Accepted for 2006 publication in User Modeling and User-Adapted Interaction:
The Journal of Personalization Research

Creating Cognitive Tutors for Collaborative Learning: Steps Toward Realization

Andreas HARRER[*], Bruce M. MCLAREN[+], Erin WALKER[+], Lars BOLLEN[*], and Jonathan SEWALL[+]

[+]*Carnegie Mellon University, Pittsburgh, PA, USA*

bmclaren@cs.cmu.edu, erinwalk@andrew.cmu.edu, sewall@cs.cmu.edu

[*]*University Duisburg-Essen, Duisburg, Germany*

harrer@collide.info, bollen@collide.info

Abstract. Our long-term research goal is to provide cognitive tutoring of collaboration within a collaborative software environment. This is a challenging goal, as intelligent tutors have traditionally focused on cognitive skills, rather than on the skills necessary to collaborate successfully. In this paper, we describe progress we have made toward this goal. Our first step was to devise a process known as *bootstrapping novice data (BND)*, in which student problem-solving actions are collected and used to begin the development of a tutor. Next, we implemented BND by integrating a collaborative software tool, Cool Modes, with software designed to develop cognitive tutors (i.e., the Cognitive Tutor Authoring Tools, or CTAT). Our initial implementation of BND provides a means to directly capture data as a foundation for a collaboration tutor but does not yet fully support tutoring. Our next step was to perform two exploratory studies in which dyads of students used our integrated BND software to collaborate in solving modelling tasks. 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. We also interviewed a domain expert who provided evidence for the advantage of bootstrapping over manual creation of a collaboration tutor. We discuss plans to use these analyses to inform and extend our tools so that we can eventually reach our goal of tutoring collaboration.

Keywords: Intelligent Tutoring Systems; Collaborative Learning; Collaboration Modelling; Action-Based Analysis

This paper (or a similar version) is not currently under review by a journal or conference, nor will it be submitted to such within the next three months.

INTRODUCTION

Intelligent Tutoring Systems (ITS) have long been used to provide one-on-one (machine-to-student) instruction [e.g., (Polson & Richardson, 1988); (Wenger, 1987)]. Cognitive tutors, a particular type of intelligent tutor that supports "guided learning by doing" (Anderson, Corbett, Koedinger, & Pelletier, 1995), have been shown to improve cognitive learning in domains like algebra and geometry by approximately one standard deviation over traditional classroom instruction (Koedinger, Anderson, Hadley, & Mark, 1997). So far, however, cognitive tutors have been used only for one-on-one instruction—a computer tutor assisting a single student – and primarily to support cognitive learning. We, on the other hand, seek to determine whether a cognitive tutoring approach, and, more specifically, the

type of tutoring approach taken by Anderson, Koedinger, and their colleagues, can support and improve collaboration as well. In addition, other research questions we aspire to answer with our work are:

- To what extent can a cognitive tutor evaluate collaboration by evaluating problem-solving actions and ignoring specific dialogue between collaborators?
- What types of collaborative problems are most amenable to a cognitive tutoring approach?
- Should feedback for collaborating students be immediate, as in traditional cognitive tutors, or is a delayed-feedback approach more appropriate for a collaborative environment?

Collaboration is recognized as an important forum for learning, as discussed in a recent book on how people learn (Bransford, Brown & Cocking, 2000), and learning research has demonstrated its potential for improving students' problem solving and learning (Slavin, 1992; Johnson & Johnson, 1990). However, collaboration is a complex process, not as constrained as individual learning. There have been steps toward providing tutoring in a collaborative environment (e.g., Lesgold, Katz, Greenberg, Hughes, & Eggan, 1992; Vizcaino, 2005; Goodman, Linton, Gaimari, Hitzeman, Ross & Zarrella, 2005; Soller & Lesgold, 2003; Mühlenbrock & Hoppe, 2001; Constantino-González & Suthers, 2002; Constantino-González, Suthers & Escamilla de los Santos, 2003), but many difficult challenges remain. Development effort, for one, presents a major barrier. For the simpler single-student case, estimates are in the range of 100-1000 hours of development time per each hour of instruction (Murray, 1999); (Anderson *et al*., 1995) estimated 200 hours for the Algebra Cognitive Tutor. In the collaborative environment, analysis of learner behavior is much more difficult, in part because the space of possible actions and conversations among collaborating users is huge for even the simplest of problems. With the relatively small number of existing tutors for collaborative support, and the conceptual problems still to answer, reliable estimates for development effort are not readily available, but it can be assumed that the effort is at least equal to, and likely much larger than, that of the single-user case. This makes it particularly attractive to leverage *live* user data as the basis for creating and fine-tuning the tutors, as we attempt to do in our work.

Our work leverages the principle of cost-effective *example-tracing tutors,*[1] which are developed by demonstration rather than by complex rule writing (Koedinger, Aleven, Heffernan, McLaren, & Hockenberry, 2004). Furthermore, instead of having an expert create the tutor by demonstration, we use *real* data gained from collaborating groups solving problems as the basis for a tutor. This automatic capturing of data will be complemented by manual and/or automatic analysis of that data. The ultimate desired result is a "cognitive" model representing specific successful and unsuccessful modes of collaboration.

To take a step toward addressing the questions raised above, we have developed an approach called *bootstrapping novice data* (BND) in which groups of students attempt to solve problems with a computer-based collaborative tool. While the students work, the system records their actions in a network representation that combines all collaborating groups' solutions into a single graph that can be used for analysis and, eventually, as the basis for a tutor. We have begun experimentation with the BND approach by integrating a collaborative work environment and a cognitive tutoring tool (McLaren, Bollen, Walker, Harrer, & Sewall, 2005; Harrer, McLaren, Walker, Bollen, & Sewall, 2005; McLaren, Koedinger, Schneider, Harrer, & Bollen, 2004a; 2004b). The collaborative modelling tool we used in our initial implementation is Cool Modes (Collaborative Open Learning and MODEling System) (Pinkwart, 2003; Pinkwart, Hoppe, Bollen & Fuhlrott, 2002), while the tutoring software is the Cognitive Tutor Authoring Tools (CTAT) (Aleven *et al*., 2006). There are two stages to our plans – data collection/analysis and tutor creation – the first of which we are well on the way to achieving. We have captured and analyzed data from live collaboration so that we can better understand how a cognitive tutor might use that data to diagnose and support student actions in a collaborative environment. We eventually want to directly use the data we collect as the basis for the cognitive tutor model of collaboration.

[1] In (Koedinger *et al*., 2004) these specialized tutors based on examples of actual problem solutions were referred as "Pseudo Tutors." However, because we believe this name does not aptly describe the tutors, we have since renamed them "example-tracing tutors" (Aleven, McLaren, Sewall & Koedinger, 2006; Aleven, Sewall, McLaren & Koedinger, 2006).

In this paper, we review relevant past work, introduce the concept of bootstrapping novice data, describe how we have implemented the BND methodology, present empirical work that explores a particular type of collaborative problem and tests the BND approach, discuss an interview we conducted with a domain expert who analyzed the trade-offs between bootstrapping and manual creation of a tutor model, and, finally, present our ideas for extending BND both to improve analysis and to lead to our ultimate goal of providing tutoring in a collaborative environment.

RELATED RESEARCH

We are not the first to attempt to provide intelligent tutoring and/or coaching in an online collaborative environment. On the other hand, this is certainly a nascent area of research with many interesting research issues remaining open, such as the ones stated above that are particular to cognitive tutoring. In this section, we discuss some of the past, related work and compare and contrast this research to our own.

One of the seminal projects to suggest the combination of intelligent tutoring and collaborative learning was the work of Lesgold and colleagues (1992). This research was an extension of the well-known Sherlock project in which Air Force mechanics were tutored (one-on-one, machine-to-human) in diagnosing failures in complex electronic systems (Lesgold, Lajoie, Bunzo & Eggan, 1993). An important component of the Sherlock system was its "reflective follow-up" module, which presented an abstracted replay of a solution after a problem solving session and prompted the student to ask Sherlock questions about that solution. Sandra Katz, one of the members of Lesgold's team, saw this phase of Sherlock as a particularly good opportunity for students to collaborate with one another or with Sherlock. The idea behind this work was not so much to directly tutor collaboration as it occurred, but rather to provide a framework to allow students (or a student and Sherlock) to critique one another and explain problem solving. In this sense, it is distinct from our research, as well as much of the research described below, which is focused more on evaluating and supporting live, computer-mediated collaboration. In addition, the collaborative features of the Sherlock system were only designed as extensions to the individual tutoring system, but they were not implemented or evaluated.

Some of the more recent efforts that have focused on live, computer-mediated collaboration and have been implemented focus on *both* student actions and collaborative conversations, in contrast to our approach which focuses on student actions while ignoring language content. The HabiPro system of Vizcaino (2005), for instance, analyses both problem solving and chat actions via a "simulated student" that coaches students as they collaboratively learn how to write a computer program online. The "live" student collaborators are not informed that HabiPro is a pedagogical agent; the program provides hints and feedback in simple natural language statements. The hypothesis here is that peer coaching may provide more welcome support than a teacher or tutor. The program focuses on and intervenes to correct three particular problems: off-topic conversations, passive students, and learning issues. To identify off-topic conversation, the system matches the language of chat messages against keyword databases containing valid domain words and "playful" (i.e., off-topic) words. An analysis of the frequency and density of contributions is used to identify passive students, while mistake frequency and answer "closeness" are used to identify learning problems. A study by Vizcaino involving 22 student dyads, one-half of which collaborated with the simulated student and one-half of which did not, demonstrated that HabiPro is moderately successful in correcting the three types of problems. The HabiPro program is a step beyond our work in development in that it already provides live collaborative tutoring, but it relies on pre-defined and fixed measures of correctness to advise students, rather than using live student data to directly build the tutor model, as we are attempting to do.

Another research project that employs the concept of a peer pedagogical agent and evaluates collaborative dialogue is that of Goodman and colleagues (2005). The pedagogical agent in their web-based collaborative system follows a discussion between students and provides feedback when it detects a problem, such as one student dominating the discussion. Their primary (but not sole) focus is on dialogue acts (Searle, 1969) and other characteristics of language communication, rather than on problem-solving

actions, as in our work. They found that neural networks worked better than Hidden Markov Models (HMMs) in evaluating communication and predicting when intervention might be helpful. Soller and Lesgold (2003) also focus on collaborative conversation and the use of HMMs as an analysis tool. They investigated knowledge sharing and knowledge construction between students as those students tried to solve object-modelling problems. Their system identifies speech acts through the technique of "sentence openers" in which collaborators communicate with one another by selecting from a small and finite set of possible ways to start a sentence (e.g., "Let me explain it this way…", "I agree because …") (McManus & Aiken, 1995). Both of these projects, unlike the HabiPro work, share our goal of leveraging live student data for the purpose of tutoring. Their neural networks and HMMs are trained on "good" and "bad" collaboration for the purpose of identifying these characteristics in live collaboration. Perhaps our work could benefit from experimenting with some of the advanced analysis techniques they have employed, although we would apply these techniques to student actions rather than speech acts, which is their primary focus.

The action-based collaboration analysis of Mühlenbrock and Hoppe (2001), which focuses primarily on actions in a graphical workspace rather than on dialogue content between participants, is closer to our approach. They use rule-based recognition of characteristic sequences of problem-related user actions in a shared workspace – for instance, in a graphical game in which puzzle pieces must be moved. Since this approach was developed primarily for collaborative face-to-face scenarios, where language communication between the students occurs outside of computer mediation, it focuses exclusively on actions and not on the discourse/dialogue between collaborators. While our initial approach similarly places emphasis on actions, our project differs in that we are interested in collaboration achieved completely in a computer-mediated way (i.e., dyads collaborating strictly over computers). At least initially, the only dialogue analysis we attempt involves identifying the fact that students are communicating via "chat actions." Mühlenbrock and Hoppe's worked is focused more on analysis techniques and less on tutoring or coaching of collaboration, as we are in our work.

Like the Mühlenbrock and Hoppe work, as well as our own, COLER (Constantino-González & Suthers, 2002) focuses exclusively on student problem-solving actions, ignoring the content of language communication between collaborating students. The task of COLER's collaborating students is to build database Entity/Relationship diagrams. A shared workspace contains the amalgamated solution of all the collaborators, and each student has his or her own individual solution workspace. Based on socio-cognitive conflict theory, COLER operates by finding structural differences between the students' evolving individual and group solutions and identifying opportunities to suggest actions based on those differences. For instance, the coach in COLER might notice that a student, George, has correctly defined a relationship in his individual diagram that is missing from the shared diagram and suggest that he volunteer this information to the group. COLER also evaluates collaborative features of student interaction, such as lack of participation by a particular student. As with HabiPro, a key difference of COLER from our work is the fact that it is not based on a model built from live student data. Rather, it operates by dynamically analyzing differences generated through comparison of the individual and group solutions.

Our approach contrasts with all of this previous work in its task-independent nature. The Cool Modes graphical modelling environment has many different plug-ins, supporting a variety of problem-solving tasks, and our BND approach is agnostic to the particular type of graphical modelling task undertaken. While supporting collaboration within different tasks has not yet been achieved, we *have* demonstrated (and published) bootstrapping in several graphical modelling tasks (McLaren *et al*., 2004a; 2004b; McLaren *et al*., 2005; Harrer *et al*., 2005). In principle, some of the prior work might be applicable to different problems, but all of the prior systems discussed above were demonstrated in a single task domain.

BUILDING A TUTOR DIRECTLY FROM STUDENT DATA: BOOTSTRAPPING NOVICE DATA

In the bootstrapping novice data approach we have developed, groups of collaborating users generate (possibly different) correct and faulty solutions to the same problem. The logs of the different collaborating groups are directly translated into a single representation of problem solving, in the form of a graph whose edges represent student actions. The graph starts initially empty. The BND approach not only generates examples of *actual* correct and buggy paths taken by students, but also provides another important piece of information: traversal frequencies of those paths across all of the collaborating groups. The same actions taken by different collaborating dyads are identified with the same edges in the graph: as these actions are recognized, they contribute to incrementing the edge traversal frequencies. After multiple groups have generated data, the graph contains the actions of all student groups and reveals the frequency of common paths, both correct and incorrect. In the final step of the BND process, the tutor author manually updates the problem-solving graph by marking buggy paths, adding hints and bug messages, and adding skills to graph edges. At this stage, the model is ready to be used for tutoring.

Use of novice data in this manner can help avoid the so-called "expert blind spot" problem, in which experienced problem-solvers and teachers fail to identify the common errors of novice students (Nathan, Koedinger & Alibali, 2001). The edge traversal counts are good indicators of which of the correct solution paths might be considered primary, which secondary, and the counts along incorrect paths provide data to show which errors occur frequently enough to merit specific buggy messages. The traversal counts can also help authors identify slips and careless errors (e.g., accidental item selections): an edge with a traversal count of 1, as compared to much higher counts on alternative edges, may indicate that an accidental action was taken by a student. Such edges can be deleted from the graph.

Instead of authors building tutors from scratch, relying on their own experience or incorporating student data "by hand," as in traditional ITS development, they can semi-automatically leverage the empirical data of a wide range of students engaged in actual problem-solving activity. A tutor author can then create a tutor directly from the graph by labelling edges with hints and buggy messages. This approach contrasts markedly with the usual ITS development method in which an expert author first models "expert" problem solutions. In our approach, the student novices create the initial solutions. The expert's judgment as to which novice solutions are correct is, of course, critical to creating a final version of the tutor. It may also be the case that an expert will have to augment a model by demonstrating a correct solution (or solutions), if the student novices fail to generate any correct solution paths. But the critical aspect of BND is how it directly captures and encodes incorrect and inefficient novice solutions, information that is invaluable in building a full ITS for a collaborative system.

There are two key advantages to the BND approach. First, direct capture of student data for use in tutor building is a novel and powerful idea. While student data has been used to *guide* cognitive tutor design (Koedinger & Terao, 2002) and *tune* tutor parameters (Corbett, McLaughlin & Scarpinatto, 2000), it has not previously been used *directly* as input in creating a cognitive tutor. Recently educational data mining approaches have been used to understand student-tutor interaction (Heiner, Beck & Mostow, 2004) and individual learning (Merceron & Yacef, 2005). Our approach, as well as that of (Goodman *et al*., 2005; Soller & Lesgold, 2003) cited above, makes use of *real* data from student collaboration to assist in building a model for tutoring. The potential timesaving in data collection, data analysis, and system development provided with a single integrated tool could be significant. Second, given the complexity of collaborative learning, a 2-D visualization, in the form of a graph, may allow for a better understanding and analysis of collaborative behavior when compared with, for instance, a non-visual, linear representation, such as production rules. The traversal frequencies, discussed above, provide additional data to the graph visualization that could further assist analysis.

A REALIZATION OF BND: THE INTEGRATION OF COOL MODES AND THE BEHAVIOR RECORDER

An important underpinning of this work is the notion of component-based tutor development (McArthur, Lewis, & Bishay, 1996; Ritter & Koedinger, 1996). Our approach is to take an existing software application (what we term a "tool") and integrate it, with little or no modification, with a tutor or tutor agent. Using off-the-shelf or pre-existing software as the basis for building tutoring systems could result in substantial timesavings, as compared to the traditional approach of building tutors "from scratch." This is particularly important in developing tutors for collaboration (Walker, Koedinger, McLaren & Rummel, 2006). As pointed out above, the underlying interaction model in collaboration is much more complex than in the single-student scenario.

Cool Modes: The Collaborative Modelling Tool

Cool Modes is a collaborative software tool designed to support "conversations" and shared graphical modelling facilities between collaborative learners (Pinkwart, 2003; Pinkwart *et al.*, 2002). The tool is intended to facilitate collaborative problem solving and learning, but provides no tutoring capability. An example of a Cool Modes problem space is shown in Figure 1.

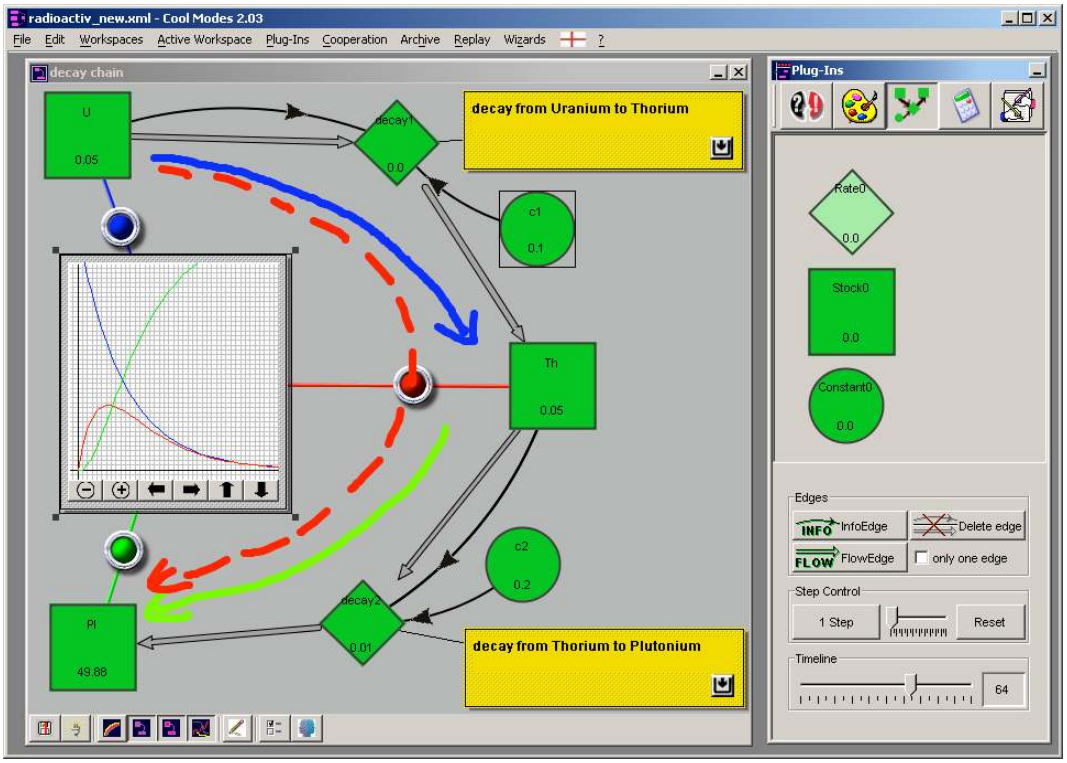

Figure 1: An example Cool Modes problem space. Here students are collaborating on a nuclear decay problem in a shared workspace.

Cool Modes provides users with a variety of plug-in objects, such as Petri nets, a turtle programming environment, a variety of text widgets, a "chat" area, etc., each of which has its own semantics and underlying representation. All of these objects are available in a toolbox from which students may drag and drop objects into workspaces for use. Each Cool Modes user has her own private workspace in which objects can be privately created and updated, while all users have access to a shared workspace, which is visible to all collaborators and may be updated by any participant. Cool Modes is extensible; new objects

adhering to a well-defined API may be added to the toolbox. Communication and translation between different object types is achieved through *reference frames*, a set of entities and rules that facilitate semantic mapping between objects (Pinkwart, 2005). Some of these sets are visual languages (Hoppe, Gassner, Mühlenbrock, and Tewissen. 2000), which have been designed as formal representations of specific domains. Some are graphical argumentation and discussion languages, which have been designed for collaboration and thus can be considered coordinating representations that help to structure collaboration (Introne & Alterman, 2006).

CTAT: The Tutor Authoring Environment

The graph building and (eventual) tutoring component of the BND integration is provided by CTAT, an authoring tool for intelligent tutors. Authoring systems comprise an important area of ITS research (Murray, Ainsworth & Blessing, 2003). Among these tools, CTAT fits into the "Domain Expert System" category described by Murray (1999). It supports authors in building cognitive tutors, a form of "model-tracing" tutor based on cognitive psychology theory (Anderson *et al*., 1995). As of the winter of 2005-2006, cognitive tutors have been deployed in over 2000 schools in the United States[2]. A cognitive tutor is composed of a problem representation and a set of production rules that model both desired and buggy behavior; it is general enough to tutor students on a range of problems within a particular domain (e.g., geometry, algebra). Model tracing involves (a) matching actual student behavior during problem solving with the desired behavior represented by the production rules and (b) identifying deviations from that behavior, either in the form of so-called "buggy rules," which model known misconceptions, or in the form of no-model conditions, where no production rules match the student action. Cognitive tutors are difficult to develop, typically requiring AI programming expertise.

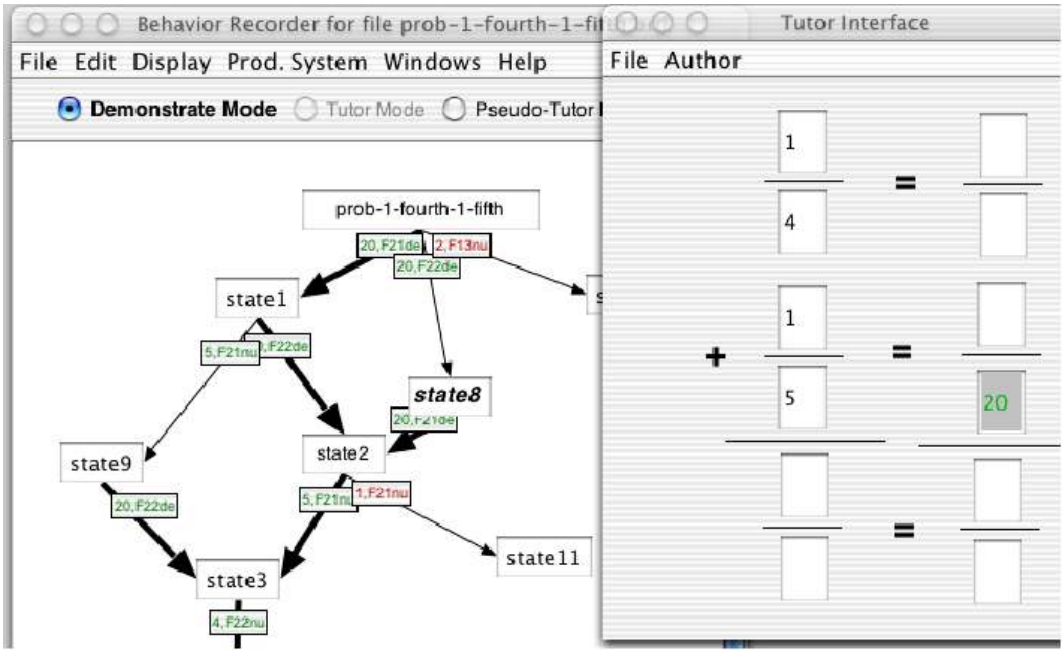

Figure 2: The Behavior Recorder (BR) records authors' actions in any interface created with CTAT's specialized GUI widgets.

A specialized type of cognitive tutor also supported within CTAT is an *example-tracing tutor*, a tutor that behaves much like a regular cognitive tutor, except that it provides instruction for only a single problem instance and is much easier to develop. Example-tracing tutors are developed using

[2] It should be noted, however, that the Algebra and Geometry Cognitive Tutors in use at schools across the United States were developed before CTAT was created, so they were not built using CTAT.

"programming by demonstration" (Lieberman, 2001), an approach that allows authors with no programming skills to build tutors. An illustration of an example-tracing tutor for fraction addition is shown in Figure 2.

In CTAT, an example-tracing tutor is developed as follows:

- First, the author builds (or uses) a graphical user interface (GUI) with a set of CTAT widgets, which are ordinary user interface components augmented for use with a tutoring system. The GUI for a fraction addition tutor is shown on the right side of Figure 2.
- Second, the author demonstrates correct, alternative correct, and incorrect actions. A CTAT tool known as the *Behavior Recorder* (abbreviated BR) records all of these actions and stores them in a structure known as a *behavior graph*, shown on the left side of Figure 2. Each edge of the graph represents an action taken by the student on a particular widget of the GUI. For instance, out of the start state (labelled "prob-1-fourth-1-fifth") are two correct actions ("20, F21den" and "20, F22den"), in which a proper common denominator is entered in either of the converted fractions, and one incorrect path ("2, F13num"). The student's action is represented as a Selection (the GUI widget selected, such as the text area named "F22den"), Action (the type of action taken, such as "Update Text"), and Input (the value provided, such as "20"). Each node of the graph represents a state of the interface after a path of edges from the root to that node has been traversed. In the case above, since the Tutor Interface shows the denominator "20" in the second converted fraction but no other student action, state8 is highlighted in the BR (the boldfaced state label) as the current state.
- Third, after the behavior graph has been created by problem demonstration, the author can annotate the graph by labelling buggy edges (e.g., the incorrect path represented by "2, F13num" in Figure 2), hints, feedback messages, and skills associated with the edges.
- Finally, the author can test the model by "executing" the example-tracing tutor, acting like a student or observing actual student use. The whole process typically iterates, as the author improves the model and fixes problems during testing.

The Technical Integration

As described above and depicted in Figure 3, the fundamental idea behind BND is to use a collaborative tool, in this case Cool Modes, to generate traces of actions created by the participants in the collaborative groups and have those actions recorded in the BR, providing the beginnings of a real tutor. Because the BR is a component with a well-defined message interface (these messages are called "Dormin Messages"), the initial integration Cool Modes / BR integration only required a translator that took Cool Modes student actions and transformed them to Dormin messages. In this initial implementation, we integrated the tool and the tutor in loosely-coupled fashion by translating Cool Modes log files, which are represented in XML format, using XSLT (McLaren *et al*., 2004a; 2004b).

Our initial implementation fed the XML-based log files of Cool Modes asynchronously into the BR and thus was not able to support live capture of user actions. It also provided only one-way communication, which could support recording of student actions but not tutoring. The only development effort required for this file-based interoperability was the definition of a semantic mapping of user events to Dormin Messages and the resulting XSL-transformation script to achieve this mapping. Cool Modes did not have to be updated to support this solution, while the BR had to be extended with the traversal frequency feature (discussed above) to make the BND approach effective.

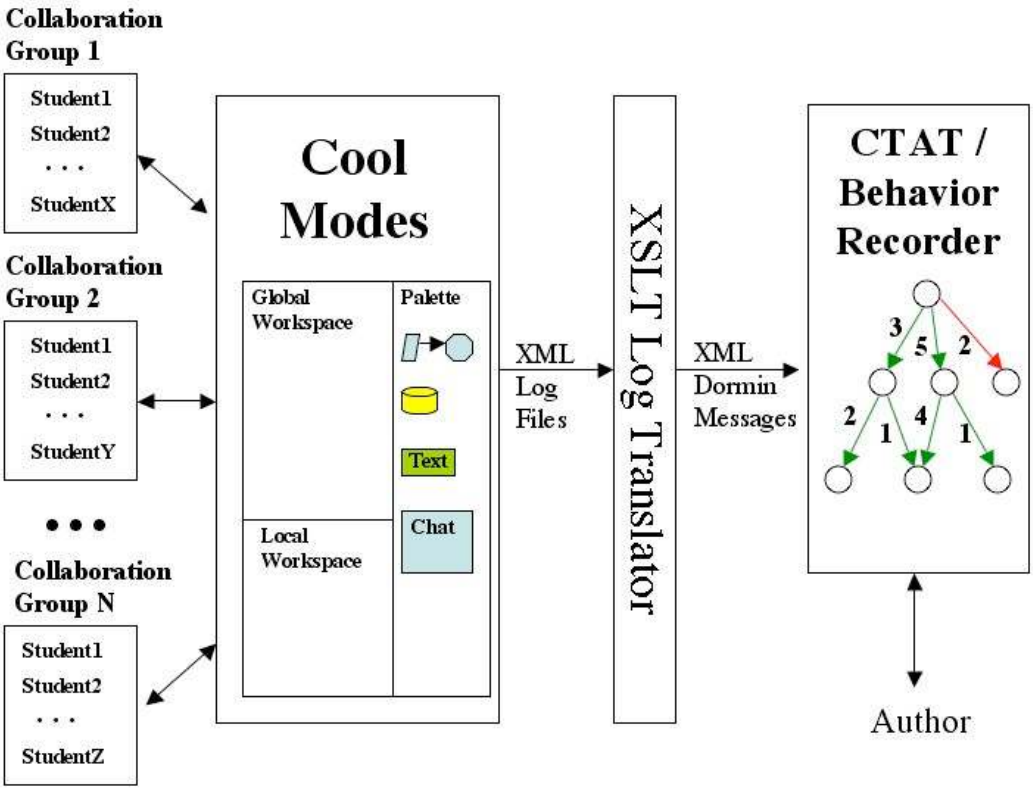

Figure 3: Bootstrapping Novice Data: Initial integration of Cool Modes and the Behavior Recorder.

In our current, more advanced implementation (Harrer *et al*. 2005; McLaren *et al*., 2005), we have integrated Cool Modes and the BR in a real-time, synchronous fashion: Both tools remain fully operational independently, but now they can exchange messages bi-directionally using Cool Modes' MatchMaker communication server and a "Tutor Adapter" (see Figure 4). Now, a student action causes the Cool Modes client to send an event to the MatchMaker server, which sends the event to the Tutor Adapter, which in turn forwards the event to the BR. If an author were to create an example-tracing tutor and switch the BR from recording to tutoring mode, then it would respond to incoming events by sending appropriate buggy messages and hints to the proper student or students[3].

The implementation effort to achieve this bidirectional solution consisted of several changes. While the Cool Modes application itself didn't need to be extended, an additional component, the Tutor Adapter, was developed to send MatchMaker events from the collaborative application(s) to the Behavior Recorder. This component uses the conceptual mapping already defined for the first version but sends messages through a socket connection. The same communication is required from the BR; otherwise no modifications were required. The rationale behind this distributed architecture is the flexibility gained by running different components on different computers. This architecture substantially reduces the cost of the computing requirements for running the entire integrated system, for no single machine is relied upon to provide a lot of processing power or memory. Since our intent is eventually to use AI and data mining techniques to support the tutor model (discussed later), it may be desirable to have a dedicated machine executing these computations, perhaps in parallel for multiple groups of students.

[3] It should be noted that we have not yet defined the incoming events in Cool Modes. While technically this is not a difficult task, it is dependent on the more difficult tasks, which we are currently undertaking, of collaborative analysis and defining the feedback we would want to provide to students.

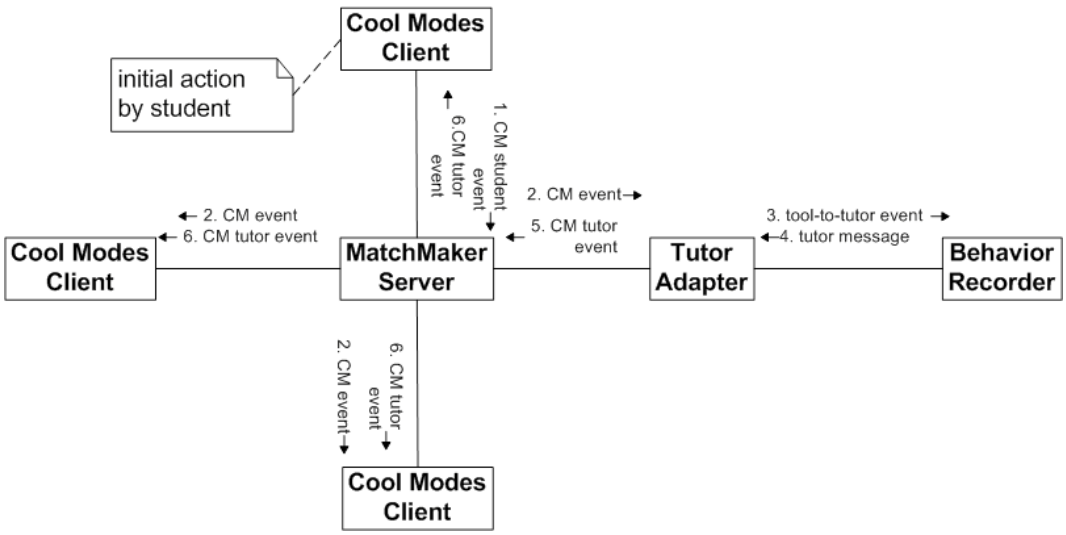

Figure 4: Collaborative system diagram showing the message flow between Cool Modes and Behavior Recorder.

While it is difficult to precisely estimate the time and cost savings due to implementing BND in a component-based fashion instead of developing either a tutor agent for Cool Modes or a collaborative application for the BR, we can say the following: the Cool Modes application on its own was a doctoral thesis project and several student programmers' work, while CTAT has been under development by a team of 2-4 programmers for just over 3 years. Given this experience, for either alternative approach we estimate at least one-person year for the development effort and probably much more. Similarly, the MatchMaker communication architecture required several person months of development time. For a collaborative application with less flexibility, restricted to a specific domain, we would still assume development time of at least 3-4 person months. On the other hand, extending our existing tools as described took less than a person week of effort on each side, demonstrating the enormous advantage of component-based development.

Another notable aspect of our implemented integration is that a change in learning domain only requires a different Cool Modes plug-in. All of our pilot tests and the two studies, described in following sections, were realized by using different Cool Modes plug-ins with no programming effort. The flexibility of this BND implementation can be seen in the variety of scenarios we tested: a priority planning task (i.e., the NASA game), fraction addition, and open graphical modelling problems, including UML diagrams and Entity/Relationship modelling.

Figure 5 depicts the user's view of one of our experimental scenarios, UML object-oriented modelling (see "Study 1"). Cool Modes, shown on the left, provides the user interface for the student, which includes a shared workspace that all collaborating students in a session can view and update. Cool Modes sends messages describing students' actions (e.g., "student *A* created classification link *L*") to the Behavior Recorder, shown on the right, which stores the actions in a behavior graph. Each edge in the graph represents a single student action, and paths through the graph represent series of student actions. The behavior graph of Figure 5 represents multiple collaborations; this can be seen by the traversal frequencies shown on the edges of the graph.

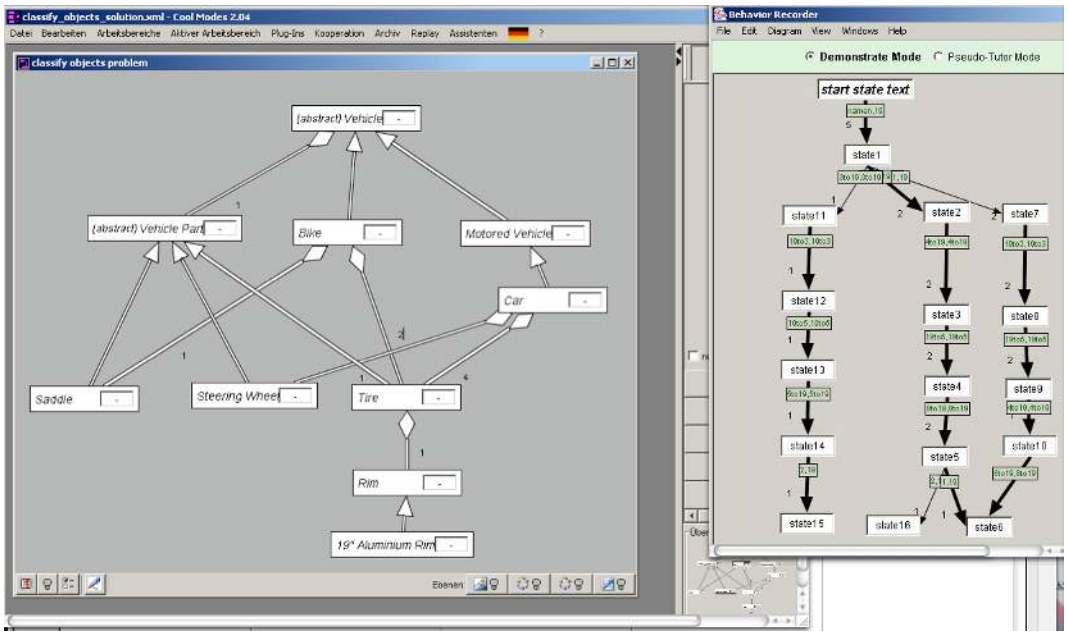

Figure 5: The user's view of the integrated Cool Modes (left) and the Behavior Recorder (right) environment. This shared Cool Modes workspace is from a vehicle classification / composition task.

STUDIES TO ANALYZE COLLABORATION USING THE INTEGRATED SYSTEM

We ran two exploratory studies to assess the information required by the BR, to investigate the types of problems that might be amenable to our approach, and to analyze two interesting collaborative tasks. Each study involved a graphical modelling problem and tested the effect of the initial organization of objects on the collaborative problem-solving effort. In Study 1, in which we had students perform a UML modelling task, we identified five elements of collaboration that were relevant in solving the problem and demonstrated the need for adding multiple levels of abstraction to the BR in order to represent those elements. In Study 2, we verified that the five elements of collaboration were generalizable to a Petri Net modelling problem and explored how the five elements could be analyzed and tutored using the BR. We believe that our results can be generalized to many other graphical modelling problems that can be tackled in Cool Modes, but the proof will be in further studies. We chose these study tasks to show the potential usage of our approach across different domains and because the same tasks were the focus of earlier studies (i.e., object-oriented modelling (Soller & Lesgold, 2003) and Petri Net modelling (Gasevic & Devedzic, 2003)). We conducted the studies using students from a course on "Modelling Techniques in Computer Science" at the University of Duisburg-Essen. All our subjects were majors in applied computer science or computer engineering.

Study 1

Study 1 Method

We performed Study 1 to help identify properties of the collaboration that might be important and to define the properties in a way that could be easily expressed in the behavior graph. To facilitate this process, we varied certain properties of the initial problem in an attempt to evoke a rich variety of collaborative behaviors. We wished to discover whether, in a graphical problem-solving domain, an

organized visual arrangement of objects leads to quicker and better collaborative solutions than a disorganized arrangement. We also were interested in whether the rearrangement of objects facilitates quicker and better results and how this rearrangement might be conducted in a collaborative scenario.

To explore these questions and test how the BND methodology might be a useful analysis tool, we assigned 16 students to 8 dyads and asked each dyad to solve an object-modelling problem using the Cool Modes / BR integrated system (one subject was a class assistant). All the students had knowledge of object-oriented modelling at the time of the study and participated in this optional exercise as a means to get additional preparation and experience for an exam. Seven of the students (pairs 6, 7, 8 and one in pair 5) had had previous experience with Cool Modes. All students received approximately 5 minutes of instruction on how to use the system (especially using the graphical modelling functionality and the chat) prior to the study. The student pairs worked at separate workstations, back-to-back in the same room. They shared a single Cool Modes workspace.

The students were asked to specify IS-A (i.e., classification) and PART-OF (i.e., composition) links between pre-defined, given objects in the workspace using the Unified Modelling Language, a graphical modelling technique. The objects in the given problem were vehicles (e.g., "Car") and parts of vehicles (e.g., "Tire"). A paper handout with detailed instructions for the task accompanied this setup of the collaborative environment. To stimulate collaboration we used an unequal resources design akin to jigsaw experiments (Aronson, Blaney, Stephan, Sikes & Snapp, 1978). One student was provided with IS-A links only and one with PART-OF links only, so that no student could solve the problem alone, and the students could take three types of actions:

- *Chat* actions: "talking" to a partner in a chat window;
- *Move* actions: repositioning an object in the shared workspace; and
- *Creation/deletion* actions: creating or deleting links between objects.

The students had approximately 25 minutes to solve the problem.

The 8 groups were randomly assigned to two experimental conditions. In the *ordered or organized* condition (Condition 1), the initial presentation showed related objects visually close to one another, to provide a well-organized display of the desired final network. For example, the two abstract classes "Vehicle" and "Vehicle Part" were located near the top of the visual space, and most of the subclasses were located near their respective super classes. In the *scrambled or disorganized* condition (Condition 2), objects were positioned randomly. Groups 1 to 5 were in the scrambled condition; groups 6 to 8 were in the ordered condition. We hypothesized that students in the ordered condition would use the visual organization as a scaffold in problem solving and would be less likely to rearrange objects in the visual space.

Study 1 Results and Analysis

To analyze the results, we looked both at students' solutions and at their process for arriving at those solutions. We looked only at elements that could be automatically gathered by the BR, and we tried to examine how actions could be evaluated at different levels of abstraction and correlated with the quality of the students' solutions.

Characterizing students' solutions. All 8 dyads completed the task. Students' solutions can be divided into three categories: ***good*** solutions (groups 5 and 8), ***incomplete*** solutions (groups 2, 6, and 7), and ***poor*** solutions (groups 1, 3, and 4). All three poor solutions were in the disorganized condition, and two of the incomplete solutions were in the organized condition. While one of the good solutions was in the organized condition, one was in the disorganized condition. There were negligible differences in the time required to complete the task among the different solution categories. The informal results suggested that a clearly organized problem state does not necessarily lead to quicker and better solutions than a disorganized problem state, but it does lead to different types of errors.

The conceptual distinctions between good, incomplete, and poor solutions were related to the errors committed in solving the classification / composition problem. In the good solutions, errors reflected misunderstandings about the meaning of words in the chosen domain (viz., vehicle parts). Otherwise,

students correctly divided the objects into subclasses and super classes and correctly placed the inheritance and composition edges. In the incomplete solutions, students only connected one inheritance or composition link from each class. As a result, they had too few links (see figure 6), and left out key relationships. In the poor solutions, students would often connect a class to multiple ancestor nodes of the same lineage (see figure 7 for the classes "Vehicle Part", "Rim", and "19'' Aluminium Rim"). For example, in Group 3, students connected "Car" to both "MotorVehicle" and "Vehicle." The poor solution groups also typically created too many edges and were logically inconsistent about the connection decisions they made.

Solutions can also be characterized by the way students tended to arrange nodes in the shared workspace. In the good solutions, the students separated the two abstract classes, placing one at the top of the workspace and the other at the bottom. IS-A and PART-OF links flowed in opposite directions and crossed infrequently and only when necessary. Objects were organized in rows that reflected their level of abstraction. In the incomplete solutions, the two abstract classes were placed relatively close to one another, all links pointed in one direction, and there were no crossed links. The objects tended to be clustered together and were organized into fewer rows than in the good solutions, but the rationale behind the organization wasn't as clear. Finally, the poor solutions had the abstract classes positioned without an obvious rationale. They had much longer edges pointing in all directions and frequently intersecting with one another (see figure 7). As a result, the poor groups tended to use the entire shared workspace.

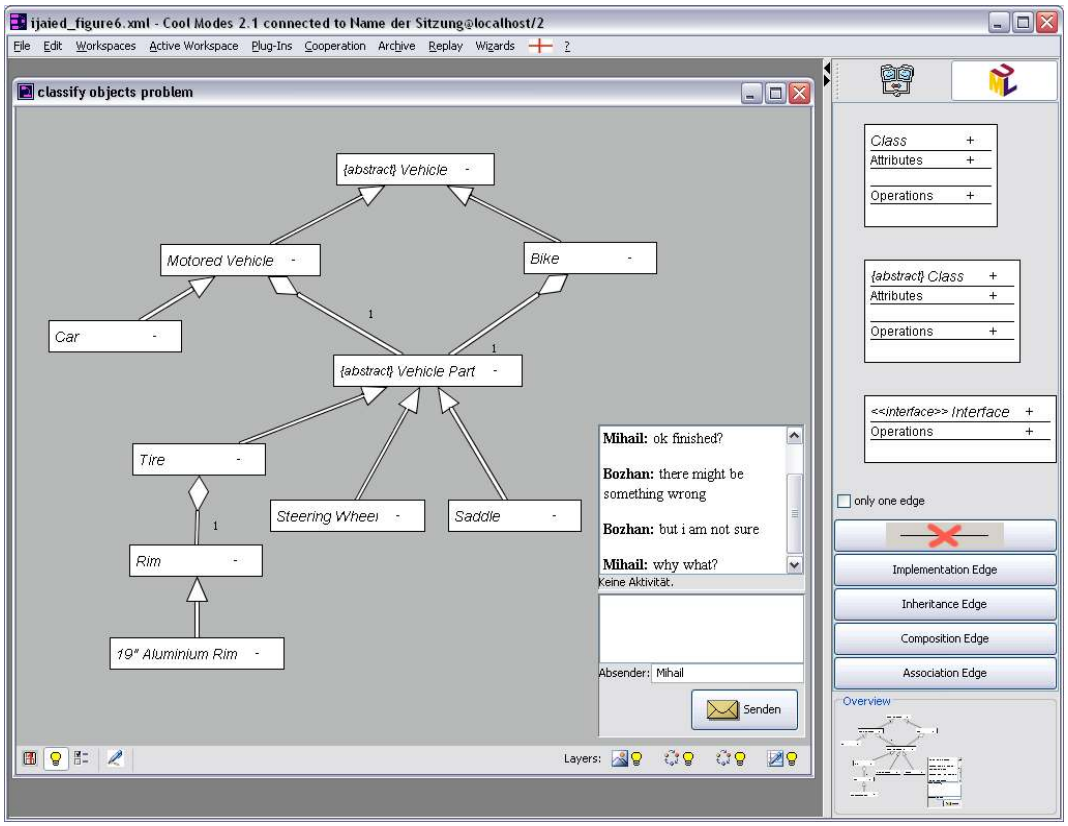

Figure 6: Incomplete Solution of the Classification/Composition Problem. Notice that only one edge was created from each class or object.

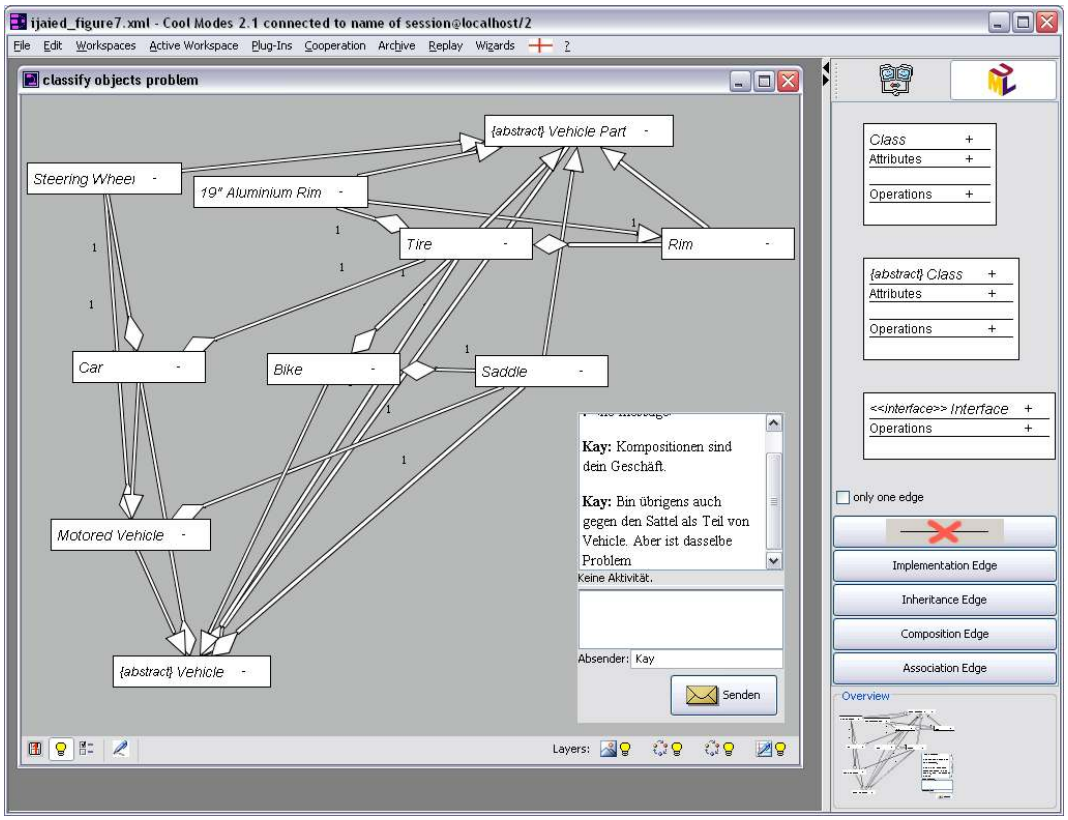

Figure 7: Poor Solution of the Classification/Composition Problem. Notice that abstract classes are positioned without an obvious rationale and there are longer intersecting edges.

Analyzing students' solution processes. We then analyzed the processes associated with the development of each solution. While working on the problem, students could take three types of actions: *chat* actions, *move* actions, and *creation/deletion* actions. Solution types showed differences in collaborative and task-oriented behavior. In terms of collaboration, the students with the good solutions had different approaches. Group 8 worked completely collaboratively. Members would take turns moving and creating objects and, for a given group of objects, one member would reposition objects while the other would create the edges. In contrast, the members of Group 5 worked completely in parallel, coordinating their actions only to correct their partner's mistakes. In the incomplete and poor groups, pair members shared the work. The poor groups were informal in their turn taking, while students in the incomplete groups would alternate taking the initiative. With the exception of Group 5, groups decided on their actions using the chat window and ensured that both members agreed on the actions being taken.

The three solution groups coordinated phases of chatting, moving, and creating/deleting differently. In the good solutions, the approaches of two pairs were dissimilar. Group 8 collaborated by alternating chat phases, move phases, and creation/deletion phases, while Group 5 alternated between move phases and creation/deletion phases, and primarily communicated visually. In both groups, adjacent phases referred to the same objects and levels of abstractions, displaying a coherent problem-solving strategy. Both pairs were the only groups to have more move actions than chat actions. The incomplete groups had fewer move actions, longer phases of chatting, and fewer deletions. They adopted the inefficient strategy of discussing many future actions, creating a single edge, and then repeating the discussion. On the other hand, they would consistently reorganize objects before creating edges, which may have contributed to the tree-like organization of the classes. The poor groups engaged in long phases of chatting or moving, but showed less coherence between the objects they were discussing, objects they were creating, and objects they were moving. They deleted a lot of edges and tended to create particular edges before repositioning them. This disorganized approach probably led to the conceptual and visual disorganization

of their final solutions. Another difference was in terms of the selection of objects: Both the good and the incomplete solutions would focus on objects based on their level of abstraction; they would manipulate a given super class and all its subclasses, and then move on to another group of objects. On the other hand, students in the poor solutions appeared to let the problem organization guide their actions. They would draw edges between classes that were physically close to one other in the shared workspace, rather than classes that were semantically related. Differences in object selection probably related to differences in the consistency of the final solutions.

These results indicate that we should focus on five elements when evaluating students' performance on graphical modelling tasks: *conceptual understanding, visual organization, task selection, task coordination, and task coherence* (see Table 1). These elements were chosen because they represent different positive and negative strategies of student action that appeared to affect the quality of groups' solutions. Students within each solution type (good, incomplete, and poor) tended to make the same types of mistakes within each of these categories.

Conceptual understanding refers to a pair's ability to correctly place the inheritance and composition edges, while visual organization refers to a pair's ability to visually arrange the classes and edges according to a schema expressing the relations intended by the student. These two elements are linked; conceptual understanding of the problem was often reflected in the visual organization, and therefore conceptual steps tended to parallel organizational steps. For example, students in the incomplete solution groups appeared to believe that they could create only one inheritance or composition link extending from each object, and their solutions thus tended to take on rigid tree structures.

Table 1: Solution Types and Dimensions of Analysis in Study 1

	Good ***(Groups 5 and 8)***	***Incomplete*** ***(Groups 2,6, and 7)***	***Poor*** ***(Groups 1, 3, and 4)***
Conceptual Understanding	**Good** – only trivial mistakes	**Incomplete** – only one link extended from each class	**Inconsistent** – too many links extended from each class
Visual Organization	**Good** - based on abstractions	**Overly organized** – had a tree-like structure	**Disorganized** – had long, intersecting links
Task Coordination	**Good** – good alternation of phases and distribution of work	**Hesitant –** long chat phases, formal turn-taking structure	**Impulsive –** creation before organization, informal turn-taking.
Task Coherence	**Good** - adjacent phases referred to similar objects and levels of abstraction.	**Good** - adjacent phases referred to similar objects and levels of abstraction.	**Poor** – adjacent phases referred to different objects
Task Selection	**Good** - based on abstractions	**Good** - based on abstractions	**Poor** - based on visual proximity

Task coherence, task coordination, and task selection reflect student strategies for collaborating on the problem. Task coordination refers to skills in coordinating actions, without reference to the content of the actions. It includes distributing the work among group members, and spending appropriate amounts of time in each phase. The good groups exhibited successful task coordination, in part because they spent more time moving objects than talking about them. Task coherence refers to the appropriateness of the content of student actions. Students in the incomplete groups showed poor task coherence by chatting at length about many different links and then creating a single link. Task selection refers to a student's

ability to set subgoals for solving the problem by drawing edges between classes in a sensible order. A breakdown in task selection leads to disorganized and incoherent solutions, as seen in the poor group. Evaluation of the students' problem-solving process should assess these three skills.

We identified these five dimensions by using a data-driven approach to analysis, as opposed to a theory-driven approach. The results of applying BND methodology are inherently domain-specific and data-driven: It is used to develop a model based on student performance on a given task. Because we wished to assess the potential for BND, during our analysis we only considered elements that can currently be analyzed and tutored using the BR. We made a conscious decision to look in detail at three basic elements: the user performing the action, type of the action, and target of the action. We avoided elements that required in-depth coding, like the content of chat actions, because those elements cannot currently be automatically identified in the BR. In general, we wished to determine not only how those basic elements affected student collaboration in this specific domain, but also how they could be relevant for discriminating good collaboration from bad collaboration in general.

Relationship Between the Five Dimensions and Other Analysis Approaches

It is important that despite our data-driven approach, we can show that the process dimensions that we have identified (i.e., task coordination, task coherence, and task selection) have a theoretical basis in collaborative learning. Spada, Meier, Rummel, and Hauser (2005) combined a bottom-up analysis with a review of the literature to identify nine dimensions of collaborative process, drawn from major theories of collaboration such as Clark and Brennan's (1991) theory of grounding and Stasser and Titus' (1995) theory of knowledge sharing. Spada *et al*. describe sustaining mutual understanding, information pooling, and reaching consensus as processes involving the convergence on similar concepts during collaboration, which can be assessed by *task coherence*, which relates to the objects used in adjacent turns by users. The high-level dimensions of coordinating communication, technical coordination, and shared task alignment all deal with the coordination of tasks between users, and can be probed with *task coordination*, which involves the nature of user turns. Finally, the Spada *et al*. planning dimensions of task division and time management relate to *task selection*, which deals with how users decompose a task into subgoals. While there is a link between the Spada *et al*. dimensions and the dimensions we have identified, there is not a one-to-one mapping, as they operate at a different level of granularity. Our dimensions are designed to involve easily detectable patterns of student action, while the Spada *et al*. dimensions are a result of extensive coding of protocol data. We gain automaticity with our approach but lose the richness of the Spada *et al*. analysis. Eventually, we hope to move toward a BR that facilitates a deeper analysis of collaborative protocols.

Nevertheless, it is likely that our dimensions will prove to be relevant for tasks in other domains, for they function as units of analysis for collaborative process and have some basis in collaborative learning theory. For example, let us look at how our dimensions might transfer to collaborative essay writing. In this domain, conceptual understanding would refer to whether the students stated the key points of the essay problem. The analogue of visual organization would be the organization of the paragraphs in the essay itself. Task coordination might refer to how students shared the task of writing the essay, and whether they alternated planning, writing, and proofreading stages appropriately. Task coherence would be whether students referred to the same elements of the essay in sequences of essay-writing actions (for example, in adjacent planning and writing phases, or in writing adjacent paragraphs). Finally, task selection would assess how well students decomposed the task of writing the essay in the first place. Although essay writing is quite different from graphical modelling, the dimensions are still useful units of analysis.

How Abstraction Might Play a Part in the Analysis of the Five Dimensions

One of the key goals of this first study was to gain insights into how we might enhance the BND methodology to provide more helpful data and analysis. In theory, the BR should be able to facilitate analysis of all the skills from Table 1. However, these five elements are not supported in the current BR, so that it is difficult to classify behavior and provide tutoring support. When all actions were input, the initial BR graph showed no common paths. In an attempt to produce convergent paths in the behavior

graphs, we restricted input to the BR to creation and deletion actions. Unfortunately, analyzing creation and deletion alone appears too limited to be useful. Although the BR records sequences of actions at a single, detailed level of generality, the nature of this problem indicates that student skills at different levels of abstraction need to be addressed, and the BR needs to be able to create hierarchies of behavior graphs.

Conceptual understanding and visual organization appear to be addressable on an action-by-action basis. On the other hand, task coordination and task coherence seem to be best evaluated through the analysis of *phases* of action, or chains of the same type of action. A chain of chat actions followed by chain of creation actions would indicate that, on a task coordination level, students have decided to discuss what objects they should create and then create some objects. This type of information is difficult, if not impossible, to extract from an action-by-action representation. Finally, task selection seems like it could be analyzed in the BR by aggregating multiple phases of action, which indicate high-level goals.

We took a step toward addressing this problem by modifying the BR to support recording at different levels of abstraction (see the section "A Granularity Analysis: Abstracted Actions versus Individual Actions"). Another approach we will consider is to classify actions and action sequences in Cool Modes as described in (Harrer & Bollen, 2004). This approach could be used to process actions at different levels of abstraction.

Study 2

Study 2 Method

We performed Study 2 to verify that abstraction is important and generalizable across different graphical modelling problems. We also wished to explore how abstract actions could be analyzed according to our five dimensions and tutored using the BR. We again varied the organization of the starting state in an attempt to produce a greater variety of collaborative actions.

We assigned 16 students to 8 dyads and asked each dyad to solve a traffic light modelling problem using the Cool Modes / BR integrated system. The objects in the given problem were traffic lights (e.g., "car.red", "ped.green") and transitions. The students were volunteers from the same course as in Study 1 at the University of Duisburg ("Modelling Techniques in Computer Science"). Five of the students (pairs 2 and 5, one in pair 3) had had previous experience with Cool Modes, and all students received approximately 5 minutes of instruction on how to use the system before the study. The student pairs worked at separate workstations, back-to-back in the same room. They shared a single Cool Modes workspace, configured in a manner similar to that shown in Figure 1 (but without the links between nodes) and each received a paper handout with the instructions to solve the task.

The students were asked to model the progression of car and pedestrian lights at a given intersection using Petri Nets (i.e., they were asked to draw links between traffic lights and transitions, and use tokens to represent an illuminated light). While working on the problem, students could take four types of actions:

- *Chat* actions, "talking" to a partner in a chat window;
- *Move* actions, repositioning an object in the shared workspace;
- *Creation/deletion* actions, creating or deleting edges and tokens; and
- *Simulation* actions, firing transitions to move tokens from one light to another.

Each dyad was assigned to one of two possible conditions. In the *ordered* condition, the pairs attempted to solve a problem in which related objects were in juxtaposition to one another, providing a well-organized visual display of the final network. In this case, the objects were organized like real-world traffic lights, with the car lights on one side, the pedestrian lights on the other side, and the transitions in the middle. In the *scrambled* condition, the pairs solved the same problem, but objects were randomly positioned in the workspace. The student pairs had 30 minutes to work collaboratively on the task.

For a post-test of their knowledge gains, each student individually (without collaboration) had to model another typical topic for the Petri net technique: regulating competing access to exclusive resources (3 processes competing for 2 processors in a computer) from an empty workspace. For this task the students had approximately 20 minutes to solve the problem.

Study 2 Results and Analysis

Again, we wanted to analyze the results in such a way that the BR could perform the analysis automatically, without excessive pre-processing. We wanted to validate the five dimensions we identified in Study 1: conceptual understanding, visual organization, task coordination, task coherence, and task selection. We also wished to assess the potential for tutoring those dimensions. Table 2 summarizes our analysis of the Study 2 data.

To evaluate *conceptual understanding*, six concepts were identified as required by the problem statement:

1) pedestrian traffic lights should change in the correct order;
2) car traffic lights should change in the correct order;
3) there should be no point where the lights stop changing;
4) depending on the situation, only two or three lights should be on at a time;
5) there should never be a solution where the pedestrians and the cars are moving at the same time;
6) the pedestrian and car lights should change simultaneously.

Solutions were rated on a scale of 9 based on these categories. The first two concepts were weighted greater than the following four, because they were more critical to solving the problem. In this case the scrambled group produced significantly better solutions than the ordered group (Ms = 5.25 versus 1.75). Solutions could be further divided into good (groups 1 and 2, M = 6.5), mediocre (groups 3, 4, and 5, M = 3.7), and poor solutions (groups 6, 7, and 8, M = 1.3). The scrambled group had two good and two mediocre solutions, and the ordered group had one mediocre and three bad solutions.

The *visual organization* of the final solutions can be described in terms of two competing schemes: "real-world," wherein the car and pedestrian lights were separated and arranged in the red/yellow/green order typical of traffic lights, versus "easy-to-follow," wherein the number of edge crossings was minimized. The "real-world" arrangement often meant that the best place for the transitions was in the center of the shared visual space; these solutions were difficult to follow, with too many intersecting links extending in too many different directions. In the ordered start state, the ideal solution corresponded to the real world, but was not easy-to-follow (see figure 8). Three out of the four ordered groups (Groups 5, 6, and 8) did not significantly reposition the objects from their original places in the start state. Consequently, the edges that they drew between objects intersected often (37% of the time), and were difficult to follow. Group 7, the other organized group, moved transitions so that the edges they drew made a square, which reflected their overly simplistic, linear conception of the problem. On the other hand, all four of the groups in the scrambled condition moved objects from their initial disorganized state to good final solutions that were relatively easy to follow. Groups 3 and 4 created a real-life arrangement of the traffic lights, but put the transitions in locations that made the edges between transitions and objects easier to follow and separate from one another. They had the best balance between a real-world arrangement and an easy-to-follow solution, with 31% intersecting links. Groups 1 and 2 organized their traffic lights in a way that was clear to follow with an average of 7% intersecting links, but had little real-world correspondence. It appears that our conception of an "organized" condition may not have been as well founded for this particular problem, since an easy-to-follow arrangement seemed to relate to better solutions than a real-world arrangement (see Figure 9).

The results for the *task coordination* differed significantly between good and bad solutions. Good groups had a significantly fewer percentage of chat actions than mediocre and poor groups (Ms = 12%, 48%, and 44%), and a significantly lower percentage of chat phases (Ms = 20%, 40%, and 39%). The good groups and the two mediocre groups in the scrambled condition also had a significantly higher percentage of move actions than the ordered groups (Ms = 28% and 8%) and significantly more move

phases (Ms = 23% and 11%). There was some statistical evidence that the ordering of phases also had an effect on whether groups did well or poorly, with the optimal sequence of phases being chat->move->creation/deletion->simulation. Further, the good groups had a less balanced work distribution than the mediocre and poor groups. The ordered (and therefore less successful) groups split their time between having one person perform the whole phase (M = 37%), the other person perform the whole phase (M = 34%), or both people taking action in the phase (M = 28%). The scrambled groups had fewer phases where both people took action (M = 15%), and a less balanced distribution of individual phases (Ms = 53% and 32%). These results were surprisingly congruent with the task coordination results for Study 1.

Table 2: Solution Types and Dimensions of Analysis in Study 2

	***Good* (Groups 1 and 2 (both in scrambled condition))**	***Mediocre* (Groups 3, 4 (scrambled), and 5 (ordered condition))**	***Poor* (Groups 6, 7, and 8 (all in ordered condition))**
Conceptual Understanding	**Good** – mean score 6.5 of possible 9 on concepts	**Mediocre** – mean score 3.7 of possible 9 on concepts	**Poor** – mean score 1.3 of possible 9 on concepts
Visual Organization	**Easy to follow** – very few intersecting links	**Real-world** – yet the scrambled groups created a layout with relatively few intersecting links	**Real-world** – many intersecting links or (group 7) overly simplistic square
Task Coordination	**Efficient** – fewer chat actions and phases; less balanced work distribution	**Less efficient –** longer chat phases; balanced work distribution	**Inefficient –** long chat with fewer move actions; more turn-taking
Task Coherence	**Good** - adjacent phases referred to same objects	**Good** - adjacent phases referred to same objects	**Good** - adjacent phases referred to same objects
Task Selection	**Object-focused** – chain of tasks remained focused on same object or class	**Object-focused** – chain of tasks remained focused on same object or class	**Object-focused** – chain of tasks remained focused on same object or class

Although *task coherence* varied between conditions in Study 1, there were few differences on this dimension between groups in Study 2. Groups referred to an average of 1.8 objects per phase in move phases, creation/deletion phases, and simulation phases. All groups tended to refer to the same objects across multiple phases.

Task selection also did not differ between groups in this study, but commonalities between groups provided insight into the collaborative process. Groups structured their actions based on the transitions from one state of traffic lights to the next. Creation/deletion actions were chained 79% of the time, in that the current edge being drawn involved an object used in the previous creation/deletion action. Groups tended to focus on either the pedestrian or the car lights at a given time; the current creation/deletion action tended to involve the same light class as the previous creation/deletion action 75% of the time.

Since the focus of this paper is on the analysis of collaboration as a step towards the creation of tutors for collaboration, we only briefly mention the results of the individual post-test of Study 2. In the post-test the solutions can be classified again in the three categories: Seven students had a good solution, including all four students from the good groups (pairs 1 and 2), two students from mediocre groups (one from pair 3 and one from pair 5), and one student from a poor group (pair 6). Five students had a mediocre solution, including three students from mediocre groups (one member each of pairs 3,4, and 5)

and two students from poor groups (one member each of pairs 6 and 8). Four students had a poor solution. One was from a mediocre group (pair 4), and three were from poor groups (one member from pair 8 and both members of pair 7). Compared to the classification of the collaborative solution one student improved by two categories (poor to good), four students improved by one category (two from mediocre to good, two from poor to mediocre) , while only one student had a worse category (mediocre to poor). This improvement in the results indeed suggests that collaboration has a positive effect on student performance. We will investigate this effect in more detail in follow-up studies, to get more reliable information if the collaboration produces better learning outcomes also for other tasks.

Additional Analyses of Studies 1 and 2

A Granularity Analysis: Abstracted Actions versus Individual Actions

After Study 2 we explored how abstracted actions (versus individual actions) within the BR might be used to help in building an appropriate example-tracing tutor of collaboration. In this section, we will discuss how the BR might be used to analyze results and, eventually, provide tutoring, focusing on conceptual understanding, a low level of abstraction which might be addressed by tutoring individual actions, and task coordination, a middle level of abstraction that might be better addressed by abstraction.

To explore tutoring of conceptual understanding, we used the BR to capture individual creation actions, and discovered that two groups (1 and 3) used the same correct strategy in creating the links necessary to have the traffic lights turn from green to yellow to red. This path in the graph demonstrated a conceptual understanding of how Petri Nets can be used to effect transitions. We will ultimately be able to add hints that encourage students to take this path, leveraging the behavior graph as a means for tutoring. In likewise fashion, the BR can also be used to identify common bugs in participants' action-by-action problem solving. For instance, the BR captured a common error in groups 1 and 2 of Study 2: each group built a Petri Net, in almost identical fashion, in which the traffic-red and pedestrian-green lights would not occur together. In situations like this, the behavior graph could be annotated to mark this sequence as buggy, thus allowing the tutor to provide feedback should a future student take the same steps.

On the other hand, it is clear that the level of individual actions is not sufficient for representing all of the dimensions. For instance, evaluating whether students are chatting "too much" or alternating phases in an "optimal" way is not easily detected at the lowest level of abstraction. To explore how we might do more abstract analysis, we wrote code to pre-process and cluster the Cool Modes logs at a higher level of abstraction and sent them to the BR. The pre-processing we did was straightforward; we simply identified groups of consecutive identical actions that were taken by the students in our dyads. So, for instance, if students 1 and 2 in a dyad exchanged 6 chat messages in a row, our pre-processing code would identify this as a (CHAT, 6) cluster. Figure 10 shows an example of this level of analysis from Study 2.

Instead of individual actions, edges in the graph represent phases of action (see the "CHAT", "MOVE", and "OBJEC" designations on the edges). The number to the right of each phase in the figure specifies how many instances of that particular action occurred during consecutive steps, e.g., the first CHAT phase, starting to the left from the root node, had 2 individual chat actions. The graph shows the first 5 phases of groups 2, 3, 5, and 6. Because the type of phase, the number of actions within each phase, and who participates (not shown in the figure), is recorded we can analyze the data and, ultimately, may be able to provide tutor feedback at this level. For instance, notice that the scrambled groups (2 and 3) incorporated move phases into their process, while at the same point; the organized groups (5 and 8) only used CHAT and OBJEC (i.e., creation/deletion) phases. Additionally, groups 5 and 8 began their collaboration with a lengthy chat phase, and group 5 continued to chat excessively (23 chat actions by group 5 leading to state22!). This level of data provided to the BR could help us to understand better the task coordination dimension. In addition, if provided at student time, the BR could also provide feedback to groups with "buggy" behavior; for instance, a tutor might have been able to intervene during group 5's long chat phase. In future work, we intend to further explore how this, and other levels, of abstraction can help us address not only the task coordination dimension but also the task coherence and task selection dimensions

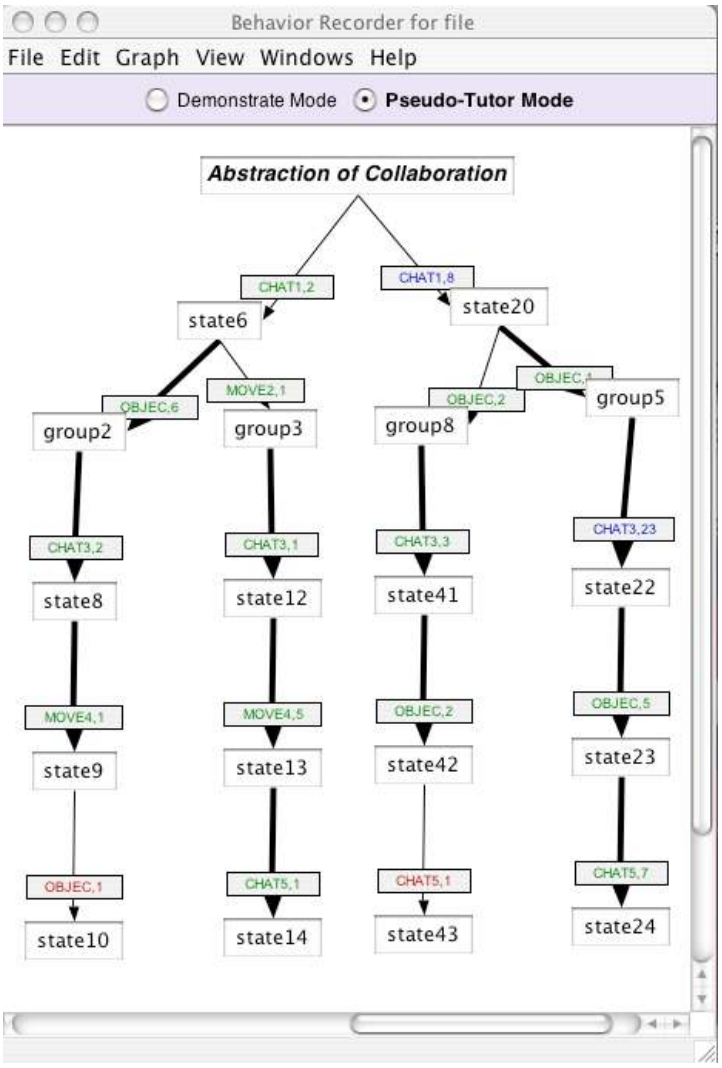

Figure 10: An Abstract-level behavior graph

The argument for having different levels of abstraction for analysis and tutoring with the BND approach is also supported by observations considering the manual creation of example-tracing tutors. Petri nets have an operational semantics, i.e. they can be simulated and tested during and after the construction process. Thus the experimentation with subparts of the created model and the configuration of the net by changing the token values will happen naturally, especially with the students that favor an experimental approach. An expert creating a tutor manually would have to pre-define phases of experimentation and re-configuration throughout the whole modelling process, which would lead to state-explosion for this problem type. This issue is discussed further in the next section.

Expert Interview and Complexity Analysis: Bootstrapping versus Manual Creation of Example-Tracing Tutors

As part of Study 1 and to further investigate whether and how the BND approach might be preferred to "from scratch" development of example-tracing tutors, we interviewed an expert in the problem domain, a computer scientist with a strong background in object-oriented modelling. We wanted to better understand what kind of misconceptions or problems students might have and thus what kind of tutoring would be required for the classification/composition problem, from the point-of-view of an expert. We also wanted to uncover inherent problems in having an expert manually define an example-tracing tutor

(versus bootstrapping). Finally, we wanted to validate our ideas about providing abstraction as part of the bootstrapping process.

First, our expert identified what he perceived to be likely student mistakes for this problem:

- leaving out levels of the hierarchies, such as connecting "Car" directly to "Vehicle," not going via "Motored Vehicle"
- defining multiple IS-A links to classes of a hierarchy path from one subclass, ignoring relation transitivity, e.g. connecting "Car" to both "Motored Vehicle" and "Vehicle" when "Motored Vehicle" is also modelled as a subclass of "Vehicle"
- providing incorrect directions of both IS-A and PART-OF links.

Given these common errors, a manually-defined example-tracing behavior graph would require at least one additional "buggy" branch for each composition link (i.e., for incorrect PART-OF link direction) and up to three additional "buggy" branches for each classification link (i.e., for missing hierarchy levels, incorrect transitivity, and incorrect IS-A link direction). However, our preliminary data suggests that, while students do make these types of errors, they are not as frequent as our expert suspected. More specifically, there appear to be particular objects that more naturally lead to these misconceptions, thus obviating the need to manually define buggy branches for *all* possible combinations. By bootstrapping data for particular problems, we could build a model that is more likely to be representative of real misconceptions, rather than what is suspected by an expert, and save effort in manually and comprehensively building a behavior graph from scratch.

Second, since most students are likely to re-arrange objects for this type of problem, especially in the disorganized condition, we asked our expert what types of re-arrangement techniques students would likely use to make sense of the problem. Our expert conjectured that students would take one of two approaches:

- "concept clustering" in the initial stage of the solution to organize the objects according to a conceptual schema, such as congregating all "Parts" close to one another
- "cleaning up" in the late stages of the problem solving to reduce the number of intersecting links or to make the visual arrangement easier to understand or aesthetically more attractive.

If our expert is correct, such a pre- and post-organization of the visual arrangement, and all permutations of those arrangements, would need to be added to each path of the behavior graph. This would be a considerable task to undertake in manual fashion. Thus, this appears to be an opportunity for abstraction techniques, in conjunction with bootstrapping, to make a difference. To reduce the number of possible variants, abstraction techniques that ignore exact screen positions and possible permutations of operations would benefit the designer of a collaboration tutor.

Finally, but most important for our work, aspects of collaboration and communication must be considered. The domain expert provided specific situations in which he assumed discussion and collaboration between students would occur:

- Choosing one of the major correct solution variants, i.e. connecting the most abstract concepts or the least abstract concepts
- Clarification of the conceptual understanding of some relations, e.g. if "Bike" has a "Steering Wheel" as a relaxed interpretation of the concept "Steering device"

The expert assumed discussion would happen after an action creating an edge. He explained that when removing edges, students tend to act before and not after an agreement. This was an interesting and surprising belief, since not all of our study groups followed this approach. Some of our subjects engaged in intense chatting before acting at the object level. Thus, the expert's assumption in this case might be viewed as an "expert blindspot" (Nathan et al., 2001); in the full specification of the example-tracing tutor he would not have specified discussion phases before domain actions. Again, the use of bootstrapping would provide the actual approach students take, rather than the approach perceived by an expert.

As a complexity analysis of the possible communication of collaborating students, we evaluated the complexity of a conversational schema, similiar to Winograd and Flores (1986). Our idealized discourse model for dyads only allowed a limited number of conversational acts for students: proposing an action, agreeing or disagreeing with the action, challenging and defending the action, and performing the action. Despite the constraints imposed by the schema, the number of possible communication sequences grows exponentially and thus cannot be, in practice, fully pre-defined by an expert. This is even more pronounced in the combination of possible workspace actions and natural conversation as used in our studies: in a graphical modeling problem, the combination of typical mistakes, graphical arrangement phases, and communication phases results in state explosion. Bootstrapping an example-tracing tutor, which will capture only the most realistic sequences of actions, is preferable to having an expert try to create one from scratch.

DISCUSSION AND FUTURE DIRECTIONS

In taking steps toward our vision of Bootstrapping Novice Data, we have learned some important lessons. In considering the results of the current project – and thinking about next steps – we asked ourselves the following questions.

- Were the five dimensions valid units of analysis for the collaboration in our studies?
- How might the BR be extended to better support collaborative analysis and model creation?
- How successful was our component-based integration and what remains to be done?
- Besides using the bootstrapped data to help build a tutor model, what other data analyses might be valuable?

In the following sections, we attempt to answer to these questions and, for the questions we cannot fully answer, briefly lay out a roadmap forward.

The Five Dimensions of Analysis

The dimensions did, indeed, provide a useful analysis framework. The conceptual understanding dimension was helpful in evaluating problem solutions; in both studies we were able to identify and rate the dyads based on salient (but different) conceptual features. Visual organization was important in both tasks, and appeared to inform problem solutions. The task coordination dimension provided the clearest tutoring guidelines of all the dimensions. The task coherence dimension provided information about object references in Study 1, but was not as clear of an aid in the analysis of Study 2. Finally, the task selection dimension was a useful measure in both studies, but was more valuable in Study 1 due to the greater number of possible strategies. In a nutshell, it appears that evaluating these five dimensions may be a useful strategy for tutoring collaboration of graphical modelling problems.

We limited our analysis to elements of the collaboration that (1) could be linked to solution quality, (2) could be operationalized based on patterns of group actions, and (3) could be displayed in the BR with a minimum of pre-processing. There is the potential to extend this analysis. We will use more sophisticated (and fine-grained) coding schemes for collaborative action, e.g. (Spada *et al*., 2005; Weinberger & Fischer, 2006), to analyze future data. The challenge would then be to translate these coding schemes into definitions the BR can recognize automatically and in real-time in order to pre-process actions and transform them into abstract actions that are coded in the particular scheme we use. We could also use data mining techniques to connect patterns of action to positive and negative solutions. For this approach, we will need to collect more data, and a way to interpret the results of this analysis so that we can add appropriate hints and bug messages. Finally, it may be appropriate to analyze sequences of actions both at the group level and at the individual level.

Of course, since we limited ourselves in the initial two studies to similar types of problems, the generality of our dimensions of analysis is unclear. On the other hand, with the exception of visual organization, it appears that the dimensions would be applicable to other collaborative problems as well. While general applicability of the dimensions is certainly an interesting topic of future research, our primary interest is the type of graphical modelling problems supported by Cool Modes and similar applications; thus, our studies will continue in this area, at least in the near term.

Extending the Behavior Recorder to Better Support Analysis and Model Creation

Extensions to the BR are clearly required to facilitate the BND approach, as discussed earlier. For instance, we will develop more sophisticated pre-processing techniques to provide abstracted, or aggregated, input to the Behavior Recorder. Besides the Goodman *et al*. (2005) and Soller & Lesgold (2003) approaches we have already discussed, Stevens & Soller (2005) have developed an interesting multi-layered approach to clustering, utilizing item response theory, artificial neural networks, and Hidden Markov Modelling, to analyze patterns in large amounts of subject data. In some of our own previous work, we investigated the problem of automatically identifying phases by aggregating similar types of actions (Harrer & Bollen, 2004) and may be able to leverage those techniques in the present work.

In deciding which specific data mining and clustering techniques to try, we will be guided by the collaborative analysis techniques discussed previously (Spada *et al*., 2005). In particular, for each of their nine dimensions (e.g., information pooling, coordinating communication, technical communication), we will attempt to identify representative action patterns for which our data mining and/or clustering approach will search. Taking an approach such as this will ground our technical method in a well-founded theoretical model, as Spada et al's approach relies on such models.

Besides fully automated techniques, we also envision that bootstrapping in the BR may benefit from a mixed-initiative approach, i.e., interleaving author actions with bootstrapping and analysis. For instance, one could imagine generalizing a bootstrapped behavior graph by manually defining appropriate feature detector functions and updating some of the edges in the graph appropriately. As a simple example, suppose we defined a "short" cluster function (**Is-Short-Cluster** (5)) which takes a path length 5 as a parameter and can identify a behavior graph path with less than or equal to 5 consecutive edges of the same action type. We could then replace an exact matching value on the corresponding edge in the behavior graph with the new **Is-Short-Cluster** function. This would allow us, in turn, to collapse branches in the behavior graph and increase the frequency counts on those branches.

To implement such a facility, we would need to develop a function library, a function library GUI, and techniques for collapsing states in the BR. We will explore techniques with real users and real data sets to understand precisely the types of mixed-initiative techniques that will be most useful during bootstrapping (or in a post-processing step) and what works to save development time. We also will experiment with using machine learning (ML) techniques to partially automate aspects of the mixed-initiative process. The general idea is to rely on the ML algorithm to induce the appropriate functions to apply to behavior graph edges, instead of relying on the author to select them. For instance, in the example given above, the author might label consecutive edges with the same skill and let the machine learning algorithm induce the appropriate matching function. A similar approach, designed to induce production rules from behavior graphs, has already been developed and is being tested at our lab at CMU (Matsuda, Cohen & Koedinger, 2005).

For non-abstracted evaluation of student actions, we also need to solve the problem of divergent solution paths for semantically similar sequences of actions. We intend to modify the BR to merge similar solution paths using an approximate, rather than exact, comparison between states. Each state in the behavior graph might be represented by a sequence of parameters. For example, a state in the "move" graph might be represented by values for link length, node cluster, link flow, and organization of rows, which are particular characteristics that we identified for visual organization. A move action would update the values of these parameters and the parameters would then be compared to other states on a qualitative basis. Combinations of parameters should be approximately equal to one another for two states to be declared the same. As with abstraction, we have done some previous work that may be helpful in

collapsing similar states (Bollen, Harrer, & Hoppe 2004). An important subproblem is that two paths with the same actions are taken as different from the point at which actions occur in a *different order*. This problem could also be addressed by providing more expressive ordering constraints to a CTAT author, in the case of paths with identical actions in a different order. We currently have only weak methods for dealing with path ordering. For instance, we cannot have an unordered group of ordered groups of actions. One approach is to define hierarchies of unordered and ordered groups of actions; we are exploring this approach already,. We will also need to develop a user interface to support authors in defining such hierarchies.

Finally, our initial studies showed us that the BR needs to be extended to handle dynamic instantiation of objects (e.g., the definition of new UML class nodes in Cool Modes). In particular, it must be capable of handling dynamic object definition across sessions to unify paths in the graph that diverge when the identity of objects is not recognized. Since Cool Modes uses a consistent, internal naming scheme for objects, we can leverage this to identify common dynamic objects across sessions using mapping tables.

Component-Based Integration

Another important lesson of our initial work relates to the component-based combination of Cool Modes and CTAT. In principle, doing plug-and-play integration should be straightforward, but in practice it can be quite challenging. Two tool characteristics identified in (Ritter & Koedinger, 1996) – *recordability* and *scriptability* – are essential. Being recordable means the tool is capable of capturing individual actions taken in its user interface (and not complete states). Cool Modes clearly meets the recordability criterion, as apparent from the current implementation and data capture described in this paper. Being scriptable, on the other hand, means the tool is capable of playing back user actions in its user interface, typically through a scripting language. Scriptability is desirable for real-time tutoring and necessary for convenient Behavior Recorder authoring. For example, to allow the author to immediately test the tutor at any step, the author should be able to select any state in the behavior graph, and the system should put the student interface into the corresponding configuration by replaying the actions along a path to that state. The MatchMaker server of Cool Modes, as described earlier and shown in Figure 4, is scriptable. One of our next steps will be to make the Cool Modes client capable of updating its interface based on the messages from the BR received via the tutor adapter and MatchMaker component of the integration architecture. This will also allow us to display in Cool Modes the hints and bug messages from the BR, so that on-the-fly student tutoring is supported by our technical integration.

Further Analysis and Use of Student Data

Besides using the student log files for building an initial version of a tutor, we envision other uses of the data. For instance, we plan to do problem profile analysis (Mark, 1998) in which we first have an expert, or group of experts, solve a problem using Cool Modes, record the steps in a behavior graph, and then have students attempt to solve the same problem, also recording the steps as in the BND approach described above. Assuming the expert solution(s) is the correct one, we can then use the resulting behavior graph to calculate the percentage of novice steps that diverge from the experts. The greater the divergence from expert behavior, the harder this problem can be assumed to be. Such data is helpful in designing and ordering problems in a curriculum, e.g., situating more difficult problems later in the curriculum. This expert-vs.-novice data can also be used to do skill proficiency analysis. Divergent paths that occur with a high frequency, according to the traversal counts collected by BND, likely relate to skills that should be the focus of additional problems and tutoring. If the author associates skills to edges of the behavior graph after BND data collection, we will have the data necessary to identify skills that, in general, cause the students more difficulty. The author could then design new problems that focus on these skills.

In order to improve an example-tracing tutor's behavior graph over time, we plan to do Learning Factors Analysis (Koedinger & Junker, 1999) with the Cool Modes log data. Learning Factors Analysis is a process whereby a cognitive model is evaluated and modified based on how closely student performance matches expected learning curves. The labelling of edges with skills – essentially the

"factors" – will allow us to subsequently check whether collaborating students gradually reduce their error rate. If the error rate for a particular skill does not yield a smooth, downward sloping learning curve, this is an indication that the skill has been mis-assigned to particular edges in the behavior graph. Eventually, we intend to extend CTAT to support authors in viewing and specifying alternative skill labellings, viewing the resulting learning curves, and making changes to the skill labels accordingly.

We see our approach possibly supporting simultaneous development of both individual and group models. Bootstrapping is designed to create a plausible collaboration model based on real data, and individual differences would not be explicitly represented in such a model. Yet the bootstrapping approach, in which a behavior graph is built, or extended, by each collaborating group, may also provide data to build student models with individual and collective characteristics (Paiva, 1997). For example, one student might perform excessive re-arrangement activities in an effort to conceptually organize a task: this is clearly an individual trait that could be supplied to a separate individual student model. On the other hand, the intensity and length of discussions may vary substantially from one group to the next, even with the same student participating, and thus may be information that should be represented in a group model. Separating and appropriately evaluating data for use in generating individual versus group models is a topic for our future research.

The creation of individual student models and/or collaborative models is also discussed in more detail in other articles in this issue: Even though the tasks supported by the medical tutoring system in (Suebnukarn & Haddawy, 2006) are very different, their approach shares many common traits: for each medical scenario or case, one specific domain model is created, somewhat like an example-tracing tutor. They use a Bayesian Network for the representations of domain and individual student models as well as group models to address uncertainty in the student input. The same computational representation has been chosen in (Read et al., 2006) for student and group modelling in the linguistic domain. Because the diagnosis cannot be guaranteed for natural language they combine their approach with interactive evaluation by more experienced students; this technique is comparable to our mixed-initiative approach of interactive creation and annotation of an example-tracing tutor.

For our analyses of collaboration scenarios, we deliberately chose a random assignment to the dyads without taking into account prior knowledge or learning styles. For our intended full version of collaborative tutoring with the BND approach, it is clearly a valuable option to assemble groups according to some group formation strategies, as are discussed in this issue in (Read et al., 2006) and (Alfonseca et al., 2006). These components could be plugged into our integration architecture to configure the setup of the collaboration session in Cool Modes.

CONCLUSION

Tackling the problem of tutoring a collaborative process is non-trivial. The tremendous variety of possible collaborative behavior makes the task quite difficult. Others have taken steps toward tutoring or coaching collaboration, but there are still challenges ahead.

For our part, we have been working on capturing and analyzing collaborative problem solving in a graphical modelling tool (Cool Modes) by capturing actions in the CTAT Behavior Recorder, a tool for building a special type of cognitive tutor called an example-tracing tutor. The technique we have devised, called "bootstrapping novice data", is based on the idea of recording problem solving behavior and then using the captured demonstration as the basis for creating a model for tutoring students. The work and empirical results we have presented in this paper have led us to the conclusion that, to reach our goal of tutoring collaboration, BR analysis needs to take place at multiple levels of abstraction. The use of techniques such as data mining and clustering in conjunction with bootstrapping – in a mixed-initiative mode with a tutor author – will be our next major area of investigation.

ACKNOWLEDGEMENTS

This research was initiated by NSF-DFG travel grants for German-American researcher exchange in 2004. We thank Kenneth R. Koedinger for his valuable contributions in the discussion of this project, Tilman Goehnert and Mike Schneider for their support in implementing this approach, and the students of the course "Modelling Techniques in Computer Science" in Duisburg for their voluntary participation in our two studies.

REFERENCES

Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. R. (2006). The Cognitive Tutor Authoring Tools (CTAT): Versatile and Increasingly Rapid Creation of Tutors; Accepted for presentation at the *8th International Conference on Intelligent Tutoring Systems*, Jhongli, Taiwan, June 26-30, 2006.

Aleven, V., Sewall, J., McLaren, B. M., & Koedinger, K. R. (2006). Rapid Authoring of Intelligent Tutors for Real-World and Experimental Use; Submitted to the *6th IEEE International Conference on Advanced Learning Technologies* (ICALT 2006) July 5 - 7, 2006 Kerkrade, The Netherlands.

Alfonseca, E., Carro, R.M., Martin, E., Ortigosa, A., & Paredes, P. (2006). The Impact of Learning Styles on Student Grouping for Collaborative Learning: A Case Study, in this issue.

Anderson, J. R., Corbett, A. T., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of Learning Sciences, Vol. 4,* pp. 167-207.

Aronson, E., Blaney, N., Stephan, C., Sikes, J., & Snapp, M (1978). *The Jigsaw Classroom*. Sage Publishing Company. Beverly Hills, CA.

Bollen, L., Harrer, A. & Hoppe, H. U. (2004). An Integrated Approach for Analysis-Based Report Generation. In Kinshuk, Chee-Kit Looi, Erkki Sutinen et.al. (eds): *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies* (ICALT 2004), Joensuu, Finland, pp. 1094-1095.

Bransford, J. D., Brown, A. L. & Cocking, R. R. (Eds.). (2000). *How people learn: Brain, mind, experience, and school.* Washington, DC: National Academy Press.

Clark, H. H. & Brennan, S. E. (1991). Grounding in Communication. In L. B. Resnik, J. M. Levine,& S. D. Teasley (Eds.), *Perspectives on socially shared cognition* (pp. 17–149). Washington, DC: American Psychological Association.

Corbett, A., McLaughlin, M., & Scarpinatto, K.C. (2000). Modelling Student Knowledge: Cognitive Tutors in High School and College. *User Modelling and User-Adapted Interaction*, 10, 81-108.

Constantino-González, M.A., Suthers, D.D. & Escamilla de los Santos, J.G. (2003). Coaching Web-Based Collaborative Learning Based on Problem Solution Differences and Participation. *International Journal of Artificial Intelligence in Education*, Vol. 13, 263 - 299.

Constantino-González, M. A. & Suthers, D. D. (2002). Coaching Collaboration in a Computer-Mediated Learning Environment. In the *Proceedings of the Conference on Computer Supported Collaborative Learning* (CSCL-02)

Gasevic, D. & Devedzic, V. (2003). Software Support for Teaching Petri Nets: P3. In the *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies*, Athens, Greece, 2003, pp. 300-301

Goodman, B. A., Linton, F. N., Gaimari, R. D., Hitzeman, J. M., Ross, H. J. & Zarrella, G. (2005). Using Dialogue Features to Predict Trouble During Collaborative Learning. *User Modelling and User-Adapted Interaction*, 15:85-134. Springer.

Harrer, A. & Bollen, L. (2004). Klassifizierung und Analyze von Aktionen in Modellierungswerkzeugen zur Lernerunterstützung. In Workshop-Proceedings Modellierung 2004. Marburg, 2004.

Harrer, A., McLaren, B. M., Walker, E., Bollen, L., & Sewall, J. (2005). Collaboration and Cognitive Tutoring: Integration, Empirical Results, and Future Directions; In the *Proceedings of the 12th International Conference on Artificial Intelligence and Education* (AIED-05), Amsterdam, the Netherlands, July 2005.

Heiner, C., Beck, J. E. & Mostow, J. (2004). Lessons on using ITS data to answer educational research questions. In the *Proceedings of the ITS2004 Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes*, pp. 1 - 9.

Hoppe, H.U., Gassner, K., Mühlenbrock, M.,& Tewissen, F. (2000). Distributed visual language environments for cooperation and learning - applications and intelligent support. *Group Decision and Negotiation 9* (3), 205-220.

Introne, J. & Alterman, R. (2006). Using Shared Representations to Improve Coordination and Intent Inference, this issue.

Jansen, M. (2003) Matchmaker - a framework to support collaborative java applications. In the *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AIED-03)*, IOS Press, Amsterdam.

Johnson, D. W. & Johnson, R. T. (1990). Cooperative learning and achievement. In S. Sharan (Ed.), *Cooperative learning: Theory and research* (pp. 23-37). New York: Praeger.

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B. M., & Hockenberry, M. (2004) Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration. In the *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS-2004)*, Maceio, Brazil.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education, 8*, 30-43.

Koedinger, K. R. & Junker, B. (1999). Learning Factors Analysis: Mining student-tutor interactions to optimize instruction. Presented at *Social Science Data Infrastructure Conference*. New York University. November, 12-13, 1999.

Koedinger, K. R. & Terao, A. (2002). A cognitive task analysis of using pictures to support pre-algebraic reasoning. In C. D. Schunn & W. Gray (Eds.), In the *Proceedings of the 24th Annual Conference of the Cognitive Science Society*, 542-547.

Lesgold, A., Katz, S., Greenberg, L., Hughes, E., & Eggan, G. (1992) Extensions of Intelligent Tutoring Paradigms to Support Collaborative Learning. In S. Dijkstra, H. Krammer, & J. van Merrienboer (Eds.), *Instructional Models in Computer-Based Learning Environments*. Berlin: Springer-Verlag, 291-311.

Lesgold, A. M., Lajoie, S. P., Bunzo, M., and Eggan, G. (1993). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In J. Larkin, R. Chabay (Eds.), *Computer assisted instruction and intelligent tutoring systems*. Hillsdale, NJ: LEA.

Lieberman, H. (ed) (2001). *Your Wish is My Command: Programming by Example*. Morgan-Kauffman Publishers.

Mark, M. (1998). *Analysis of Protocol Files: PACT Center User's Manual*. Carnegie Mellon University.

Martínez, A., Dimitriadis, Y., Rubia, B., Gómez, E., & de la Fuente, P. (2003). Combining qualitative evaluation and social network analysis for the study of classroom social interactions, Workshop proceedings "Documenting collaborative interactions" in *Computers and Education, special issue on Documenting Collaborative Interactions*.

Matsuda, N., Cohen, W. W., & Koedinger, K. R. (2005). Building Cognitive Tutors with Programming by Demonstration. In S. Kramer & B. Pfahringer (Eds.), Technical report: TUM-I0510 (*Proceedings of the International Conference on Inductive Logic Programming*) (pp. 41-46): Institut fur Informatik, Technische Universitat Munchen.

McArthur, D., Lewis, M. W., & Bishay, M. (1996). ESSCOTS for learning: Transforming commercial software into powerful educational tools. *Journal of Artificial Intelligence in Education*, 6 (1), 3-33.

McLaren, B. M., Bollen, L., Walker, E., Harrer, A., & Sewall, J. (2005). Cognitive Tutoring of Collaboration: Developmental and Empirical Steps Toward Realization;, In the *Proceedings of the Conference on Computer Supported Collaborative Learning* (CSCL-05), Taipei, Taiwan in May/June 2005.

McLaren, B. M., Koedinger, K. R., Schneider, M., Harrer, A., & Bollen, L. (2004a). Bootstrapping Novice Data: Semi-Automated Tutor Authoring Using Student Log Files; In the *Proceedings of the ITS2004 Workshop on Analyzing Student-Tutor Interaction Logs to Improve Educational Outcomes*.

McLaren, B. M., Koedinger, K. R., Schneider, M., Harrer, A., & Bollen, L. (2004b). Toward Cognitive Tutoring in a Collaborative, Web-Based Environment; In: M. Matera, S. Comai (Eds.), *Engineering Advanced Web Applications*, Rinton Press, Princeton, NJ., pp. 167-179.

McManus, M. & Aiken, R. (1995). Monitoring Computer-Based Problem Solving. *Journal of Artificial Intelligence in Education, 6(4)*, 307-336.

Merceron, A. & Yacef, K. (2005). TADA-Ed for Educational Data Mining, *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, Volume 7, Number 1, May 2005. http://imej.wfu.edu/articles/2005/1/03/index.asp.

Mühlenbrock, M. & Hoppe, H. U. (2001). A Collaboration Monitor for Shared Workspaces. In the *Proceedings of the International Conference on Artificial Intelligence in Education* (AIED-2001).

Murray, T., Ainsworth, S., & Blessing, S. (eds.) (2003). *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective, Adaptive, Interactive, and Intelligent Educational Software*. Kluwer Academic Publishers. Printed in the Netherlands.

Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, Vol. 10, pp. 98-129.

Nathan, M., Koedinger, K., & Alibali, M. (2001). Expert blind spot: When content knowledge eclipses pedagogical content knowledge. Paper presented at the *Annual Meeting of the American Educational Research Association*, Seattle.
Paiva, A. (1997). Learner Modelling for Collaborative Learning Environments. In *Proceedings of the 8th International Conference on Artificial Intelligence in Education*, pp. 215-222. Kobe (Japan), August 1997.
Pinkwart, N., Hoppe, H.U., Bollen, L., & Fuhlrott, E. (2002). Group-Oriented Modelling Tools with Heterogeneous Semantics. In S. Cerri, G. Gouarderes, F. Paraguacu (Eds.), In the *Proceedings of the 7th International Conference on Intelligent Tutoring Systems (ITS-2004)*, Maceio, Brazil. Springer, Berlin, pp. 21-30.
Pinkwart, N. (2005). Collaborative Modeling in Graph Based Environments. Berlin, Germany: dissertation.de – Verlag im Internet.
Pinkwart, N. (2003) A Plug-In Architecture for Graph Based Collaborative Modelling Systems. In U. Hoppe, F. Verdejo & J. Kay (eds.): In the *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AIED-03)*, 535-536.
Polson, M. C. & Richardson, J. J. (1988). *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates Publishers.
Read, T., Barros, B., Barcena, E. & Pancorbo, J. (2006). Coalescing Individual and Collaborative Learning to Model User Linguistic Competences, this issue.
Ritter, S. & Koedinger, K. R. (1996). An Architecture For Plug-In Tutor Agents. In: *Journal of Artificial Intelligence in Education*, 7 (3 / 4), 315-347.
Searle, J. (1969). *Dialogue Acts: An Essay in the Philosophy of Language*. London: Cambridge University Press.
Slavin, R. E. (1992). When and why does cooperative learning increase achievement? Theoretical and empirical perspectives. In R. Hertz-Lazarowitz & N. Miller (Eds.), *Interaction in cooperative groups: The theoretical anatomy of group learning* (pp. 145-173). New York: Cambridge University Press.
Soller, A., & Lesgold, A. (2003). A Computational Approach to Analyzing Online Knowledge Sharing Interaction. In the *Proceedings of the 11th International Conference on Artificial Intelligence in Education (AIED-03)*, Sydney, Australia, 253-260.
Spada, H., Meier, A., Rummel, N., & Hauser, S. (2005). A New Method to Assess the Quality of Collaborative Process in CSCL. In T. Koschmann, D. Suthers, & T. W. Chan (Eds.), In the *Proceedings of the Conference on Computer Supported Collaborative Learning* (CSCL-05), Taipei, Taiwan in May/June 2005. pp. 622-631. Mahwah, NJ: Lawrence Erlbaum.
Stasser, G. & Titus, W. (1985). Pooling of Unshared Information in Group Decision Making: Biased Information Sampling during Discussion. *Journal of Personality and Social Psychology*, *48*, 1467-1478.
Stevens, R. & Soller, A. (2005). Implementing a Layered Analytic Approach for Real-Time Modelling of Students' Scientific Understanding. In the *Proceedings of the 12th International Conference on Artificial Intelligence and Education (AIED-05),* Amsterdam, the Netherlands, July 2005.
Suebnukarn, S. & Haddawy, P. (2006). Modeling Individual and Collaborative Problem-Solving in Medical Problem-Based Learning, this issue.
Suthers, D. D. (2003). Representational Guidance for Collaborative Learning. In the *Proceedings of Artificial Intelligence in Education* (AIED-03), IOS Press, Amsterdam.
Vizcaino, A. (2005). A Simulated Student Can Improve Collaborative Learning. *International Journal of Artificial Intelligence in Education*, 15 (2005) 3-40, IOS Press.
Walker, E., Koedinger, K. R., McLaren, B. M. and Rummel, N. (2006). Cognitive Tutors as Research Platforms: Extending an Established Tutoring System for Collaborative and Metacognitive Experimentation; Accepted for presentation at the *8th International Conference on Intelligent Tutoring Systems*, Jhongli, Taiwan, June 26-30, 2006.
Weinberger, A., & Fischer, F. (2006). A Framework to Analyze Argumentative Knowledge Construction in Computer-Supported Collaborative Learning. *Computers & Education*, 46, 71-95.
Wenger, E. (1987) *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers, Inc.
Winograd, T. & Flores, F. (1986). *Understanding Computers and Cognition - A new Foundation for Design*. New Jersey, Ablex Publishing Comp.

Vitae / Biographies

Dr. Andreas Harrer is senior research assistant at the Department of Computer Science at Universität Duisburg-Essen in Germany. He received his Diploma and Doctoral Title in Computer Science at Technische Universität München. His main areas of interest are in computer-supported collaborative learning, intelligent tutoring and learner support, and the development of educational

systems from a software engineering perspective. He is speaker of the special interest group "Artificial Intelligence and Education" of the "Kaleidoscope" European Network of Excellence. The joint research in this article combines these research fields and was initiated by a DFG-NSF researcher exchange program.

Dr. Bruce McLaren is a systems scientist at Carnegie Mellon University and the Pittsburgh Science of Learning Center in the United States. Dr. McLaren is focused on developing educational technologies to support metacognitive learning skills, such as knowing how to collaborate and knowing when to seek help. He co-manages a team of 8 programmers and research associates in the development and enhancement of the Cognitive Tutor Authoring Tools (CTAT). Dr. McLaren holds a Ph.D. and M.S. in Intelligent Systems from the University of Pittsburgh, an M.S. in Computer Science from the University of Pittsburgh, and a B.S. in Computer Science (*cum laude*) from Millersville University.

Erin Walker is a Ph.D. candidate in Human Computer Interaction at Carnegie Mellon University. She received her B.Sc. in Computer Science and Psychology from the University of Manitoba in 2004. Her primary interests lie in the areas of computer-supported collaborative learning and cognitive tutoring in ill-defined domains.

Lars Bollen holds a high school teachers degree (Master of Science equivalent) for computer science and physics from University Duisburg-Essen. In 2002, he joined the Collide research group as a research and teaching assistant. He has been involved in various projects on the national and international level in the field of CSCL, wireless and mobile technologies in education and collaboration analysis. His PhD thesis research area is the computer-based support for teachers by automatic reporting facilities from collaborative learning processes, utilizing action and collaboration analysis.

Jonathan Sewall is a project director on the CTAT project at the Human-Computer Interaction Institute of Carnegie Mellon University. He holds a B.A. degree from Dartmouth College in Classics and a Master of Computer Science from Rice University. Before coming to Carnegie Mellon he held several positions in government and industry developing or extending systems having to do with low-level data communications protocols, wide-area networking, distributed systems, data base management, and planning and scheduling.