

## Chapter 7

# **ANALYZING USER ACTIVITY AND GROUP INTERACTION IN SHARED WORKSPACES**

Martin Muehlenbrock  
*Gerhard-Mercator-University Duisburg, Germany*

### **1. SHARED WORKSPACE SYSTEMS**

In recent years, shared workspace systems have been an active area of research and development in computer-supported collaborative learning. A particular type of shared workspace systems provides graph-oriented visual representations for synchronous interaction, which are well suited for co-constructive activities. This type of systems is used in face-to-face settings as well as in distant settings with additional communication means such as chat tools or video conferencing systems. Simply speaking, a shared workspace is an electronic version of a sheet of paper or a blackboard that is jointly used by a group of people for writing and drawing. However, shared workspaces have a number of additional features in comparison to paper and pencil representations. On the one hand, there are general features such as computer-based storing and retrieving of shared workspaces. On the other hand, shared workspaces can provide some sort of component-based structure, which is visualized graphically and can be modified flexibly by direct manipulation. A learning situation is promoted by the provision of problem-related workspace material that has a potential for stimulating discussions and for supporting the objectification of inherent problem structures. This means that in collaborative learning, the activity is joint problem solving, and learning is expected to occur as a side effect (Roschelle & Teasley, 1995).

One of the first shared-workspace systems that has been developed for collaborative learning is the system C-Chene, which provides a graphical structure for the joint modeling of energy chains (Baker & Lund, 1996). The shared workspace provides components that represent domain concepts such as sources and drains of energy as well as links between components that are to represent the flow of energy. This system is designed for a specific problem, which is reflected by the closed sets of links and symbolic representations for domain objects. It can be regarded as a specific realization of a concept-mapping tool (Novak, 1990; Cicognani, 2000) for the domain of energy chains. Another well-known shared-workspace system for collaborative learning is the system Belvedere (Suthers, Weiner, Connelly, & Paolucci, 1995). Belvedere provides labeled components for textual statements and links to represent argumentative relations. In contrast to C-Chene, in Belvedere the shared workspace is not used to jointly model domain relations but to collaborate in a scientific inquiry process. Both C-Chene and Belvedere provide component-based structures for specific problems. They can be regarded as specific instances of distributed graphical environments that provide generic visual languages for collaboration (Lakin, 1990; Stefik, Foster, Bobrow, Kahn, Lanning, & Suchman, 1987).

The application CardBoard is a generic shared workspace system that features two-dimensional visual representations (Hoppe, Gassner, Muehlenbrock, & Tewissen, 2000). These visual languages consist of pre-defined typed objects, so-called cards, which contain information in the form of text or images. Users can freely add cards to a workspace from a palette, or drag and drop cards from a workspace to another one. Cards can as well be moved within the workspace or be deleted. Furthermore, card nets can be constructed by connecting cards with graphical links. *Figure 1* shows the CardBoard user interface with two shared workspaces (top left and bottom right), a palette with predefined cards (bottom left), and a popup menu to select a visual language for a new workspace. In addition to shared workspaces, each user can create private workspaces for individual problem solving, before entering a collaborative session in which the users combine their individual results in a shared workspace. An agent-based framework system adds components for problem-specific interpretations of card nets and intelligent support functions (Muehlenbrock, Tewissen, & Hoppe, 1998). As a benefit, the computerized form of traditional paper and pencil-based collaboration not only allows for a better storage and re-use of intermediate and final results of the joint problem solving, but it also makes possible an automatic monitoring and analysis of the group interaction in the shared workspace.

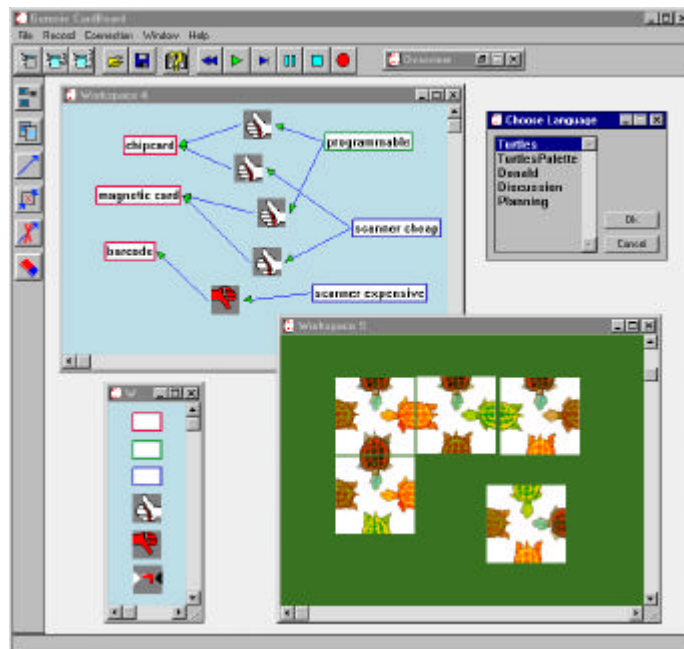


Figure 1. Shared workspace application CardBoard.

## 2. AUTOMATIC ANALYSIS OF COLLABORATIVE ACTIVITIES

When users communicate and collaborate via a computer system, the system usually has no kind of awareness of the interaction that is going on. This means that information on the communication and collaboration is not accessible by the system in terms of a descriptive representation. However, such a descriptive representation is a necessary pre-requisite for any kind of intelligent system support. For instance in audio and video conferencing, the system usually is unaware of the course of interaction that it broadcasts. In chat interfaces with text input, the data is in a machine-readable

format and could in principle be used for an automatic interpretation, but up to now few have bothered with the complexities of natural language processing. Instead, some collaborative learning systems use additional descriptive labels to characterize textual contributions. However, a problem is that the user has to provide these labels, and which necessitates an additional action in the course of the creation of a text message. For instance, for each contribution users are asked to select a label from a list of labels such as proposal, contra-proposal, question, comment, clarification, and agreement (Barros & Verdejo, 1999), or to fill in incomplete sentences (Harrer, 2000), or text input is required in a formal language (Tedesco & Self, 2000). A more subtle characterization of contributions can be done by providing sentence openers that ease text typing and that are related to speech acts (Soller, Linton, Goodman, & Lesgold, 1999). However, a general problem with having the user selecting a label for his contribution is that this additional step is a potential source of error and misclassification, e.g. when labels are used other than intended by the system developer (Robertson, Good, & Pain, 1998).

Shared workspaces with structured graphical representations are frequently used to enrich chat or face-to-face communication. It has been observed for a chat interface combined with a shared workspace that task-related communication via the chat interface was progressively reduced to a minimum and users' interaction appeared to be mostly action-based through the shared workspace (Dillenbourg & Baker, 1996). For instance, two users can disagree by controversial utterances or by opposed actions in the shared workspace. Jointly building up and maintaining shared workspace representations is regarded as being helpful for the externalization of problem related and possibly conflicting conceptions (Roschelle & Teasley, 1995). However, most research concerning shared-workspace interaction has *not* investigated the collaboration in the shared workspace itself, but has focused on analyzing the verbal interaction that additionally took place (Veerman, 2000, Derry & DuRussel, 1999).

A primary goal of automatically analyzing collaboration in shared workspaces is to provide an advanced monitoring of group interactions, which is a main item on the CSCL research agenda (Dillenbourg, 1999). There are two different ways in which the result of the analysis could also be used in an automatic way to influence the learning (Jermann, Soller, & Muehlenbrock, 2001). On the one hand, the protocol with intelligible higher-level information about the course of the interaction could be displayed to the users in order to stimulate self-reflection and meta-level discussions (Hoppe & Ploetzner, 1999). On the other hand, the information could be used to guide the learning process by means of suitable system interventions. For this remediation to be beneficial, the system has to unambiguously identify the learning problem and to incorporate pedagogic knowledge for the remediation. In the approach presented here, we pursue a conservative strategy that leaves intervention to human interpretation, be it a tutor or the group of learners themselves (Muehlenbrock, Tewissen, & Hoppe, 1998).

A typical example of a problem solving setting with two users in a face-to-face situation is shown in *Figure 2*. The user interfaces are comprised of private and shared workspaces, and the users collaborate by moving files from their private workspaces with initially different subsets of necessary cards to the shared workspace and by freely arranging the cards in the shared workspace to solve the puzzle in form of a three times three square. For this task there are a number of sub-optimal solutions in which most but some cards match. Therefore, in the course of the problem solving the users are frequently faced with impasses and have to coordinate their individual steps in solving the problem in order to come to a solution. This puzzle problem is only one example of a variety of tasks that can be represented by visual languages in the shared workspace application. Other worked examples include visual languages for discussion, planning, and rule-based agent programming among others.

*Figure 3* shows a sequence of actions in the shared workspace in the course of some problem solving. In this sequence, the actors take turns in adding further objects and in re-arranging

existing objects. Each action by an individual user can be regarded in the context of preceding and subsequent actions by the same user or by another user. Some actions form sequences of actions that solve some sub-task. In addition, group interactions can be interpreted as a coordination or a conflict. Coordinated actions in constructing a problem solution indicate mutual agreement, whereas conflicts indicate different strategies or different point of views that may initiate negotiation to maintain a shared understanding (Doise & Mugny, 1984). In fact, a major characteristic of the co-constructive problem solving is the negotiation of different strategies and solutions. For instance, users can disagree by uttering contradictory verbal statements, but also by activating opposed commands. That is, negotiating the next action can be performed through discourse or by simply doing it (Dillenbourg, Baker, Blaye, & O'Malley, 1995).

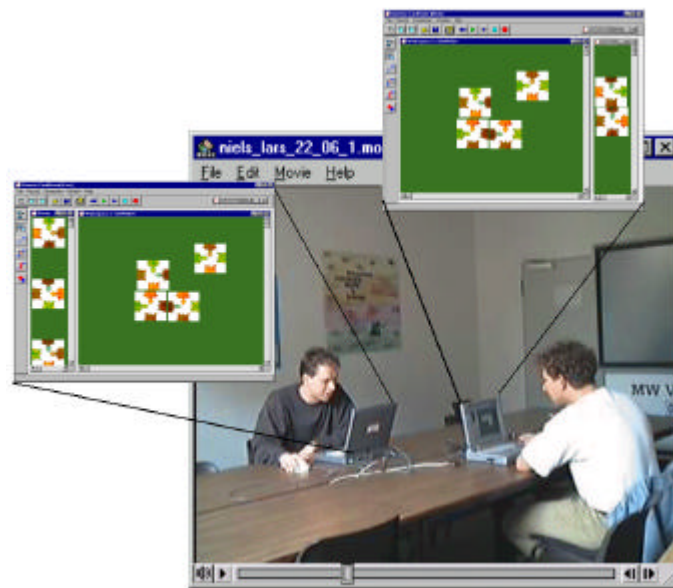


Figure 2. Setting with two users working together to solve a puzzle problem by using the shared-workspace application with one synchronised workspace and a different private workspace each.

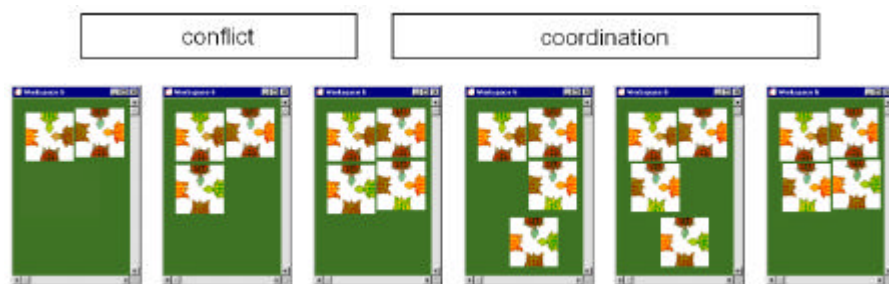


Figure 3. Excerpt of a collaborative problem solving interaction.

In the example in *Figure 3*, sequences of actions can thus be interpreted as exhibiting conflicts and coordinations between the users. At first, each user adds an object. The two new cards match with the existing arrangement of cards, but they do not match with each other. Therefore these two

actions are in conflict with each other. To overcome the conflict, one user removes the card that has just been added by the other user. Then both users re-arrange cards in order to revise the partial solution, which can easily be interpreted as a coordination.

### 3. A FRAMEWORK SYSTEM FOR INTELLIGENT SUPPORT

From a systems engineering point of view, automatic processing and analysis should be clearly separated from the group-interactive functionality and particularly from the user interface. Whereas *interactive technology* will typically make use of UI and windowing toolkits, e.g. implemented on a Java platform, automatic processing or *intelligent functions* may benefit from using rule-based inference engines, e.g. based on Prolog. A unifying principle for such heterogeneous architectures is the notion of “agents”: We can conceive a groupware environment which is combined with intelligent analysis mechanisms and support functions as an ensemble of interacting human and machine agents (Muehlenbrock, Tewissen, & Hoppe, 1998).

Reasonable feedback from the system to the users can only be given if the signature of a visual language, i.e., its syntactical structure, is extended with a mechanism for the interpretation of its card symbols. A general mechanism is providing operational semantics for types of card nets, i.e. an inference engine assigns operations to symbols introduced by some visual language specification. This approach for operational semantics serves the following goals:

- Rules or constraints can impose restrictions on the combination of cards or help resolving conflicts.
- Semantic values can be represented by card attributes such as colour, shape or content. These values are propagated on the basis of the types of linked cards. On this basis, the consequences of card manipulations can be shown to the user.
- The interpretation of cards in terms of propositional logic or first order logic prepares the ground for standard student modelling techniques and intelligent support.
- In a self-referential manner, a visual language can be interpreted in such a way that it modifies the interpretation of another language at runtime.

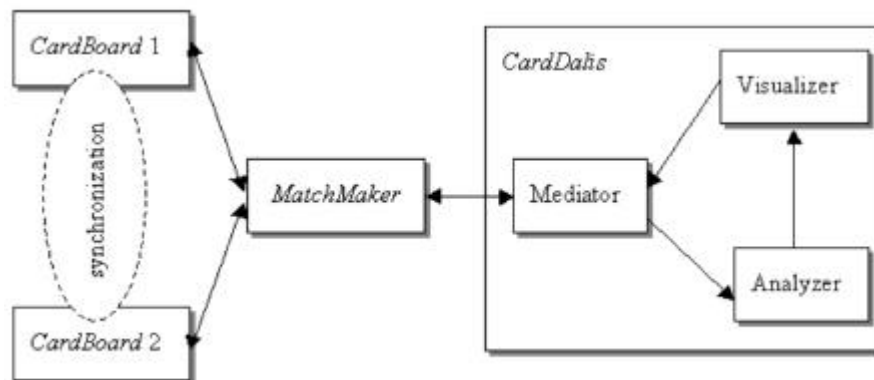


Figure 4. Framework system with plug-in support components for intelligent support.

Operational semantics and intelligent support are added to a shared-workspace application by using a framework system called CardDalis (Muehlenbrock, Tewissen, & Hoppe, 1998). Different components for interpretation and support can be plugged in as agents into the distributed CardDalis system. This is comparable to the approach of Ritter and Koedinger (1996) for incorporating tutoring elements into pre-existing software packages. The CardDalis architecture is

based on the design principle of mirroring the topology and interactivity of the real learning group by a Prolog-based agent system. It is composed of specialised agents for monitoring and modelling the human participants. It is quite simple to add arbitrary Prolog programs as new types of agents to the environment. CardDalis establishes a federation architecture that facilitates the message flow between its internal agents and the shared-workspace applications that are connected to a central synchronisation server called MatchMaker (see *Figure 4*).

Within this framework system, three kinds of agents have been developed particularly for shared workspaces with visual languages: mediator, analyser, and visualizer agents (see *Figure 4*):

- The **mediator** relates the visual card language to the underlying interpretation by translating card nets into specific interpreter representations and vice versa. In addition, it aggregates and explicates information implicit in the user interface, such as time of user actions and spatial card structure.
- The **analyzer** provides operational semantics for the card environments and tracks the user interaction in the shared workspace. It relates lower-level actions to higher level activities that are relevant in terms of the collaboration.
- The **visualizer** displays results of the analyser in the shared workspace by temporarily inserting additional cards and links to existing cards.

The shared-workspace application and the mediator communicate by means of basic message primitives for the creation, deletion and modification of card objects. Each message specifies the identifier of the workspace object and the identifier of the user who performed the particular action. Further arguments relate to object-specific attributes. The mediator monitors continuously all the modifications in the card workspaces and incrementally reconstructs a descriptive representation of the state of the card environment. Conversely, the visualizer produces messages that result in the addition or deletion of cards and links for indicating interaction analyses.

#### 4. ACTIVITY RECOGNITION

For analyzing collaborative problem solving activities in shared workspaces, a process-oriented perspective has been adopted for providing feedback to the learning group and to stimulate self-reflection. In the case of individual problem solving in an interactive learning environment, a process-oriented learner model has shown to be useful for providing intelligent support in open tasks (Akhras & Self, 1996).

The analysis approach is based on the idea of tracking user actions and indicating specific patterns of activity and interaction. *Activity recognition* (Muehlenbrock, 2001) is formally based on the situation calculus and on approaches for plan and task recognition (Kautz, 1990, Hoppe, 1993). The recognition and interpretation of user and group actions is organized hierarchically, starting from basic actions in the shared workspaces, and successively deriving higher-level activity from these. This is achieved by observing user actions in the context of the problem-solving product as well as by relating actions to preceding and potentially subsequent actions. Typical basic actions in card workspaces are the creation, modification, and deletion of cards and connections between cards. Taking a stream of action messages from the shared workspace as an input, the activity recognition automatically and incrementally infers more abstract concepts related to group activities and interpretations of conflicting and coordinated action sequences.

In activity recognition, actions are conceptualized as occurring in the context of some state and in the context of other actions. These relationships are specified by means of operators. An operator is a tuple that defines an action

*operator*(*<action>*)

*pres:*            <*preconditions*>  
*posts:*           <*postconditions*>  
*backgrs:*        <*backgrounds*>  
*decomps:*       <*decompositions*>  
*constrs:*        <*constraints*>

Preconditions of an operator are prerequisites that need to hold before the action in the header can actually be applied. Postconditions are effects that hold after the action has been successfully executed. Backgrounds are statements that refer to additional background information. Decompositions refer to other actions, i.e. an action may be complex by being decomposed into basic actions and complex actions. Also associated with each operator is a set of applicability conditions called constraints.

These operators are applied in the following way: Taking a stream of action messages from the shared workspace and a set of operators as an input, activity recognition automatically and incrementally infers abstract notions of group activity and interpretations of problem-related conflicts and coordinations. Activity recognition maintains an explicit description of the current workspace state and a set of action sequences that have potentially been initiated but have not yet been completed. The approach does not rely on domain or task knowledge, though it can use possibly available background information to complement its results.

A shared workspace with a visual language is conceptualized as a collection of objects and directed edges between these objects. Typical basic actions like the creation, modification, and deletion of objects and links are represented e.g. as

*create\_object(object8, turtle, pos(187, 163), dim(89, 87), 'Andreas')*  
*modify\_pos(object2, pos(240, 24), 'Andreas')*  
*delete\_object(object8, 'Lars')*  
*create\_link(before, object13, object 3, 'Lars')*  
*delete\_link(before, object13, object 3, 'Andreas')*

All actions relate to corresponding actions schemas, which group types of actions under the heading of predicate names. In contrast to approaches with only propositional expressiveness such as Kautz (1990), the relational representation covers a potentially infinite number of actual actions and greatly simplifies the formulations of rules for the analysis. The corresponding action schemas for the actions above are (with capital initial letters indicating variables):

*create\_object(O, Type, pos(Xpos, Ypos), dim(Xdim, Ydim), Actor)*  
*modify\_pos(O, pos(Xpos, Ypos), Actor)*  
*delete\_object(O, Actor)*  
*create\_link(Label, O1, O2, Actor)*  
*delete\_link(Label, O1, O2, Actor)*

For each action schema (for simplification reasons also called *action* in the sequel), there are one or more operators defining its preconditions, postconditions, backgrounds, and constraints. Operators for higher-level actions also include decompositions to other actions, whereas operators for basic actions include decompositions to external actions, i.e. workspace messages. The first action listed above, i.e. the creation of a new object with a given type, position, and dimension by a certain actor, is for instance defined by the following operator

*operator(create\_object(O, Type, Pos, Dim, Actor))*

```

pres:      not object(O, Type, Pos1, Dim1)
posts:     object(O, Type, Pos, Dim)
decomps:   external_create_object(O, Type, Pos, Dim, Actor))

```

The operator specifies that before the action is performed the object does not exist (precondition), that afterwards the object exists, and that it relates to (is triggered by) an external action, i.e. a message send from the shared workspace. The second action in the list above is the movement of an object to another position, which is defined by an operator as

```

operator(modify_pos(O, Pos, Actor)
pres:      object(O, Type, Pos1, Dim)
posts:     not object(O, Type, Pos1, Dim),
           object(O, Type, Pos, Dim))
decomps:   external_modify_pos(O, Pos, Actor))

```

The precondition states that the respective object has to already exist before the action is performed, and the postconditions state that the object is no longer at position *Pos1*, but at position *Pos*. Both *Pos1* and *Pos* are variables, which means that this single operator covers all movements of objects on all potential positions in the workspace.

The definition of basic actions such as these provides the ground for specifying more complex actions. For instance, by arranging cards close to each other, an implicit relation between these is created by the user. This is expressed by the following operator, which incorporates the background knowledge that these objects can in fact be related by spatial arrangement (predicate *composable*). Moreover, there are further operators for establishing relations between objects, e.g. by creating a visual link.

```

operator(create_relation(Dir, O1, O2, [], Actor)
pres:      object(O1, Type1, Pos3, Dim1),
           object(O2, Type2, Pos2, Dim2)
posts:     relation(Dir, O1, O2, [])
backgrs:   composable(Type1),
           composable(Type2)
decomps:   modify_pos(O1, Pos1, Actor)
constrs:   adjacent(Dir, Pos1, Dim1, Pos2, Dim2))

```

Having defined what creating (and deleting) a relation means, an action can be defined that incorporates a sequence of two actions with the first one creating a relation and a second one deleting the very same relation (expressed by identical variable names), which descriptively specifies an immediate rejection of the relation, i.e.

```

operator(immediate_relation_rejection(Functor, [Actor1, Actor2])
decomps:   create_relation(Functor, O1, O2, _, Actor1),
           delete_relation(Functor, O1, O2, _, Actor2))

```

Finally, this actions sequence can be interpreted as a conflict if two different users have performed the actions of creating and deleting the relations, i.e.

```

operator(interaction_conflict(immediate_relation_rejection, Actors)
decomps:   immediate_relation_rejection(Actors),

```

*constrs: different(Actors))*

These are some simple examples of operators, which are the basic building blocks for the activity recognition. A number of operators have been pre-defined in form of an activity library for shared-workspace analysis.

## 5. ACTIVITY LIBRARY

By composing actions into higher-level activities, a hierarchy of actions for the analysis of shared-workspace interaction has been developed. This activity model is composed of six levels, with actions on each level being defined by actions on lower levels:

- **External actions:** creation, deletion, and modification of any kind of workspace object, together with their parameters such as position, size, and content (interface to the shared-workspace application).
- **Basic actions:** identification of relevant actions and some of their consequences, e.g. the deletion of an object implicitly also deletes all its connections to other objects.
- **Relation actions:** they represent the underlying structuring principle of the workspace such as relating cards by direct connections, by using special connector cards, or by spatial adjacency.
- **Phase actions:** they indicate uninterrupted sequences of actions that are constructive or deconstructive in nature concerning the combination of cards and indicate an aggregation or revision phase in the problem solving.
- **Transrelation actions:** they describe the effects of actions on the transitive closure of relations, and also indicate the creation or deletion of circular structures.
- **Sequence actions:** they represent multi-user problem-related action sequences, e.g. successive actions performed by different users that reveal an ambiguity in the sub goal that is to be achieved next.
- **Interaction actions:** they interpret group-related action sequences in terms of conflicting or coordinated problem solving activities and indicate the initiator of the interaction sequence.

The library defines an overall number of 32 different actions by providing more than 80 operators. This hierarchical activity model does not strive for an unambiguous analysis of the workspace interaction. It is possible that more than one analysis is produced by the system, which is seen as to stimulate further interpretations and discussions among the learners and with their teacher.

The activity library abstracts from specific problems by using action schemas and background knowledge is a general mechanism for incorporating task specific properties. In addition to the problem of solving the turtle puzzle, it has been used to analyze collaboration in solving other puzzles, but also to analyze interaction by using visual languages specific for discussion tasks and for planning tasks (Muehlenbrock, 2001). An example interaction from a planning task is shown in *Figure 5*. In the course of activity, the actors take turns in adding and deleting objects and relations in the shared workspace. The example interaction includes sequences of actions that can be interpreted as exhibiting conflicts between the group members. First, one user adds a link that contradicts the link created by the other user, then the other user deletes an object and thereby also the link created by the first user.

The system provides different ways of visualizing the analysis results. One option for the visualizer is to directly act in the workspace in which the collaboration takes place. Therefore the visualizer uses the same type of actions that are used by the users, but potentially with card types adapted to giving feedback such as text cards. For instance, a conflict from different goals between the users (sequence action *goal\_ambiguity* based on horizontal constraints on the problem

representation *adjacent\_left*) is indicated in the shared workspace by selecting the cards involved in the analysis (as if selecting cards for moving or deleting them altogether; the selected cards are marked by a thicker border) and by creating a text card that describes the analysis (see *Figure 6*). Another way the system can present the analysis results is by creating an additional workspace with a time stamped protocol of relevant activities.

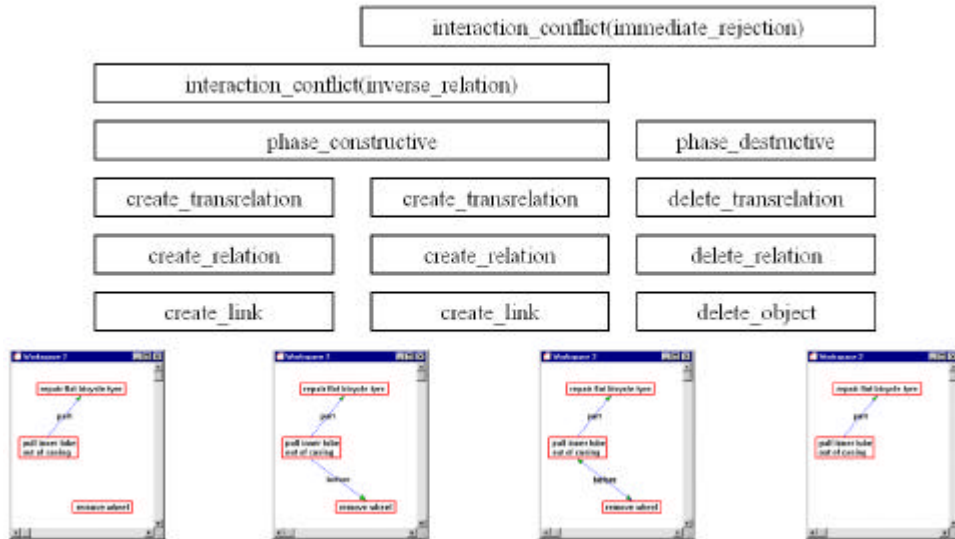


Figure 5. Example interaction in a shared workspace in the planning domain

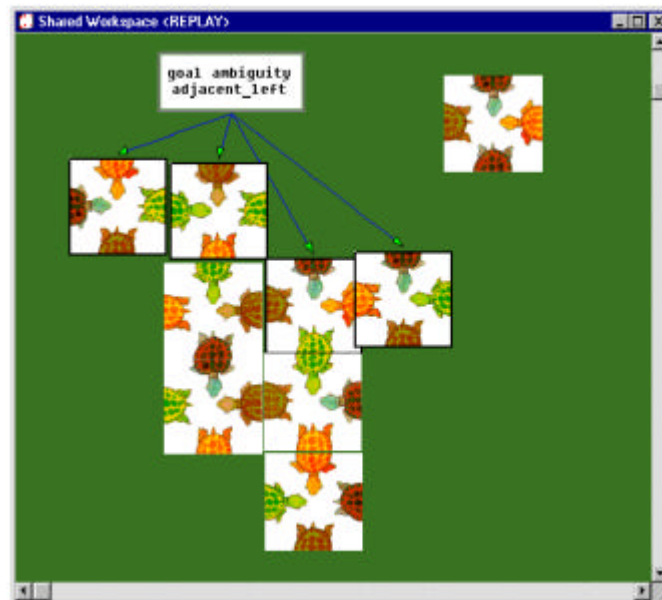
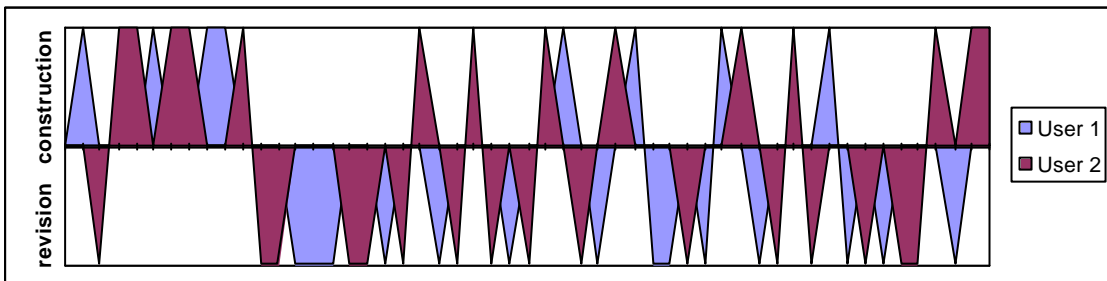


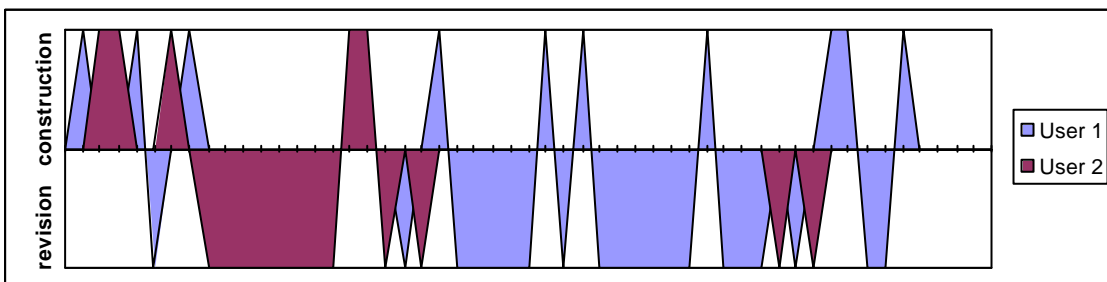
Figure 6. Feedback by the system in the shared workspace.

## 6. SYSTEM EVALUATION

One output of the system is the presentation of constructive and destructive action sequences concerning the intermediate state of the problem solution. Constructive actions aggregate elements in the workspace, whereas destructive actions are part of some revision process, e.g. in case of some impasse in the problem solving or in case of conflicting strategies on how to proceed. *Figure 7* and *Figure 8* below show sequences of constructive and destructive actions from two collaborative sessions with different users. In both sessions the users start by a number of actions that construct a partial problem solution. In both sessions, after a few actions, some local impasse is encountered, which give rise to a number of actions in which the intermediate product of the problem solving is revised. Then, the sessions continue by an alternation of constructive and destructive actions. However, in the two sessions, the users are involved in quite different ways in the problem solving. The first session (*Figure 7*) is characterized by a frequent turn taking between the users, whereas the second session (*Figure 8*) shows only a few alterations between user initiatives and the dominance of single users on the problem solving.



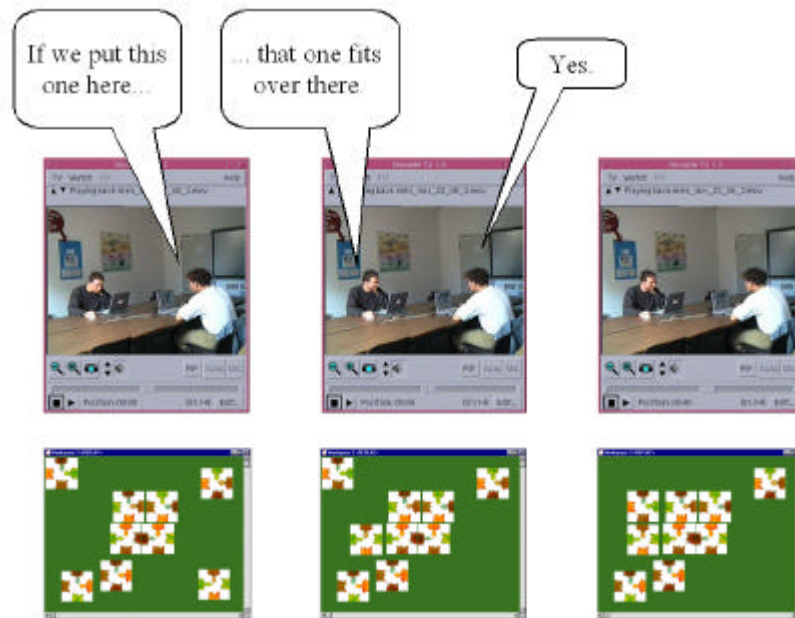
*Figure 7.* Example of a high-frequency turn taking pattern.



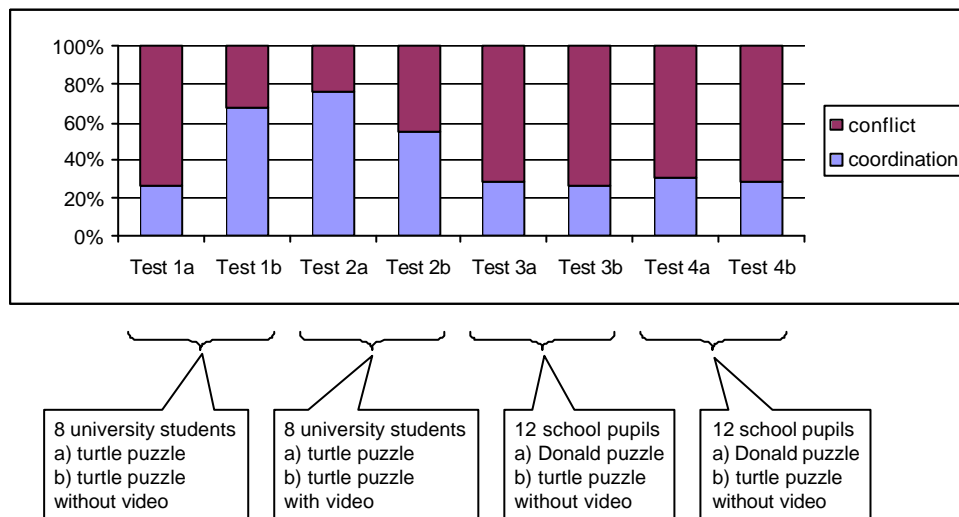
*Figure 8.* Example of a low-frequency turn taking pattern.

Video recordings that have been done in a number of collaborative sessions showed different modes of usages of the shared-workspace system in connection to verbal interaction. Sometimes the participants used exclusively the shared workspace for all communication purposes and did not talk to each other in addition. Sometimes the users were deeply engaged a verbal discussion and did not use the system for a while. However, mostly the users talked and collaborated in the shared workspace at the same time. The transcription of a portion of the video recordings showed that there are coordinative and conflicting action sequences indicated by the activity analysis that

correspond to coordinative or conflictive verbal interaction, respectively. An example for a coordination is illustrated in *Figure 9*, which shows the actions in the shared workspace, the corresponding segment in the video, and a transcription of the dialogue.



*Figure 9.* Verbal interaction that reflects the coordinative action sequence that has been recognized in the shared workspace.



*Figure 10.* Results of the tests concerning the conflicts and coordinations in different user groups.

In a series of experiments, the interaction of university students and of primary school pupils in a computer-integrated classroom (cf. Muehlenbrock, Tewissen, & Hoppe, 1998) has been investigated (see *Figure 10*). Dyads of users worked together to solve the turtle puzzle. The

university students had to solve the puzzle twice with a different initial distribution of cards in the private workspaces, whereas the school pupils firstly solved an easier puzzle (Donald puzzle). Altogether there were 16 university students and 24 school pupils, with half of the dyads being recorded by an additional video camera. In general it was noted that dyads that worked well with each other progressively reduced redundancy between verbal and action-based interaction in their second round. A comparison of the ratio between conflicting and coordinated problem solving activities showed that the coordination among pairs of university students increases in consecutive tasks, but remained on rather the same level with the primary school students. As a side effect, the video recording had an apparent influence on the proportion of coordinative action sequences with the university students, which were more coordinated in its presence.

## 7. SUMMARY AND OTHER WORK

In this chapter, a system for automatically analyzing collaborative interaction in shared workspaces has been presented. The system has been implemented as components for intelligent support that are plugged into the shared-workspace application that provides visual languages to feature collaboration concerning given problems. The analysis of the collaboration is based on activity recognition, which is defined as a plan recognition approach, relating basic workspace actions to higher-level action sequences. In addition to identifying relevant action sequences, the activity library also provides operators that indicate conflicts and coordinations. However, the approach is seen only as a starting point in developing intelligent computer-based support for collaborative learning, and much more work needs to be done in order to understand and automatically support collaboration in shared workspaces extensively and effectively.

Up to now, collaboration in shared workspaces has been investigated by focusing on the verbal interaction between the users (e.g. Roschelle & Teasley, 1995, Derry & DuRussel, 1999, Veerman, 2000). With the tools developed here, it is possible to combine this type of investigation with an automatic analysis of what is simultaneously going on in the shared workspace. Recently, a number of approaches have combined chat tools with shared workspaces for analyzing aspects such as discussion intensity, participation balance, and time on task (Angeles Constantino-Gonzales & Suthers, 2000), belief revision (Tedesco & Self, 2000), and knowledge-sharing episodes (Soller, this book). Furthermore, in a recent interdisciplinary project, the influence of the system feedback from the activity recognition on the group interaction has been investigated. The analysis system has been plugged into a more recent, Java-based implementation of the shared-workspace system, and the system feedback is presented to the user groups in form of diagrams (see *Figure 11*). Initial results indicate some effects of tracking parameters of group interaction and feeding them back to the group members (Zumbach, Muehlenbrock, Jansen, Reimann, & Hoppe, 2002).

## REFERENCES

- Akhras, F. N., & Self, J. A. (1996). A process-sensitive learning environment architecture. In C. Frasson, G. Gauthier, & A. Lesgold, editors, *Proceedings of the Third International Conference on Intelligent Tutoring Systems, ITS-96*, pages 430-438, Berlin: Springer.
- Angeles Constantino-Gonzales, M. de los, & Suthers, D. D. (2000). A coached computer-mediated collaboration learning environment for entity-relationship modeling. In G. Gauthier, C. Frasson, and K. VanLehn, editors, *Proceedings of the 5th International Conference on Intelligent Tutoring Systems, ITS-2000*, pages 324-333. Berlin: Springer.

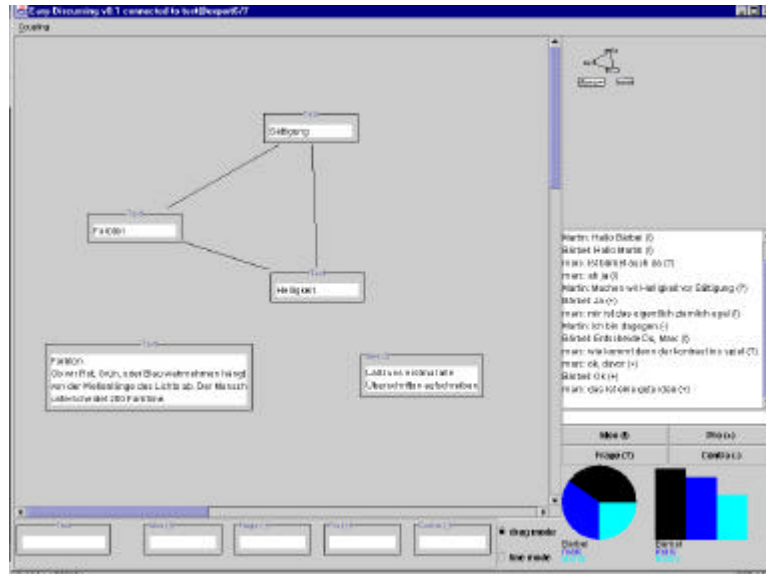


Figure 11. User interface for psychological study on co-construction with and without user feedback (bottom right)

- Baker, M. J. & Lund, K. (1996). Flexibly structuring the interaction in a CSCL environment. In P. Brna, A. Paiva & J. Self (editors), *Proceedings of the European Conference on Artificial Intelligence in Education* (pages 401-407). Lisbon, Portugal, September.
- Barros, B., & Verdejo, M. F. (1999). An approach to analyse collaboration when shared structured workspaces are used for carrying out group learning processes. In S. P. Lajoie & M. Vivet, editors, *Artificial Intelligence in Education: Open Learning Environments*, pages 449-456. Amsterdam: IOS Press.
- Cicognani, A. (2000). Concept mapping as a collaborative tool for enhancing online learning. *Educational Technology & Society*, 3(3), 150-158.
- Derry, S. J., & DuRussel, L. A. (1999). Assessing knowledge construction in on-line learning communities. In S. P. Lajoie & M. Vivet, editors, *Artificial Intelligence in Education: Open Learning Environments*, pages 431-438. Amsterdam: IOS Press.
- Dillenbourg, P. (1999). What do you mean by "collaborative learning"? In P. Dillenbourg, editor, *Collaborative Learning: Cognitive and Computational Approaches*, pages 1-19. Amsterdam: Pergamon.
- Dillenbourg, P., & Baker, M. (1996). Negotiation spaces in human-computer collaborative learning. In *Proceedings of the International Conference on the Design of Cooperative Systems*. Juan-les-Pines, France, June 1996.
- Dillenbourg, P., Baker, M., Blaye, A. & O'Malley, C. (1995). The evolution of research on collaborative learning. In P. Reimann & H. Spada (editors), *Learning in humans and machines: Towards an interdisciplinary learning science*, pages 189-211. Amsterdam: Elsevier.
- Doise, W. & Mugny, G. (1984). *The social development of intellect*. Oxford: Pergamon.
- Harrer, A. (2000). Interaction analysis at coordination level with semi-structured interfaces. In M. Muehlenbrock, editor, *Proceedings of the workshop Analysis and Modelling of Collaborative Learning Interactions at 14th European Conference of Artificial Intelligence, ECAI-2000*, pages 17-22. Berlin, August.
- Hoppe, H. U. (1993). Intelligent user support based on task models. In M. Schneider-Hufschmidt, T. Kuehme, & U. Malinkowski (editors), *Adaptive user interfaces* (pages 167-181). Amsterdam: Elsevier Science.
- Hoppe, H. U., Gassner, K., Muehlenbrock, M., & Tewissen, F. (2000). Distributed visual language environments for cooperative learning: Applications and intelligent support. *Group Decision and Negotiation*, 9(3):205-220.
- Hoppe, H. U. & Ploetzner, R. (1999). Can analytic models support learning in groups? In P. Dillenbourg, editor, *Collaborative Learning: Cognitive and Computational Approaches*, pages 147-168. Amsterdam: Pergamon.
- Jermann, P., Soller, S., & Muehlenbrock, M. (2001). From mirroring to guiding: A review of state of the art technology for supporting collaborative learning. In *Proceedings of the European conference on Computer Supported Collaborative Learning, EuroCSCL-2001*. Maastricht, The Netherlands, March.
- Kautz, H. (1990). A formal theory of plan recognition and its implementation. In J. A. Allen, H. A. Kautz, R. N. Pelavin, & J. D. Tenenber (editors), *Reasoning about plans*. (pages 69-125). San Mateo, CA: Morgan Kaufman.

- Mizoguchi, R., Ikeda, M., & Sinita, K. (1997). Roles of shared ontology in AI-ED research. In B. du Boulay & R. Mizoguchi (editors), *Artificial intelligence in education: Knowledge and media in learning systems* (pages 537-544). Amsterdam, The Netherlands: IOS Press.
- Lakin, F. (1990). Visual languages for cooperation: A performing medium approach to systems for cooperative work. In Galegher, J. & Kraut, R. & Egido, C. (eds.), *Intellectual Teamwork* (pp. 453-488). Hillsdale, NJ: Lawrence Erlbaum Associates.
- Muehlenbrock, M. (2001). *Action-based collaboration analysis for group learning*. Amsterdam, The Netherlands: IOS Press, Dissertations in Artificial Intelligence.
- Muehlenbrock, M. & Hoppe, H. U. (1999). Computer-supported interaction analysis of group problem solving. In C. Hoadley and J. Roschelle, editors, *Proceedings of the conference on Computer-supported Collaborative Learning CSCL-99*, pages 398-405. Mahwah, NJ: Erlbaum.
- Muehlenbrock, M., Tewissen, F., & Hoppe, H. U. (1998). A framework system for intelligent support in open distributed learning environments. *International Journal of Artificial Intelligence in Education*, 9:256-274.
- Novak, J. D. (1990). Concept mapping: A useful tool for science education. *Journal of Research in Science Teaching*, 27, 937-949.
- Ritter, S. & Koedinger, K. R. (1996). An architecture for plug-in tutor agents. *International Journal of Artificial Intelligence in Education*, 7(3/4), 315-347.
- Robertson, J., Good, J., & Pain, H. (1998). BetterBlether: The design and evaluation of a discussion tool for education. *International Journal of Artificial Intelligence in Education*, 9:219-236.
- Roschelle, J., & Teasley, S. D. (1995). The construction of shared knowledge in collaborative problem solving. In C. O'Malley, editor, *Computer Supported Collaborative Learning*, pages 67-97. Berlin: Springer.
- Soller, A., Linton, F., Goodman, B., & Lesgold, A. (1999). Toward intelligent analysis and support of collaborative learning interaction. In S. P. Lajoie & M. Vivet, editors, *Artificial Intelligence in Education: Open Learning Environments*, pages 75-82. Amsterdam: IOS Press.
- Stefik, M., Foster, G., Bobrow, D. G., Kahn, K. Lanning, S., & Suchman, L. (1987). Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *Communications of the ACM*, 30 (1), 32-47.
- Suthers, D., Weiner, A., Connelly, J., & Paolucci, M. (1995). Belvedere: Engaging students in critical discussion of science and public policy issues. In Greer, J. (ed.), *Proceedings of the 7<sup>th</sup> World Conference on Artificial Intelligence in Education* (pages 266-273). Washington DC, August.
- Tedesco, P. & Self, J. A. (2000). Using meta-cognitive conflicts to support group problem solving. In G. Gauthier, C. Frasson, and K. VanLehn, editors, *Proceedings of the 5th International Conference on Intelligent Tutoring Systems, ITS-2000*, pages 232-241. Berlin: Springer.
- Veerman, A. L. (2000). *Computer-supported collaborative learning through argumentation*. Enschede: Print Partners Ipskamp.
- Zumbach, J., Muehlenbrock, M., Jansen, M., Reimann, P., & Hoppe, U. (2002). Multi-dimensional tracking in virtual learning teams: An exploratory study. In G. Stahl (editor), *Proceedings of the Conference on Computer Supported Collaborative Learning CSCL-2002*, pages 650-651.