to interact with other VLab instances of the collaborators, thus allowing for collaborative experimentation, and second, it enables the data exchange between the

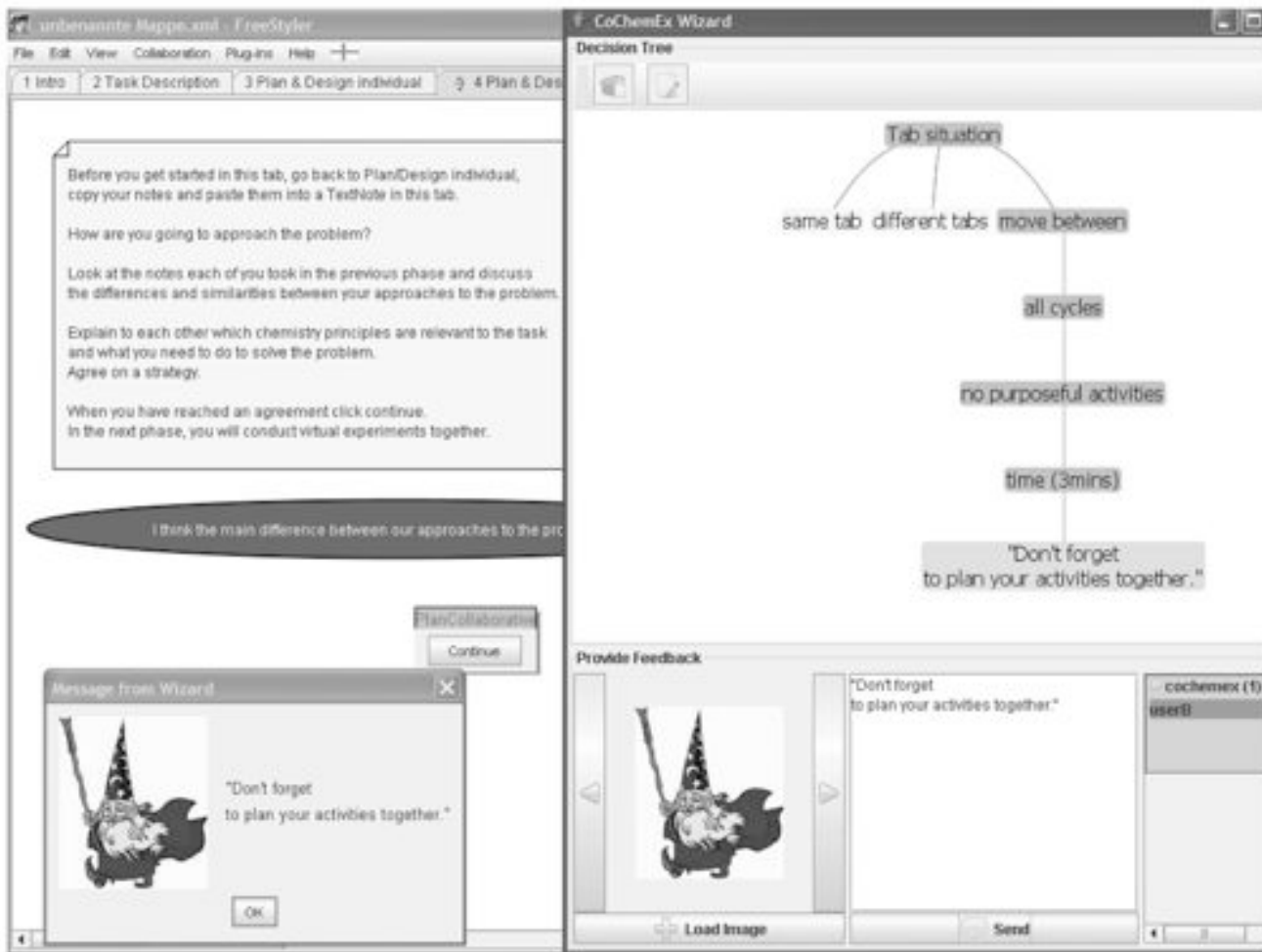


Fig. 8. Integration of the collaboration tool FreeStyler and the Wizard interface.

experimentation functions in VLab and the FreeStyler **learning environments** that are valuable in the experimentation process, such as hypothesis generation or documentation of experiments. Both features are important contributions toward richer learning experiences that only integrated solutions combining various learning tools can provide.

Technically, the interaction between VLab instances is realized by synchronizing their local states each time an instance changes (usually, when a user takes an interface action). The user action is encapsulated as a *VLabEvent* object and published (locally) to all registered system components. The *VLab adapter* is just one of these components. It forwards the *VLabEvent* via MatchMaker to all remote VLab instances taking part in this collaborative session. At the remote end, VLab adapters accept the *VLabEvent* and initiate its replay in the attached VLab instances (the actual replay is a service provided by VLab itself), finally resulting in identical states of all participating VLab instances. This technical solution facilitates also a unified logging of the collaboration, because the VLab actions transmitted via the MatchMaker are merged into the existing logging automatically, which is useful for interaction analysis based on log files.

Collaboration scripts were the third type of technology used in CoChemEx [26]. They were used to structure the

learner collaboration and were formally represented as an IMS Learning Design document, an e-learning standard for educational processes. The IMS LD documents control the available tools for each phase of the learning activity according to an architectural proposal first presented in [30] and the usage of a third-party component for the scripting engine, the CopperCore learning design engine [31]: actions conducted by the learners in the learning tool are propagated to the scripting engine, analyzed there, and subsequently, the learning environment is reconfigured based on the information contained in the script (e.g., a chat system is added for collaborative experimentation). For the configuration of collaborative activities in the learning scenario, we used the composite data structure of the Scalable Adapter pattern and create one specific page/workspace for the collaboration. This is directly created via commands of the scripting engine.

This system-initiated regulation of the learning process is complemented with the possibility of having a trained person supervising the collaboration and giving advice in a Wizard-of-Oz fashion. This wizard component allows the human observer to send text messages and pictorial information directly to an arbitrary set of collaborators.

A prototype implementation has been completed and tested in lab experiments with the script and human wizard support. Fig. 8 shows one student's learning

environment (left) and an active message just recently sent by the human wizard via the wizard interface (right).

Considering all the different components used in our implementation, such as the collaborative modeling tool FreeStyler, the virtual chemistry simulation VLab, the scripting engine CopperCore using the IMS/LD standard, and the wizard component for the human supervisor of the scenario, the technical complexity of the overall CoChemEx system is remarkable. Nevertheless, the actual implementation and assembly of the existing components according to our design approach and the coding of the 'glue' between the components was achieved in few weeks of programming effort (approximately 10 nonfulltime calendar weeks from the end of specification to practical usage, including work on the VLab event subscription and event replay services). This rapid development was made possible by the cost-effective use of existing code, the compliance to well-defined interfaces, and through loose coupling of the components. The Scalable Adapter pattern, described in this paper, was a key piece and success factor for this design. Our future development plans include a crosstool tutoring approach, i.e., a machine tutor whose advice and feedback is based on a combination of data from the different tools. This tutor would be integrated as another learning environment using the Scalable Adapter pattern.

The practical experiments with the CoChemEx software in small-scale lab studies demonstrated the robustness and real-time capabilities of the overall system in authentic situations using intensive collaboration of the student-student-wizard triads. First experimental results are presented in [32] and [27]. In brief, these results suggest that adaptive scripts that react to problematic collaboration behavior (e.g., ignoring requests for explanations) have advantages for fostering effective student collaboration, as compared to nonadaptive scripts.

4 DISCUSSION AND CONCLUSION

In many educational settings, different learning tools can be gainfully used. Communication and data exchange between those tools hold the promise of maximizing the overall learning benefit. In addition, teachers or researchers may want to experiment with ideas about how different learning processes can work together and be supported through technology. To meet these needs, the critical issue is to make learning tools more interoperable.

This paper presented an architectural design pattern for the integration of different learning environments by means of a flexible and scalable data exchange mechanism using adapter and composition techniques. With minor modifications to existing tools, this microarchitecture can be reused by developers of learning support systems in other contexts, provided that a conceptual mapping between (arbitrary parts of) the tool data can be defined (i.e., which tool uses what data for which purposes?). For existing applications that use proprietary and ill-structured data the gap for a conceptual mapping to a composite data structure is larger and the development effort will be higher; additionally, the programmers of the original learning environment will very likely have to make the implementation of the adapter themselves if the data is not well documented and understandable for somebody used to our framework.

As for the runtime qualities and flexibility of our technical solutions, we got good results: all three scenarios described here were able to work with on-the-fly interoperability without any perceivable lags and problems from the users' perspective. The dynamic addition of learning environments was also possible, such as using a different CoLab simulation than the usual WaterTank with Cool Modes or additional collaborators for the BND approach with dynamic collaboration groups. The only thing required is that the new component uses a data format that other components have subscribed for in the composite data structure.

While the three example cases presented in this paper use some similarly targeted learning environments and particularly the same tool for the composite data structure, there are other initiatives under way that follow the same ideas, such as the European Research Team CIEL in the Kaleidoscope Network of Excellence [33] and the SAIL initiative [10]. All these approaches strive to overcome the limitations of specific learning tools and expand the scope of learning activities toward longer and more elaborate experiences, using the best tools available for each specific subtask and phase. These learning experiences can be individual activities, but the proposed technical design is also ideally suited to conduct at least parts of the learning process collaboratively. One challenge that is still in front of us is that for every learning environment, a specific adapter must be defined. Also, the shared composite data structure is currently still scenario specific, i.e., demands a basic semantic grounding between all the participating learning tools, and depends on the tools that are connected and the purposes of this connection. If tools use different terminology for their data and expect the same vocabulary from the incoming data, the interoperation is inhibited, currently, because of ontological differences.

While this may be acceptable for many educational scenarios (especially if the programming effort to create the adapter components and the data structure is low), mechanisms for a more flexible declarative configuration of the common data structure by explicit specification of structure levels seem preferable to hard coding all dependencies. One promising approach for that end is to define a conceptual metamodel that bridges between previously existing proprietary data formats and to create the adapters automatically from a description mapping the conceptual model to a platform-specific model (PSM), similarly to the model-driven-architecture approach (<http://www.omg.org/mda/>). Thus, the task of creating an adapter would become a modeling task, not a low-level and potentially erroneous coding task. While creating a general-enough metamodel may be a challenging task, this approach would enable also designers and practitioners to integrate learning tools into a unified scenario. To succeed in a proposal for such a modeling approach, a suitable modeling language is desirable that is capable of representing existing data formats and mapping between them; an important factor for such an initiative will be agreements and standardization between the members of the scientific community and their respective tools. International networks and developer communities, such as that mentioned above, are a starting point to get results on this.

ACKNOWLEDGMENTS

The principles manifested in this design pattern proposal have been discussed by the authors with numerous colleagues in a variety of integration projects, such as the NetCoIL scientific network for Collaborative Inquiry Learning (funded by the German DFG), the BND approach for integration of tutors with collaboration tools, and in the CoChemEx project for loose integration of a Virtual Chemistry Lab with a Collaboration and Argumentation tool. The CoChemEx project was conducted within the Pittsburgh Science of Learning Center (PSLC) in the US, funded by US National Science Foundation (NSF) Grant 0354420. Many thanks to all the participants in the discussions. The authors especially thank all of their numerous collaborators in the CoChemEx project for their great support. They express a particular gratitude to Nikol Rummel for the contributions of her and her team to this project. A short version of this paper appeared in the Proceedings of the Conference on Intelligent Tutoring Systems 2008.

REFERENCES

- [1] K.R. Koedinger, J.R. Anderson, W.H. Hadley, and M.A. Mark, "Intelligent Tutoring Goes to School in the Big City," *Int'l J. Artificial Intelligence in Education*, vol. 8, pp. 30-43, 1997.
- [2] N. Pinkwart, "Collaborative Modeling in Graph Based Environments," PhD dissertation, Universität Duisburg-Essen, 2005.
- [3] W. van Joolingen, A. Lazonder, T. de Jong, E. Savelsbergh, and S. Manlove, "Co-Lab: Research and Development of an Online Learning Environment for Collaborative Scientific Discovery Learning," *Computers in Human Behavior*, vol. 21, pp. 671-688, 2005.
- [4] M. Jansen, M. Oelinger, K. Hoeksema, and H. Hoppe, "Exploring the Use of Mobile Devices to Facilitate Educational Interoperability Around Digitally Enhanced Experiments," *Proc. Second IEEE Int'l Workshop Wireless and Mobile Technologies in Education (WMTE '04)*, J. Roschelle, T.-W. Chan, Kinshuk, and S.J.H. Yang, eds., pp. 83-90, 2004.
- [5] J. Roschelle, C. DiGiano, A. Repenning, J. Phillips, N. Jackiw, and D. Suthers, "Developing Educational Software Components," *Computer*, vol. 32, no. 9, pp. 50-58, 1999.
- [6] J. Roschelle, C. DiGiano, and M. Chung, "Reusability and Interoperability of Tools for Mathematics Learning: Lessons from the ESCOT Project," *Proc. Int'l Congress on Intelligent Systems and Applications (ISA '00)*, F. Haghdy and F. Kurfess, eds., pp. 664-669, 2000.
- [7] S. Ritter and K. Koedinger, "An Architecture for Plug-In Tutor Agents," *Int'l J. Artificial Intelligence in Education*, vol. 7, pp. 315-347, 1996.
- [8] E. Walker, K. Koedinger, B. McLaren, and N. Rummel, "Cognitive Tutors as Research Platforms: Extending an Established Tutoring System for Collaborative and Metacognitive Experimentation," *Proc. Intelligent Tutoring Systems (ITS '06)*, M. Ikeda, K.D. Ashley, and T.-W. Chan, eds., pp. 207-216, 2006.
- [9] L. Bollen, A. Harrer, U. Hoppe, and W. van Joolingen, "A Broker Architecture for Integration of Heterogeneous Applications for Inquiry Learning," *Proc. Seventh IEEE Int'l Conf. Advanced Learning Technologies (ICALT '07)*, J. Spector, D. Sampson, T. Okamoto, Kinshuk, S. Cerri, M. Ueno, and A. Kasihara, eds., pp. 15-17, 2007.
- [10] SAIL, *Scalable Architecture for Interactive Learning Project Home Page*, <http://docs.telscenter.org/display/SAIL/Home>, 2007.
- [11] WISE, *Web-Based Inquiry Science Environment, Supported by the National Science Foundation NSF*, <http://wise.berkeley.edu>, Aug. 2005.
- [12] S. Zeini, N. Malzahn, and U. Hoppe, "Kooperationswerkzeuge IM Kontext Virtualisierter Arbeit," *Virtuelle Organisationen und neue Medien, Gemeinschaften in Neuen Medien (GeNeMe)*, 2004.
- [13] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1995.
- [14] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. John Wiley & Sons, 1996.
- [15] N. Carrier and D. Gelernter, "Linda in Context," *Comm. ACM*, vol. 32, no. 4, pp. 444-458, 1989.
- [16] P. Brusilovsky, "KnowledgeTree: A Distributed Architecture for Adaptive E-learning," *Proc. 13th Int'l World Wide Web Conf. Alternate Track Papers & Posters (WWW Alt. '04)*, pp. 104-113, 2004.
- [17] M. Jansen, "Matchmaker—A Framework to Support Collaborative Java Applications," *Proc. 11th Conf. Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies*, U. Hoppe, F. Verdejo, and J. Kay, eds., pp. 529-530, 2003.
- [18] A. Harrer, B. McLaren, E. Walker, L. Bollen, and J. Sewall, "Collaboration and Cognitive Tutoring: Integration, Empirical Results, and Future Directions," *Artificial Intelligence in Education—Supporting Learning through Intelligent and Socially Informed Technology*, Frontiers in Artificial Intelligence and Applications, C.-K. Looi, G. McCalla, B. Bredeweg, and J. Breuker, eds., vol. 125, pp. 266-273, IOS Press, 2005.
- [19] K. Koedinger, V. Alevan, N. Heffernan, B.M. McLaren, and M. Hockenberry, "Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behaviour by Demonstration," *Proc. Intelligent Tutoring Systems (ITS)*, 2004.
- [20] V. Alevan, B.M. McLaren, J. Sewall, and K. Koedinger, "Example-Tracing Tutors: A New Paradigm for Intelligent Tutoring Systems," *Int'l J. Artificial Intelligence in Education*, special issue on authoring systems for ITSs, in press, 2008.
- [21] H. Lieberman, *Your Wish Is My Command: Programming by Example*. Morgan Kaufmann, 2001.
- [22] A. Harrer, B. McLaren, E. Walker, L. Bollen, and J. Sewall, "Creating Cognitive Tutors for Collaborative Learning: Steps toward Realization," *User Modeling and User-Adapted Interaction: The J. Personalization Research*, special issue on user modeling to support groups, communities, and collaboration, vol. 16, pp. 175-209, 2006.
- [23] D.L. Gabel, R.D. Sherwood, and L. Enochs, "Problem-Solving Skills of High School Chemistry Students," *J. Research in Science Teaching*, vol. 21, no. 2, pp. 221-233, 1984.
- [24] J.L. Fasching and B.L. Erickson, "Group Discussions in the Chemistry Classroom and the Problem-Solving Skills of Students," *J. Chemical Education*, vol. 62, pp. 842-848, 1985.
- [25] R.B. Kozma, "The Use of Multiple Representations and the Social Construction of Understanding in Chemistry," *Innovations in Science and Math. Education: Advanced Designs for Technologies of Learning*, M.J.R. Kozma, ed., pp. 11-46, Erlbaum, 2008.
- [26] B.M. McLaren, N. Rummel, D. Tsovaltzi, I. Braun, O. Scheuer, A. Harrer, and N. Pinkwart, "The CoChemEx Project: Conceptual Chemistry Learning through Experimentation and Adaptive Collaboration," *Proc. Workshop Emerging Technologies for Inquiry Based Learning in Science at AIED*, submitted, 2007.
- [27] D. Tsovaltzi, N. Rummel, N. Pinkwart, A. Harrer, O. Scheuer, I. Braun, and B.M. McLaren, "Cochemex: Supporting Conceptual Chemistry Learning via Computer-Mediated Collaboration Scripts," *Proc. European Conf. Technology Enhanced Learning (EC-TEL '08)*, 2008.
- [28] T. de Jong and W. van Joolingen, "Scientific Discovery Learning with Computer Simulations of Conceptual Domains," *Rev. Educational Research*, vol. 68, no. 2, pp. 179-201, 1998.
- [29] D. Yaron, K. Evans, and M. Karabinos, "Scenes and Labs Supporting Online Chemistry," *Proc. 83rd Ann. AERA Nat'l Conf.*, 2003.
- [30] A. Harrer, N. Malzahn, K. Hoeksema, and U. Hoppe, "Learning Design Engines as Remote Control to Learning Support Environments," *J. Interactive Media in Education*, special issue on advances in learning design, 2005.
- [31] CopperCore, *CopperCore, the IMS Learning Design Engine*, <http://www.coppercore.org>, May 2005.
- [32] D. Tsovaltzi, B.M. McLaren, N. Rummel, O. Scheuer, A. Harrer, N. Pinkwart, and I. Braun, "Cochemex: Supporting Conceptual Chemistry Learning via Computer-Mediated Collaborative Scripts," *Proc. Intelligent Tutoring Systems (ITS)*, 2008.
- [33] CIEL, *Collaborative Inquiry and Experiential Learning Project Home Page*, <http://ciel.gw.utwente.nl/>, 2007.



Andreas Harrer received the PhD degree in computer science from the Technical University of Munich with a doctoral thesis in the area of CSCL (intelligent support of collaborative learning interactions). Since Winter 2007, he has been an associate professor in informatics (computer science) at Catholic University Eichstätt-Ingolstadt. He was with the Collide Group, University Duisburg-Essen, from 2002 to 2007, with a research line of analysis and intelligent support

of group learning as well as software engineering principles for educational systems. He was a speaker of the special interest group "Artificial Intelligence and Education," the European Research Team "Computer-based Analysis and Visualization of Collaborative Learning Activities." He is a member of the steering group of the European Research Team "Computer-Supported Scripting of Interaction in Collaborative Learning Environments" in the "Kaleidoscope" Network of Excellence and has been involved in numerous international projects and collaborations.



Niels Pinkwart received the PhD degree from the University of Duisburg-Essen in 2005, with a thesis on "Collaborative Modeling in Graph-Based Environments" and a postdoctoral degree from Carnegie Mellon University. He is currently with the Clausthal University of Technology, where he currently leads a research group that investigates collaborative systems with their software architectures, design patterns, user interfaces, and usage by humans. A specific

focus of his research is set on applications in the domain of educational technology, particularly on distributed and collaborative software systems that provide intelligent support to students in order to help them learning. In this field, he has published more than 60 referred papers in national and international conference proceedings and journals.



Bruce M. McLaren is currently with Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, Germany, as a senior researcher, and with Carnegie Mellon University (CMU), Pittsburgh, as a systems scientist. In both institutions, he is engaged in research of human learning and educational technology. He does his research within the Competence Center for e-Learning at DFKI and within the Pittsburgh Science of Learning Center (PSLC) at

CMU. At CMU, he comanages a team of eight programmers and research associates in the development and enhancement of the Cognitive Tutor Authoring Tools (CTAT). His research interests and experience include educational technology, collaborative learning, intelligent tutoring, case-based reasoning, and artificial intelligence. He has more than 50 publications in journals, conference proceedings, workshops, and symposiums.



Oliver Scheuer received the Dipl-Inform (master's) degree from the Saarland University in 2007. He is currently a PhD student in the Competence Center for e-Learning, Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken, Germany. His research interests include the application of machine learning, online data analysis, and feedback techniques to collaborative learning systems.

▷ **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.**