Intelligently Supported Collaborative Learning
Environments based on Visual Languages:
A Generic Approach

Katrin Gaßner, Frank Tewissen, Martin Mühlenbrook,
Andreas Loesch, H. Ulrich Hoppe
Dept. of Mathematics / Computer Science (FB 11), University of Duisburg
47048 Duisburg, Germany
gassner@informatik.uni-duisburg.de

ABSTRACT
An increasing number of collaborative learning environments is based on shared workspace systems using two-dimensional graph-structured visual representations such as argumentation networks and concept maps. We propose an integrated framework that allows for flexibly specifying a wide range of visual languages and plugging in components for operational semantics, adequate feedback and intelligent support. The generic application CardBoard provides card-based user interaction and collaboration by means of different languages in shared workspaces. To enhance modifiability, the interfaces between the distributed heterogeneous system components, particularly between intelligent components and user interfaces, have been standardized.

KEYWORDS
collaborative learning, distributed learning environment, intelligent support, visual language

1. INTRODUCTION
An increasing number of collaborative learning environments is based on shared workspace systems using two-dimensional visual representations. Particularly, there is an interest in argumentation networks and concept-mapping tools (Lakin, 1990; Novak, 1990). The common characteristic of these types of systems is that they support both spatial metaphors and direct manipulation with objects that may be themselves of symbolic nature. The formal structure underlying these representations is usually a directed graph with labeled links. Whereas concept maps in a narrower sense are only based on taxonomic relations, there are similar approaches using rhetoric or argumentative relations or more
domain-specific relations. Our own experience is based on a framework system for constructing shared workspace environments as networks of cards for different purposes, e.g. a general Discussion Board or a system for representing and constructing solution plans for physics exercises (CardMan; Tewissen, 1996).

Adaptivity or intelligent support in group-oriented learning environments is still an open issue. It can be based on syntactic features of the underlying representation, as e.g. the detection of deficient argumentation patterns in Belvedere’s argumentation networks (Suthers, Weiner, Connelly, & Paolucci, 1995). Similarly, syntactic constraints were used to check the consistency of solutions in CardMan. Intelligent support can also make use of previously assessed individual student models as in multiple student modeling (Hoppe, 1995). In the future, we may even have systems that derive student or user models from ongoing group activities (Paiva, 1997). From a modern systems engineering point of view, any case of intelligent or semantic support should be based on a clear interface between the external environment and the internal modeling systems.

Drawing on previous experience with intelligently supported collaborative learning environments (Plötzner, Hoppe, Fehse, Nolte & Tewissen, 1996; Mühlenbrock, Tewissen & Hoppe, 1997), we have developed an integrated framework for (1) flexibly defining a wide range of visual language environments as variants of a generic system, (2) easily plugging in intelligent components for semantic support and learner modeling, and (3) standardizing the interface between the group-interactive visual language environment and the „intelligent add-on“ to maximize software productivity and modifiability.

2. EXAMPLE ENVIRONMENTS AND GENERAL FUNCTIONALITY

Figure 1 shows a new version of the CardMan application that was originally developed as a domain-specific system to support the solution of exercises in the domain of mechanics (Tewissen, 1996). To introduce variables, constants, or operators, students have to select cards, connect the operands, and arrange the mathematical formulae to create a solution plan for a given exercise. We use the term card as a general denominator for any basic object in any of our workspace environments. The first domain-specific implementation has now been replaced by a version of a generic CardBoard that offers a high degree of flexibility in defining visual languages in terms of both the visual representation and the syntactic constraints for a set of cards. As shown in figure 1, the external representation of a card can be an image, a text or a dynamic component (function display).

The user interacts with cards in private or shared workspaces. Private workspaces are only visible to and accessible by the local user. With drag & drop operations, cards can be freely moved or copied from one workspace into another. Shared workspaces support collaborative problem solving by domain specific card sets. During a collaborative session, the content of a number of shared workspaces is synchronized by broadcasting and multiplying actions on objects according to the „shared user interface objects approach“ (Zhao & Hoppe, 1995). Based on this approach, a general groupware construction kit called MatchMaker supports the development of distributed applications. The MatchMaker library implements classes of user interface objects with built-in protocols for synchronization (coupling). The system independent communication protocol between the different components of the system enables us to distribute the components of the system over heterogeneous system platforms.

With a mechanism similar to shared workspace communication, the CardBoard environment also provides a record & replay function. Starting with a description of the initial state, it allows for reconstructing subsequent states based on an action history. The action history also serves as a transcript for empirical purposes.

Figure 1 shows an environment with an example of the differential calculus in a physics context. It gives an impression of what visual languages may look like and how different types of workspaces are used in the CardBoard environment. The current state of the problem solution (in the lower left), a given exercise, and the feedback of the intelligent subsystem (‘Query Result’) are located in private workspaces. The intelligent subsystem gives feedback to a single student working in a private workspace or to a group of students working together in a shared workspace. The visual representation of the feedback is given as a set of cards generated by the intelligent subsystem. These new cards can be re-used for solving the exercise.
A shared workspace is used to contact a human tutor, e.g. to discuss the problem solution and student errors (tutorial hotline). Within this workspace the union of two languages is used: the mathematical operations and variables and a small set of cards for structuring the discussion. These cards may also be incorporated into the student’s own solution network. The computer supported construction of concept maps has been investigated by an empirical psychological study at the institute of cognitive science at the University of Freiburg (Bodemer, 1998).

![Figure 1: An environment for derivation tasks in mechanics](image)

The test persons, in this case pupils from a local school, had to work in pairs on the task of solving simple mechanics (physics) problems by the construction of concept maps (cf. Plötzner et al. (1996) for the background of collaborative problem solving based on the representation of concept maps). Two of the scenarios that were investigated differed in the material available to the test persons to solve the problem, i.e., paper cards to be arranged on a desktop versus the CardBoard with a specific visual language. The results indicate that the computerized card environment is equally suited for solving these tasks. While the resulting concept maps in each of the scenarios were of equal quality, the time that was spent by the test persons to solve the problems particularly decreased in the computer supported scenario as an effect of practice. This leads to the presumption that collaborative work in shared workspaces could be even more effective than traditional settings.

3. DISTRIBUTED VISUAL LANGUAGE ENVIRONMENTS

The term visual language is used to denote a language specification together with its internal and multiple external representations. In contrast to numerous work on visual programming in which the graphical language is instrumental for the programming task, we conceive diagrammatic structures or graphs primarily as media of communication and representation (external memory) in cooperative work and collaborative learning. The Distributed Visual Language Environment (DVLE) is the basis for implementing various domain-specific applications. It provides the card-based interaction and the general communication and distribution facilities. For each workspace, the user can select a specific visual language from a set of predefined language specifications. This will generate a palette for the corresponding card set and a
new workspace. Each workspace has to be initialized with a single language to avoid ambiguities and inconsistencies. The following three representational levels are relevant for the DVLE:

1. the specification level,
2. the internal representation level,
3. and the external representation level.

On the specification level, the visual language is provided as a resource file which initializes a workspace of a DVLE instance. Figure 2 presents two examples of the specification and of the external representation of connector cards in a visual language for discussion support. The distinction between connector and content cards is essential for the language definition. Content cards are containers for domain-specific data, whereas connectors define the general structure of the language.

Connector cards (cf. figure 2) and content cards are distinguished by their Type. The Style item tells how to display the Content. In the mentioned example, the Content strings are interpreted as names of bitmaps, but textual content and dynamic components (see figure 1) are also used. The specification of connector cards also includes the Link definition, which is composed of five parts: the link type, its entry in the popup menu, its direction, its label, and the link color. SemanticType is used to identify the card type for semantic interpretation (cf. Section 4). ReadOnly, Shape, and ShapeColor are default values for the appearance of a card.

On the internal representation level, any instance of the DVLE is a generalized graph structure with n-ary relations (as in the formal concept of a hypergraph) and labeled slots or links associated with these relations. Notably, in the hypergraph paradigm with n-ary relations, connectors stand for edges and nodes are represented by content cards. Accordingly, links are substructures of hypergraph edges and not edges themselves. Additional attributes like time stamps and authorship are also handled on the internal implementation level and can support different representational views.

On the external representation level, the cards are generated according to their specification. Different shapes or colors can be used easily to distinguish connector cards and content cards, as shown in figure 2. This example presents the external representation for the two connector cards ‘Conflict’ and ‘Pro’, which are already linked with some content cards that symbolize different contributions to a discussion. The connector cards shown in figure 2 are part of a richer visual language for open discussions, which has been evaluated recently. They exemplify a distributed usage of a visual language within the DVLE. Every participant of the discussion uses an independent application

![Image](https://example.com/image.png)

**Figure 2: Card specifications in a resource file**

---

### Example Specifications

<table>
<thead>
<tr>
<th>Type</th>
<th>Structure</th>
<th>Content</th>
<th>Link Type</th>
<th>Link Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>connector_card</td>
<td>[Description]</td>
<td>&quot;Conflict&quot;</td>
<td>&quot;reference&quot;, &quot;Reference&quot;, 1,&quot;&quot;,0,0,0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>bitmap</td>
<td></td>
<td>true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shape= circle</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ShapeColor=0,0,0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Link=&quot;contradiction&quot;, &quot;Conflict&quot;, -1,&quot;&quot;,0,0,0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| connector_card | [Description]     | "Pro"               | "reference", "Reference", 1,"",0,0,0   |
|                |                    | bitmap             |           | true       |
|                |                    | Shape= circle     |           |            |
|                |                    | ShapeColor=0,0,0   |           |            |
|                |                    | Link="contribution", "Pro", -1,"",0,0,0         |
instance. Privately prepared contributions may be published in shared workspaces to have a public discussion.
For flexible presentation, the layout of the external representation is not inherent in the language, but only exists as set of default values on the specification level. In our current work, we aim at extending the flexibility to work with multiple languages and support the menu-driven definition of visual language specifications during run-time.

4. ADDING MODELING AGENTS FOR INTELLIGENT SUPPORT

Reasonable feedback from the system to the users can only be given if we extend the signature of a visual language by a mechanism for the interpretation of its card symbols. This general mechanism is to provide operational semantics for specific types of card nets, i.e., an inference engine assigns operations to symbols introduced by some visual language specification. This descriptive approach serves the following four 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 color, shape or content. These values are propagated on the basis of the semantic types of linked cards. On this basis, the consequences of card manipulations can be shown to the user.
- A card denotation that allows for a clear and unequivocal description of statements and arguments by the users can be achieved by standard formalisms such as propositional and first order logic.
- The interpretation of cards prepares the ground for standard student modeling techniques and intelligent support.

Operational semantics and intelligent support are added to a DVLE using a general architecture called Dalis (Mühlbrock et al., 1997). Different components for interpretation and for individual and group support can be plugged in as agents into the distributed Dalis system. This is comparable to the approach of Ritter and Koedinger (1996) for incorporating tutoring elements into pre-existing software packages. The Dalis 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 specialized agents for monitoring and modeling the human participants. It is quite simple to add arbitrary Prolog programs as new types of agents to the environment. Dalis establishes a federation architecture that facilitates the message flow between its internal agents and the DVLE components that are connected to the MatchMaker server (cf. section 2).

![Image of diagram](image_url)

Figure 3: Interfacing the card environment and the domain interpreter

The system reacts to the needs of the various learning and collaboration modes in a flexible manner. As a prominent feature, dynamically updated HTML files are used as a means for inspecting the flow of information between the system components (cf. figure 4). This is particularly useful on heterogeneous system platforms with no common file system in order to generate an overview of the
registered agents and user interfaces.
Within this support framework, we propose two kinds of agents particularly dedicated to card environments: mediator and interpreter agents. Basically, the mediator and interpreter agents act in the following ways (see figure 3):

- 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 and ownership of user actions.

- The interpreter agents provide operational semantics for the card environments. General formalisms for propositional and first order logic as well as special purpose formalisms for planning, discussions etc. can be incorporated as interpreter agents.

<table>
<thead>
<tr>
<th></th>
<th>general</th>
<th>specific</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>application</td>
<td>workspace</td>
</tr>
<tr>
<td>create</td>
<td>id, actor</td>
<td>user, language</td>
</tr>
<tr>
<td>modify</td>
<td>id, actor</td>
<td>title, mode, dimension, visible_area</td>
</tr>
<tr>
<td>delete</td>
<td>id, actor</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Message primitives for the creation, deletion and, modification of DVLE objects (obligatory arguments in bold)

A DVLE and a mediator communicate by means of some basic message primitives for the creation, deletion and modification of card objects. Each message refers to the id of a DVLE object, i.e., an application, a workspace, a card, or a link, together with the name of the actor who initiated the particular action of creation, deletion, or modification (see table 1). Further arguments relate to object-specific attributes. On the basis of messages from the DVLE, the mediator incrementally reconstructs the state of the card environment by receiving and accumulating relevant information. On the other hand, it actively constructs and modifies card nets.

The mediator monitors continuously all the modifications in the card workspaces. It then translates the current card net into the internal formalism of the interpreter based on a descriptive representation of the card symbols. In the example described in section 2, the interpreter analyzes derivatives and generates hypotheses about incorrectly solved subproblems (Hoppe, 1995). For instance, in the course of the sample problem solving that is shown in figure 1, the user has asked the system to check his derivative by placing a question card on the derivation connector. Triggered by this user action in the DVLE, the mediator converts a part of the sample card net in figure 1 into the expression \( \text{term}(\text{deriv}(\text{cos}(x), \text{sin}(x))) \), which is given as input to the interpreter agent. The interpreter analyzes the student’s solution, finds that it is incorrect, and builds a hypothesis on the student’s failure, which is represented as \( \text{term}(\text{deriv}(\text{cos}(x), \text{sin}(x))) \). The answer of the interpreter is returned to the mediator, which then generates a further workspace ‘Query Result’ and constructs a tree which represents the incorrect part of the derivation. The message flow between mediator and DVLE due to this user request is partly shown in figure 4.

In the context of group learning, individual students models are generated by the interpreter and are accumulated and integrated to derive a model of group problem solving. This group model initiates and supports help activities such as arranging groups on selected topics and suggesting an appropriate peer helper for a student with difficulties (Mühlenbrock et al., 1997).
5. DISCUSSION

As compared to earlier approaches to providing distributed graphical environments for collaboration (Lakin, 1990; Stefik, Foster, Bobrow, Kahn, Lanning, & Suchman, 1987), our work is focused on the following new aspects: (i) the flexible definition and modification of the visual languages used; (ii) the integration with different kinds of intelligent support; (iii) the development of systems engineering principles for building such environments.

The Belvedere project at LRDC Pittsburgh follows a partly similar rationale (Suthers & Jones, 1997). One difference to our approach lies in Belvedere’s restriction to scientific argumentation and inquiry. In Belvedere, the shared workspace facility relies on a central database which can also be used as an archive for storing and retrieving arguments. Our rationale is different in that we focus on process features and coordinated action as transient phenomena when interacting through shared workspaces. Accordingly, our communication model is based on synchronization through actions and full replication of the shared workspace environments as originally stated in (Zhao & Hoppe, 1995).

Although our communication model does not naturally support archiving functions, it has some other inherent strengths: E.g. in peer-to-peer tutoring or tutorial hotlines, the partners build or modify a shared workspace which exists in two synchronized instances. When the joint session ends, each partner is left alone with one of these instances, which are now independent. Each user is free to store, delete, or further develop this workspace. This avoids the overhead of maintaining several potentially divergent versions of a shared document in a centralized database. Replication is also of advantage as compared to windows sharing with a centralized master application as supported in a number of current commercial products. Here, only the master version “survives” after terminating the joint session, which implies that the results are by default lost for the other participants.

Distributed visual languages can support a wide spectrum of subject areas, ranging from very formal domains, as with the physics and logic examples, to semi-formal representations as e.g. argumentation structures in group discussions. Only for some formal domains, it is possible to provide complete semantic models that allow for checking the inherent correctness and consistency of given diagrams. For semi-formal representations, local advice can be based on domain-specific constraint rules. In our work, this approach has been adopted in the mechanics domain (cf. section 1). Another example is
Belvedere’s mechanism for detecting faulty argument patterns as described in (Suthers, et al., 1995). The Dalis architecture has also been used to implement more sophisticated types of error diagnosis and advice for learning groups (Mühlenbrock, et al., 1997), but originally not within the DVLE. A sophisticated formal model with diagnostic capabilities is currently being implemented for the logic environment.

One possible function of distributed visual languages in group communication is the one of an external memory which stores and represents intermediate or final results. However, this sort of externalization is more than a passive resource. We assume that externalization supports problem solving in a variety of ways, and that a common shared workspace is necessary for collaboration as explained in (Burton & Brna, 1996). We claim the positive effect of externalization not only for problem solving, but also for more open forms of dialogues and discussions.

A language to support open group discussions has been developed to represent discussion contributions and their relations. The relations have to be chosen from a finite and predefined set of connector cards that are used to relate textual contributions to others. (Figure 2 shows two connector cards of this language.) The discussion support is intended to be used during a discussion situation to put in and edit statements and their references by each group member who works on his/her own application instance. In this scenario, a technically enriched face-to-face situation, both the verbal discussion and the common workspace are means for collaboration where the workspace serves as an external memory. The kind and frequency of textual contributions is a result of the group’s self-organisation.

To build hypotheses about the communicative behavior of groups that use this visual language, first tests in a setting of small groups of students and a moderator have taken place at the University of Duisburg. In the tests the common workspace has been successfully used to refer to statements and reflect propositions. Nevertheless, it was obvious that the communication situation became more complex in comparison to the traditional face-to-face situation.

A previously underestimated but interesting problem consisted in the coordination of speech and writing. This goes beyond the task to protocol during listening. One intuitive method was to write down a verbal contribution immediately. However, this prevented to listen to feedback and therefore this method destroyed the discussion flow in any case. Another method used was to wait putting down the contributions. In this case moderation was needed to remind the participants to put down their contributions. This caused a loss of information as well as the difficulty to refer to other contributions, that have not been put down to the common workspace yet. As a solution we successfully applied a cyclic strategy to coordinate speech and textual contributions in that different activities take turns. In one period a traditional verbal discussion takes place. This period is followed by a phase in which each participant puts down the contributions on content cards that he/she remembers. As mentioned above, a loss of information is caused by the delay of the textual input, but it seemed that the subjectively interesting and important contributions was remembered best. In the next period the relations among the content cards are under discussion. In comparison to an individual input of connections among textual contributions in the CardBoard, this improved the common understanding of the connector card symbols and the reflection of propositions. Moreover, the discussion of the adequate use of connector cards caused an easy transition to the first phase of the cycle.

We assume that appropriate distributed visual languages support the co-construction of explanations as well as argumentation and reflection. The target of reflection can be on the level of content as on the meta level of roles that participants adopt during a discussion. Certain roles can be assigned to participant by providing them with specific languages (cards sets) to support complementary interaction in the style of dialogue games. In our current practice, we are studying the use of the distributed visual language environment in academic education.
REFERENCES


