A Plug-In Architecture for Graph Based Collaborative Modeling Systems

Niels PINKWART
University Duisburg-Essen, Germany

Abstract. This paper presents ongoing work on a graphical modeling and discussion support system. One of the properties of the presented approach is that it tries to implement a "collaborative mind tool" approach, bridging the gap between a communication means and a system with AI functionality. The paper also points out the XML based plug-in mechanism of the system and illustrates it by several categorized examples.

1. Introduction : Graph Based Collaborative Modeling Systems

In recent years, cooperative systems with a special focus on enabling sharing and commenting of resources and material have been a prominent subject in the research area of computer technology in education. The representation mode of these shared resources plays an important role [1] for the learning process. One frequently used mode is a graph based notation. Here, the shared visual representation consists of objects (nodes) and relations (edges) between them. With a focus on argumentation support, this representation has been used in a number of successful environments such as Belvedere [1] or gIBIS [2]. Belvedere also aimed to teach students scientific argumentation – the system contained constraints that had to hold for the graph structure developed by the users. This approach of enriching cooperative graph based environments with flexible rules and interpretation patterns goes in line with pedagogic approaches like "discovery learning in science" [3] or "Model Facilitated Learning" [4] that show the possible benefit of integrating computer based modeling methods and intelligent techniques with cooperative environments. As a number of modeling languages (like, e.g., Petri Nets) make use of a graph based notation, this promising integration step seems possible smoothly. Yet, generic approaches for the integration of modeling languages and interpretation systems with discussion or group knowledge elicitation support are rare. Current environments are, like COLER [5], often written for one domain (in this case, database modeling), or, like Microsoft Visio, lack simulation and/or cooperative features. Some existing systems that combine argumentation support with simulation tools (like the Science Learning Space described in [6]) are based on tool *inter*operability - this paper outlines a system that offers "pluggable" domain semantics and AI functionality based on visual languages and focuses more on an intelligent *intra*-tool interpretation mechanism. It puts collaborative aspects more in the foreground than the comparable math-oriented ESCOT system [7] or the microworlds approach of e-Slate [8].

2. The Cool Modes Framework

Cool Modes (COllaborative Open Learning and MODEling System) is a collaborative tool framework designed to support discussions and cooperative modeling processes. It provides the potential to mix different conceptual representations and therefore is not only a

communication means for educational cooperation but also supports the integration of AI based functionalities such as feedback and interpretation of workspace states.

Like in some other environments [1,2,5], the cooperation support is achieved by means of a shared workspace environment with synchronized objects (realized by a MatchMaker TNG server [9]). The synchronized objects together with their visual representations and underlying semantics can be defined externally in "reference frames" (cf. 3.) which offers the option to develop domain-dependent plug-ins: visual languages, interpretation mechanisms and intelligent user support. These system extensions can differ considerably with respect to the underlying formal semantics (e.g. Petri Net simulation vs. handwriting annotation) but yet be used synchronously in an integrated way, mixing different conceptual representations, with the aim of supporting open modeling tasks without strictly predefined means but a variety of available options.

Figure 1 illustrates this approach of flexibly mixing elements of different conceptual representations while retaining specific domain semantics as far as possible. The screenshot shows a prototypical situation in a computer science course: a group of students collaborates (by shared workspaces) in order to fulfill a specific programming task.

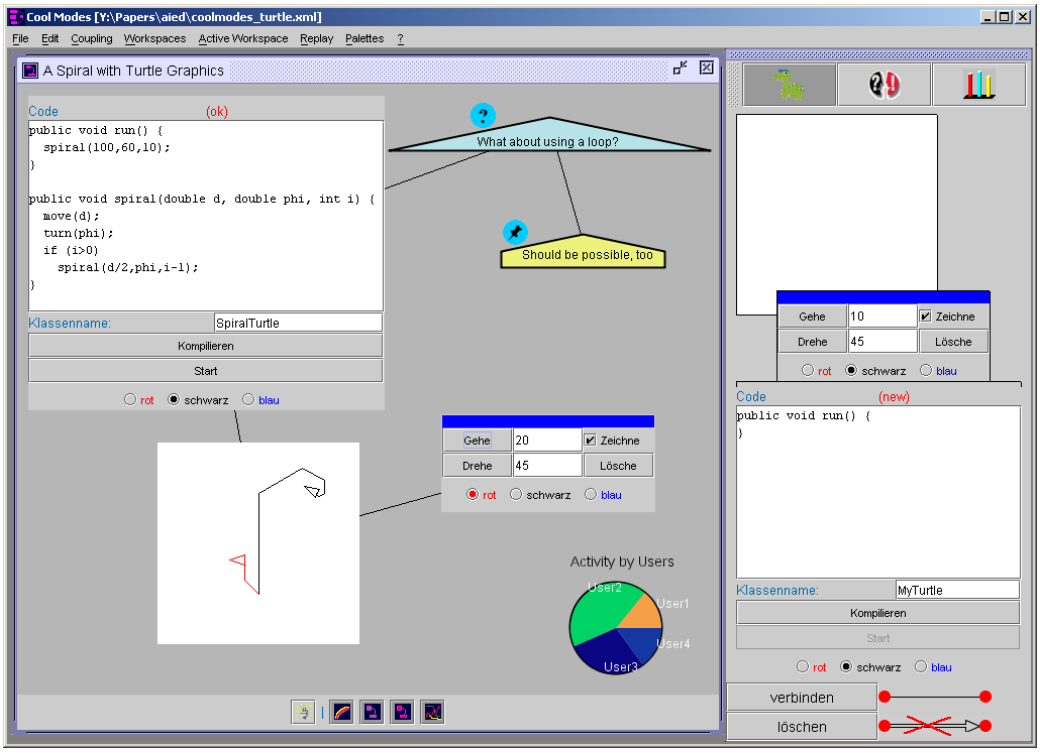

Figure 1. Cool Modes Screenshot

On the right side of the screenshot, you can see that there are currently three plug-ins loaded - one of them offering discussion support (icon with question and exclamation mark), another one that gives feedback to the users about their work (bar chart icon), and one that offers turtle graphics programming features within the framework. The user interface of the turtle plug-in shows the available objects to drag into the workspaces: one node at the bottom for Java-based turtle programming, the middle one for direct control of a turtle and the one at the top for visualizing the results. Inside the workspace, the mentioned mixture of elements from different languages within one workspace is visible. The elements of the "discussion support" plug-in are used to allow the students to comment on the collaborative programming effort; the pie chart feeds back to the users their amount of participation in the group work (here measured in number of contributions to and

modifications of the workspace content: user 2 has been the most active one in that example).

3. Plug-Ins for Cool Modes: Reference Frames

In an extensible system such as Cool Modes, there will be various representations being created and manipulated by the user, which may or may not be related to each other. Thus, besides the task of being able to define flexible representations, we need a way to know not only the semantics of individual representations, but also how they relate to each other – which belong together and are to be interpreted as a unit, and how this interpretation can be done. The structures that deal with these problems in the Cool Modes environment are called "reference frames" and consist of entities and rules that belong together. The entities are available to the user for direct manipulation and describe objects (nodes) and possible relations between them (edges). The rules offer interpretation methods for the structures that can be generated from entities of the frame. A reference frame also contains the description of a user interface (called "palette") and defines constraints for implicit integrity conditions, e.g., to forbid the connection of places and places in a Petri Net.

3.1 Definition and Integration of Reference Frames

The definition of a reference frame for the Cool Modes System can be done data based or code based. While the latter is more flexible but requires some abilities in Java programming and essentially comes down to implementing an interface, the first solution is more suited for non-programmers. It uses XML and tries to encapsulate as much model information as possible from the reference frame into the data file. The Document Type Definition is shown in figure 2.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- DTD for Reference Frames -->

<!ELEMENT ReferenceFrame (TemplateClass?, Objects, Metadata)>

<!ELEMENT TemplateClass (#PCDATA)>

<!ELEMENT Objects (Node*,Edge*,Rule*)>
<!ELEMENT Node (ClassRef|(Model|View|Controller))>
<!ATTLIST Node id CDATA #REQUIRED>
<!ELEMENT Edge (Model|View|Controller)>
<!ATTLIST Edge id CDATA #REQUIRED>
<!ELEMENT Rule (EgdeRule|CycleRule)>
<!ATTLIST Rule Message CDATA #IMPLIED>
<!ELEMENT Model ANY>
<!ELEMENT View ANY>
<!ELEMENT Controller ANY>
<!ELEMENT ClassRef (#PCDATA)>
<!ELEMENT EdgeRule (NodeRef,NodeRef,EdgeRef)>
<!ELEMENT CycleRule (NodeRef*,EdgeRef*)>
<!ELEMENT NodeRef (#PCDATA)>
<!ELEMENT EdgeRef (#PCDATA)>

<!ELEMENT Metadata (Package*,Name,Icon?)>
<!ELEMENT Package (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Icon (#PCDATA)>
```

Figure 2. The DTD for a Reference Frame

Each reference frame description consists of (beside some meta information like its name, icon and package information) the nodes and edges it offers. These objects can be defined in XML or by referring to a class file (more details on the approaches for the XML definition of the nodes and edges are outlined in [10]). The rules in the DTD hold the

implicit integrity conditions for the allowed structures: EdgeRules describe how many edges of a specific type are allowed between specific types of nodes (if no rule exists, the system does not limit the number); CycleRules define forbidden cycles consisting of specific edge and/or node types. The Class template contained in the DTD contains a link to a class that defines at least the controller and view aspects of the reference frame: the reaction to Cool Modes specific events like selection or deselection of the reference frame and the definition of the user interface. The treatment of rules that contain domain specific operational semantics, which is needed e.g. for dynamic behavior of specific models, is usually also realized in this class. A template with no specific behavior is taken as default.

The integration of the palettes into the framework is relatively easy – the code based extensions can be positioned anywhere in the Java classpath, the data based ones in a dedicated folder. Cool Modes then finds these definitions automatically and integrates the reference frame by dynamic class loading or by parameterization of existing templates.

3.2 Reference Frame Based Interpretation

The question of defining and integrating reference frames has been treated in the previous section. The interpretation of user-generated structures through reference frames is guided by the following principles (here simplified to the one workspace case):

- The workspace content built by the users is a graph (N,E) consisting of a set N of (typed and positioned) nodes and a set E of (typed) edges, connecting the nodes of N.
- For each node and edge type, there is exactly one reference frame that defines it. The set of nodes and edges defined by a frame F is called DEFINES(F). The set KNOWS(F) consists of all the node and edge types that F has detailed information about. As F knows at least the elements it defines, we have the relation KNOWS(F) $\supseteq$ DEFINES(F). Usually, these sets are not equal because frames may use "generic" elements defined elsewhere and because reference frames can extend each other to build hierarchy structures.
- Generic information like ownership or position of nodes is available to all frames.
- Each reference frame F can interpret the structure (N,E). The base for interpretation is the generic information about the whole graph structure and the detailed information about Knows(F). The interpretation of (N,E) by F is denoted by Ip (F,N,E).
- (N_F,E_F) denotes the largest sub graph of (N,E) with $N_F \subseteq$ KNOWS(F) and $E_F \subseteq$ KNOWS(F), the terms Ip_{gen} and Ip_{sem} stand for generic interpretation (without knowledge of node and edge types) and semantically enriched interpretation (with access to details about the node and edge models). Then, Σ denoting the set of available reference frames and $\otimes$ denoting an aggregation operator, the interpretation of (N,E) breaks down to: $\mathrm{Ip}(N,E) := \bigotimes_{F \in \Sigma} \left(\mathrm{Ip}_{gen}(F,N,E) \otimes \mathrm{Ip}_{sem}(F,N_F,E_F) \right)$.

Practically, this interpretation scheme is realized in Cool Modes in an event based way (for the concrete event types, see [11]): Upon any change in the workspace content (like, e.g., moving elements or changing their model), the system looks up the element that had caused the change and distributes the event as a "local change" to all the neighbor nodes and edges (to simplify local interpretation like, e.g., activating a transition in a Petri Net) and as a "global change" to all the reference frames. This way, the generic interpretation is enabled. The specific, semantically enriched, interpretation, is completely left to the reference frames. These can make use of the mentioned events, but also define different mechanisms when needed.

4. Examples

A number of different reference frames have been implemented within master thesis, for teaching purposes, within project work [12,13], or are currently under development. They can be classified into four basic categories. To illustrate the flexibility of the approach (all the plug-ins are fully cooperative and can be used synchronously), here a short description of the categories and some brief examples:

The first type is mainly designed to support collaboration, information exchange and knowledge communication. Examples are a reference frame with nodes to support structured discussions (with elements like "question", "answer" or "hypothesis", see figure 1) or one that supports handwritten input. Both plug-ins do not interpret any models but provide the users with collaboratively useable means of expression.

Modeling languages like Petri Nets or System Dynamics are examples of the second existing type of reference frames. Here, the collaborative functionality reached by means of sharing objects is "only" an add-on to the "real" field of application – the interpretation of graph structures by algorithms and conditions as described in the reference frame. These reference frames are typically designed to be used together with those described in the first type: e.g. the collaborative construction of a Petri Net model can be supported quite well with "informal" shared comments.

The third category of reference frames can be characterized with the term "meta operations". Their functionality is based on that of other reference frames, examples include the analysis of user interactions (a simple reference frame that counts discussion contributions per user and mirrors that statistics back to them has already been developed, see figure 1) and the intelligent provision of feedback. Reference frames have "sensors" that can be used for user interaction analysis and "effectors", e.g. the option of modifying the current workspace content. A domain independent XML based checking and feedback mechanism has recently been realized [14].

The integration of nonstandard input and output devices makes up the fourth type of reference frames. These developments try to improve the usability of the overall system by offering to the user flexible and appropriate ways of interacting with the system. While handwritten input is accepted by default (cf. 4.1) and approaches of adapting the whole Cool Modes environment to specific use cases on PDAs have also been undertaken successfully [15], the inclusion of other devices like pens with optical character recognition (for transfer of paper-based material into the digital environment) is currently in development. Not only usability issues, but also pedagogic claims are met with the inclusion of real data sources as a base for modeling: within the COLDEX project, a reference frame that allows access to real-time seismic data in Chile has been developed. It will serve as input for e.g. school or university courses dealing with the phenomenon of earthquakes – we expect a greater motivation for students when they know that the data they are dealing with is real.

5. Evaluation and Outlook

The Cool Modes environment has been used in several courses (school and university level) in different domains, ranging from language learning to mathematics. We have conducted (informal) interviews with the teachers of these courses – they were mostly content with the environment and gave constructive criticism on usability issues. We are currently working on modifying parts of the user interface to meet the mentioned points.

To evaluate the plug-in meachanism of the reference frames and in particular its ease of use and extensibility (which is the central point of the system apart from usability), we are planning to distribute a questionnaire to the people who have developed reference frames. The questions will treat their experiences with extending the system, and their view of typical difficulties doing so, we will relate these answers to a self-evaluation of the users concerning programming and general IT abilities. We hope to get an impression how skilled one has to be in order to a) use the system and b) develop plug-ins for it.

A topic of current research is motivated by the question in how far transitions between reference frames can be realized in a generic way. This would allow a definable and easy shift method between different modeling languages, which would be extremely useful, e.g., for UML reference frames (transitions between different diagram types) or the System Dynamics modeling frame (automatic change between quantified models and causal feedback loop models).

References

[1] Suthers, D., Connelly, J., Lesgold, A., Paolucci, M., Toth, E., Toth, J., & Weiner, A. (2001) Representational and Advisory Guidance for Students Learning Scientific Inquiry. In Forbus, D. & Feltovich, P. (Eds.): *Smart Machines in Education*, pp. 7-35. AAAI Press, Menlo Park

[2] Conklin, J. & Begemann, M. L. (1987). gIBIS: A hypertext tool for team design deliberation. In *Proceedings of Hypertext'87* (pp. 247-251). Chapel Hill, NC (USA).

[3] 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.

[4] Milrad, M., Spector, M. & Davidesen, P. (in Press). Model Facilitated Learning. Book chapter to appear in *"E- Learning: Technology and the Development of Learning and Teaching"*. Kogan Page Publishers UK.

[5] Constantino-González, M. & Suthers, D. D. (2000). A Coached Collaborative Learning Environment for Entity-Realationship Modeling. In Gauthier, G., Frasson, C. & VanLehn, K. (Eds.): *Intelligent Tutoring Systems* (Proceedings of ITS 2000, Montreal, Canada, June 2000). Berlin: Springer.

[6] Koedinger, K.R., Suthers, D. & Korbus, K.D (1999). Component-based construction of a science learning space. *International Journal of Artificial Intelligence in Education*, 10.

[7] Repenning, A., Tager, S. & Treinen, M. (2000). Reusability and Interoperability of Tools for Mathematics Learning: Lessons from the ESCOT Project. In *Proceedings of Intelligent Systems & Applications at University of Wollongong,* pp. 664-669. ICSC Academic Press, Wetaskiwin, Canada.

[8] Kynigos, C. & Koutlis, M. (2002). E-Slate: A "black and white box" approach to component computing. Paper presented at the Annual Meeting of the American Educational Research Association.

[9] Jansen, M., Pinkwart, N. & Tewissen, F. (2001) MatchMaker - Flexible Synchronisation von Java-Anwendungen. In *LLWA 01 – Proceedings of GI-Workshop "Lernen-Lehren-Wissen-Adaptivität"* (eds. Klinkenberg, Rüping & Fick), pp. 180-186. Forschungsberichte der Universität Dortmund.

[10] 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.

[11] Pinkwart, N., Hoppe, H.U., Bollen, L. & Fuhlrott, E. (2002) Group-oriented modeling tools with heterogeneous semantics. In *Proceedings of ITS 2002* (eds. Cerri, Gouardères & Paraguacu), pp. 21-30. Springer, Berlin.

[12] SEED, Seeding Change in the School System through the Generation of Communities Engaged in Integrated Educational and Technological Innovation. EU IST project No. 2001-25214

[13] COLDEX, Collaborative Learning and Distributed Experimentation. EU IST project No. 2001-32327

[14] Herrmann, K., Hoppe, H.U. & Pinkwart, N. (2003). A Checking mechanism for Visual Language Environments. To appear in *Proceedings of AIED 2003*.

[15] Pinkwart, N., Schäfer, C. & Hoppe, H.U. (2002). Lightweight Extensions of Collaborative Modelling Systems for Synchronous use of PDA's. In *Proceedings of IEEE Workshop on Wireless and Mobile Technologies in Education*, pp. 125-129. IEEE Press, Los Alamitos, CA (USA).