Group-oriented Modelling Tools with Heterogeneous Semantics

Niels Pinkwart, H. Ulrich Hoppe, Lars Bollen, Eva Fuhlrott

Institute for Computer Science and Interactive Systems
Faculty of Engineering, University of Duisburg
47048 Duisburg, Germany
Phone: (+49) 203 379-1403
pinkwart@collide.info

Abstract. This paper describes an approach of how to support collaborative modelling tasks. The presented system, Cool Modes, implements the approach using "plug-in" reference frames encapsulating the semantics of the used models. Details on the extensibility of the system and the definition and interpretation of these reference frames and models in the framework are shown and the co-operation support using the underlying MatchMaker communication server is explained. Furthermore, two examples from the domains "System Dynamics modelling" and "Jewish Ceremonies" are given.

1 Introduction

Currently, information technology or computer support for collaborative learning is mainly based on computer-mediated communication such as, e.g., on conferencing techniques and sharing of resources and materials as well as on digital archives. This implies that the information exchanged, voice, text or images, is "passed through" the system, but not semantically processed. On the other hand, originally motivated by the limitations of conventional individualised computer tutors, there was another tendency to have more interactivity in learning environments using rich and powerful "computational objects to think with". This lead to the development of interactive cognitive tools or "mind tools" [1], which were essentially based on the direct manipulation of visual objects by the user-learner but also based on the computational processing of related symbolic objects and representations. Typical examples are visual languages for argumentation and discussion as well as visual tools for simulation and scientific modelling. A first suggestion of how to support collaboration with modelling tools in "discovery learning" has been made by van Joolingen [2].

We see a new challenge in providing "computational objects to think with" in a collaborative, distributed computing framework. This is typically achieved through shared workspace environments which allows a group of learners to synchronously co-construct and elaborate external representations. Systems may differ considerably with respect to the degree of semantics or structure that is explicitly captured by the computerised representation: Whereas the internal structure of whiteboard (drawing) tools

is based on strokes, colours, and geometrical shapes, concept mapping tools constitute "semantic" relations between certain objects or nodes. However, it is not clearly defined in how far the semantics of the representation is really interpreted by the machine. Internally, a concept mapping tool may just be based on an abstract graph structure, whereas more specific representations such as System dynamics models [3] or visual programming languages come with a clearly defined and rich internal operational semantics. They provide a complete semantic definition of all objects and thus allow for "running" the models. On the other hand, less specific systems like Belvedere [4] do not interpret the semantic content of the objects but the rhetorical or argumentative types and relations between objects (e.g. "hypothesis", "conclusion"). The system is aware of the developed argumentation structure and points out missing relations via a support agent. A recent example of a collaborative learning environment based on a domain-specific visual language is the COLER system [5] which supports the co-construction of entity-relationship (ER) models for database modelling.

The CardBoard environment [6] allows for creating "collaborative visual languages" by parameterising a general shared workspace environment. The particular language profile specifies the syntax of the respective language, i.e. the given set of relations, their argument slots, and the basic object types. To add semantics in terms of domain models or knowledge bases, an interface is provided that transfers actions from the visual language environment to the semantic plug-in component [7]. This architecture allows for flexibly defining semantically enriched tools, such as e.g. a co-operative editor and simulator for Petri Nets.

Our current work on the Cool Modes (*CO*llaborative *O*pen *L*earning and *MODE*lling *S*ystem) environment draws on the CardBoard experience, yet with an improved underlying communication mechanism (Java MatchMaker TNG) and the new orientation to provide multiple "language palettes" to choose from and the possibility of mixing different types of languages or representations, ranging form free-hand drawings over concept maps to semantically defined modelling languages (Petri Nets, System Dynamics), in one workspace.

In our vision of future applications (and already starting in our own practice), we see these multi-functional and multi-representational tools as digital, active extensions of chalkboard and paper & pencil as demonstrated in the NIMIS project [8]. The tools should ideally be used in networked ubiquitous and potentially mobile computing environments to support modelling, interactive presentation and group discussion in a variety of educational scenarios, including traditional lectures (presentation) as well as tutorials and collaborative work in small groups.

2 Cool Modes – an Extensible Platform

Cool Modes is a collaborative tool framework designed to support discussions and co-operative modelling processes in various domains. Like in some other environments [4,9], this is achieved through a shared workspace environment with synchronised visual representations. These representations together with their underlying semantics can be defined externally which offers the option to develop domain-dependent "plug-

in" visual languages and interpretation patterns, encapsulated in so-called "palettes". The languages can differ considerably with respect to the underlying formal semantics (e.g. System dynamics simulation vs. handwriting annotation) but yet be mixed and used synchronously in the framework which from our point of view is a suitable approach for supporting open modelling tasks with potentially unknown means.
With the aim of having an extensible and useful platform suitable for the integrated use in multiple domain contexts, Cool Modes offers some generic representation elements and co-operation support. These are shortly described in 2.1, the extensibility of the system ("How can my favourite elements be integrated?") is shown in 2.2. and 2.3. These descriptions are mainly to outline the principles of realising semantically enriched extensions (e.g. simulations) in the framework, concrete examples are given in chapter 3.

2.1 Generic System Functions

Cool Modes allows the use of multiple workspaces represented in different windows which can be arranged freely. Each workspace consists of a number of transparent layers which can contain "solid" objects like e.g. handwriting strokes, images and other media types. Four predefined layers with different functionality exist by default - one for a background image, one for hand-written annotations and two for other objects - more can be dynamically added.
While the workspaces contain the results of the user's work and interaction with the system or other users, the elements for this interaction are available in "palettes". These can be dynamically added and removed and are the basic means of extending the system (cf. 2.2 and 2.3). Yet, some standard palettes useful for any domain are predefined: Cool Modes offers a "handwriting" palette allowing the user to directly annotate anything within a workspace. The second more general palette consists of different patterns for discussion support like "question" or "comment". The elements of this palette are designed to support the users in discussing their current work and structuring their argumentation.
As mentioned, the co-operation support integrated in Cool Modes basically relies on the provision of synchronously shareable representations. Technically, this is realised using the MatchMaker server [10,11] offering a replicated architecture, partial synchronisation features and dynamic synchronisation. According to the system structure with its workspaces and layers, the synchronisation of objects is flexibly possible, e.g. in the following different ways:

- By workspace, allowing the users to co-operate by the means of completely synchronised workspaces (and having private ones additionally).
- By layer, providing e.g. the option to have private hand-written annotations on synchronised workspaces as shown in figure 1.

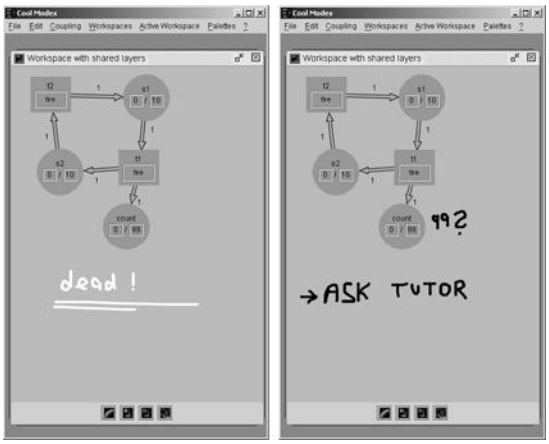

Fig. 1. Layer-wise synchronisation

- By Element. In this "low" level of synchronisation, the synchronised objects “to be discussed co-operatively” can be explicitly defined; any other objects will be private. It is imaginable to set up e.g. a modelling scenario in which only the model elements are shared, but the rest of the user's workspace content (like annotations, images, supporting notes and experimental results) is kept private.

MatchMaker supports this partial synchronisation by providing synchronisation trees. These trees normally reflect the structure of the coupled application, in the case of Cool Modes the subdivision workspace-layer-element. Applications can join different nodes in the tree and thus only receive and deliver synchronisation events concerning the selected shared parts of the representation.

2.2 Definition of Domain-dependent Elements

The structures in Cool Modes containing the domain dependent semantics and thus offering the possibility for simulations, modelling and, more generally, integrating (potentially arbitrary) algorithms, are called reference frames. As outlined in [12] in more detail, these reference frames serve as structures which

- list the representation elements together with their semantic relations
- Cool Modes refers to in order to interpret the constructed results of user's work
- describe visual interfaces allowing the user to make use of the elements

The most important elements described in these reference frames are the visual elements it offers to the user, together with their semantics and, potentially, functions. The single available elements are called “nodes” and “edges”, their relations are described in the reference frame itself whose visual interface is a “palette”.

Palettes
A palette provides access to the reference frame for the users. Using the menu bar, it can at runtime be added and removed; the synchronous use of multiple palettes is possible. The two basic functions of palettes are:

- The provision of the nodes and edges defined in the reference frame. Nodes in a palette can be dragged into workspaces and connected via the edges
- The option of receiving and reacting to “global” events generated by the workspaces, nodes or edges (cf. 2.3).

Nodes
A node is a standard Java object extending a predefined class `AbstractNode`. It is structurally divided in three parts:

- The node itself acting as controller, pointing to its reference frame and capable of receiving, sending and processing events - in both senses of "standard" Java events and internal, more semantically enriched, events within the Cool Modes system or even the reference frame (cf. 2.3).
- The node model, with information according to the definition in the reference frame. Due to the fact that the model is used by MatchMaker for coupling, it must implement `java.io.Serializable`.
- The view. Principally, any `javax.swing.JComponent` can be used as a view, yet Cool Modes offers some patterns like arbitrary-shaped views with auto-resizing text input fields.

Edges
Edges serve as connections between the nodes in the constructed graph structures and, similar to the nodes, consist of model, view and controller. Additionally, it is possible to define rule sets describing allowed and forbidden connections.

2.3 Interpretation of Domain-dependent Semantics

While the principal way of *defining* custom semantically enriched components has been described in the previous chapter, the following part tries to give a rough idea about the general possibilities of *interpreting* the semantic relations within Cool Modes. Basically, the approach relies on four foundations:

1. The reference frame defines the semantic relations (expressed in data model types and associated algorithms) and is responsible for the objects of one domain "understanding" each other.
2. General events generated through user actions contain node and edge model as parameter. These encapsulate the current state of the object, in terms of the associated domain.
3. There are "local" and "global" events and, accordingly, the option of having local and global listeners.
4. Specific, additional events can be freely defined in the reference frame.

Global Events and Control
A global event is fired upon a change in a workspace. Typically, the palettes and therefore, indirectly, the reference frames, listen and react to it:

```
void nodeAdded(NodeEvent e);
void nodeRemoved(NodeEvent e);
void nodeMoved(NodeEvent e);
void edgeAdded(EdgeEvent e);
void edgeRemoved(EdgeEvent e);
void edgeMoved(EdgeEvent e);
```

As the listening component has access to the workspace, it can check the workspace content upon reception of an event and can this way, e.g., check the following aspects:

Element presence / absence
As shown in chapter 3.2, the pure information about which nodes or edges are currently in a workspace might already be of interest. This type of "lightweight" model-checking is useful for some modelling tasks with a more or less predefined solution: the system can keep track if the required elements have already been added by the users and, if desired, give hints.

Spatial relations in the workspace
Making use of one more piece of available information allows the system to react upon the absolute or relative positioning of the nodes and edges in the workspace. An example for this is shown in chapter 3.2, where only the correct relative arrangement of four required images fulfils a task completely.

Abstract graph structure
In addition to the above point related to the visual positioning of workspace content, it is also possible to run arbitrary graph algorithms. This allows, e.g., connectivity checks in a Petri Net model or automatic testing of the correlation between topics of a discussion.

Node and Edge models
While the three ways of reaction to events listed above have been rather independent of a concrete domain, it is also possible for the global listener to interpret or change the semantics embedded in the node and edge models. An application for this is the global simulation of models through the use of control elements contained in the palette as shown in the System Dynamics example (chapter 3.1), although the "trigger" for this simulation is in this case not the change of the workspace content but a specific user action (pressing the "step"-button). Potentially, any domain-dependent algorithm that requires the whole graph structure as input, can be realised this way.

Local Events and Control
In addition to the global control and checking options and events pointed out above, some local events are available. Any node can be a listener for these and thus be kept up-to-date about changes within its direct neighbourhood in the graph. The events cover information about which other nodes have been attached by what type of edge, which nodes have been disconnected from the listener node and about model changes in the neighbour nodes and edges. Two typical applications that make use of these events are *local graph algorithms and model changes* (like e.g. the change of the activation status of transitions in a Petri net simulation whenever the model of a place or an edge has changed, or, more generally, searching or distance calculations in the graph) as well as the provision of *context-based feedback.* Even in an open task with no strict rules defined, it is possible to check the modified local structure against a known "ideal" solution or heuristic rule set and give hints.

3 Examples

3.1 System Dynamics: The DynaBoard

In this example we will show the basic functions of the „DynaBoard“, one of the reference frames included in Cool Modes at the moment. The creation of the DynaBoard has been part of a master thesis [13] and provides basic support for modelling and simulating System Dynamics models [3]. Figure 2 shows a group of learners trying to solve a problem about the growth of a golf course area with limited resources. Besides being an example for the use of more than one palette at a time and in one shared workspace, this figure shows the basic elements of the DynaBoard palette. You can see a stock (the „golf_area“), a rate („development“) and a constant („fraction“). By using different types of edges, you can distinguish between the flow of information (thin, black, cracked arrows) and the actual flow of values (thick, grey, straight arrows). The right

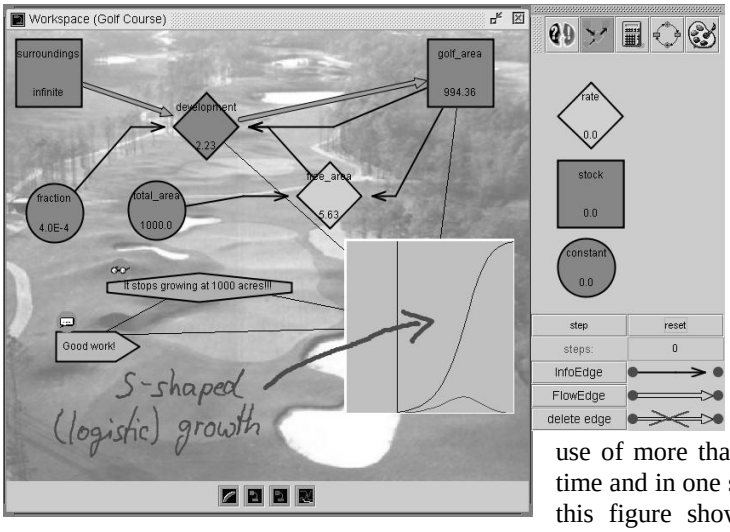

Fig. 2. Cool Modes: System Dynamics modelling

part of figure 4 shows the user interface of the DynaBoard palette with buttons to control the simulation of the model.

3.2 Jewish Ceremonies

During an interdisciplinary project, students at the University of Duisburg developed the "Jewish Ceremonies" frame within Cool Modes. It addresses students at the age of 11-16 and, thought as addition to the curriculum of religious instruction, it gives the opportunity to test and to intensify knowledge about Jewish life and ceremonies. This task is inherently open like most things currently being supported by Cool Modes. Yet, there are some defined instructions and tasks together with their solution implicitly defined in the corresponding reference frame.

Available palettes: “Ceremonial Objects” and “Hebrew”

The "Hebrew" palette can be used to write Hebrew words (from right to left) in special nodes. Consequently, the palette (see figure 3) contains this node and the available letters. Writing is done by clicking the buttons. To practise already known words and their pronunciation, the program gives feedback by presenting a loudspeaker icon if the word exists (in a local database) and is written correctly. Clicking the loudspeaker icon starts the matching sound file. Writing in this framework includes functions such as deleting a letter, inserting spacebars, changing letters automatically to “sofits” (if they appear at the end of a word) or the decision whether the text box shall be active and thus editable or not.

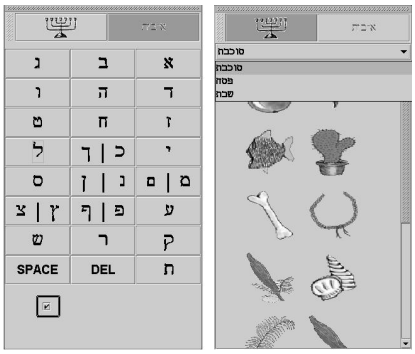

Fig. 3. “Jewish Ceremonies” palettes

The selection of one of the Jewish (religious) ceremonies takes place on top of the "Ceremonial Objects" palette (see figure 3) and returns a specific assortment of items on the palette. The names of the ceremonies are written in Hebrew. This palette shows pictures of items belonging to jewish rites, with Hebrew tool-tips. Not all of the items of one category are in correct context with the celebration. The user has to choose the right ones and to arrange them.

As usual in Cool Modes, these two palettes can be combined, showing the arranged items of both palettes at the same time in a workspace. An easy possible assignment could be to show an item and to ask the user to write its name.

Usage Scenario and Result checking

Several assignment scenarios exist for the "Ceremonial Objects" palette. One assignment is to build a lulaw, a bunch of branches for the celebration of Sukkot. Four branches of plants must be arranged correctly in a workspace to fulfil the task. As soon as the four items are in one workspace, a message box opens to report that they

are complete and now have to be arranged (see figure 4). When the lulaw is built correctly, another message box comments that the task was completed successfully. Technically, this check is done using the events described in chapter 2.3. The listener here just has a list which contains the models of all needed items and compares it to all currently present items in the workspace. If this comparison succeeds, the internal state "correct items" is entered and presented to the user via a message. If the internal state is already "correct items" and an item is moved, the second checking routine tests if the relative positions of the items are correct and, if so, delivers the final output that the task is fully completed.

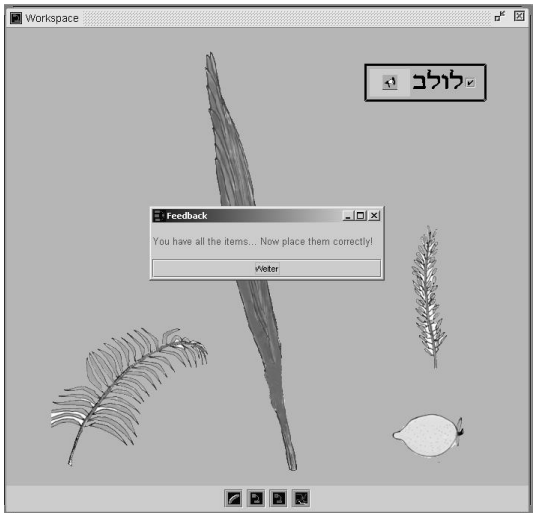

Fig.4. Model-checking – constructing a lulaw

4 Outlook

We see the Cool Modes environment with its example applications as a first step towards providing collaborative mind tools in a variety of realistic educational and training settings. The flexible coupling model provided by MatchMaker allows for flexibly supporting different co-operation modes with limited time and partial synchronisation. Ongoing work is focused on the following extensions:

- a standardised parameterisable protocolling mechanisms to support "undo" and several replay options on a general level, as well as the provision of action transcripts for analysis and reflection
- the development of "lightweight Cool Modes clients" to be used as mobile interactive frontends on wireless PDAs or tablets
- the enhancement of the included XML support towards using XML not only for data storage but also for synchronisation, e.g. by providing a SOAP interface.

References

1. Jonassen, D. H., Tessmer, M. & Hannum, W. H. (1999). Task Analysis Methods for Instructional Design. New York: Erlbaum.

2. Joolingen, W. R., van (2000). Designing for Collaborative Learning. In Gauthier, G., Frasson, C. & VanLehn, K. (Eds.): Intelligent Tutoring Systems (Proceedings of ITS 2000, Montreal, Canada, June 2000) (pp. 202-211). Berlin: Springer.
3. Forrester, J. W. (1968). Principles of Systems. Waltham, MA: Pegasus Communications.
4. Suthers, D. D., Weiner, A., Connelly, J., & Paolucci, M. (1995). Belevedere: Engaging students in critical discussion of science and public policy issues. In Proceedings of the World Conference on Artificial Intelligence in Education. Washington, DC: American Association for the Advancement of Computation in Education.
5. Constantino-González, M. & Suthers, D. D. (2000). A Coached Collaborative Learning Environment for Entity-Realationship Modeling. In G. Gauthier, C. Frasson & K. VanLehn (Eds.), Intelligent Tutoring Systems, 5th International Conference, Montréal, Canada. Berlin: Springer.
6. 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 (Kluwer), vol. 9, 205-220.
7. Mühlenbrock, M., Tewissen F. & Hoppe H. U. (1997). A Framework System for Intelligent Support in Open Distributed Learning Environments. In B. du Boulay & R. Mizoguchi (Eds.), Artificial Intelligence in Education: Knowledge and media in learning systems, Proceedings of the 8th International Conference on Artificial Intelligence in Education , AIED-97. Amsterdam: IOS Press.
8. Tewissen, F., Lingnau, A., H. Hoppe, H.U. (2000). "Today's Talking Typewriter" - Supporting early literacy in a classroom environment. In Gauthier, G.; Frasson, C.; VanLehn, K. (Eds.): Intelligent Tutoring Systems (Proceedings of ITS 2000, Montreal, Canada, June 2000) (pp. 252-261). Berlin: Springer.
9. Wang, W., Haake, J. & Rubart, J. (2001). The Cooperative Hypermedia Approach to Collaborative Engineering an Operation of Virtual Enterprises. . In Proceedings of CRIWG 2001 (Seventh International Workshop on Groupware, Darmstadt, Oktober 2001) (pp. 58-67). Los Alamitos: IEEE Press.
10. Tewissen, F., Baloian, N., Hoppe, H.U. & Reimberg, E. (2000). "MatchMaker" – Synchronising objects in replicated software architectures. In Proceedings of CRIWG 2000 (Sixth International Workshop on Groupware, Madeira, P, Oktober 2000) (pp. 60-67). Los Alamitos: IEEE Press.
11. Jansen, M., Pinkwart, N. & Tewissen, F. (2001). MatchMaker - Flexible Synchronisation von Java-Anwendungen. In R. Klinkenberg, S. Rüping, A. Fick, N. Henze, C. Herzog, R. Molitor & O. Schröder (eds.) LLWA 01 - Tagungsband der GI-Workshopwoche "Lernen-Lehren-Wissen-Adaptivität". Forschungsbericht 763, Oktober 2001. Universität Dortmund, Germany
12. Pinkwart, N., Hoppe, H.U., Gaßner, K. (2001). Integration of domain-specific elements into visual language based collaborative environments. In Proceedings of CRIWG 2001 (Seventh International Workshop on Groupware, Darmstadt, Oktober 2001) (pp. 142-47). Los Alamitos: IEEE Press.
13. Bollen, L. (2001). Integration einer visuellen Modellierungsprache in eine kooperative Diskussionsumgebung für den naturwissenschaftlichen Unterricht. Unpublished Master Thesis, University of Duisburg, Institute for Computer Science and Interactive Systems.