Modeling Multiuser Interactions

Cornelia Haber[*]
Carl von Ossietzky Universität Oldenburg
Fachbereich Informatik, Escherweg 2, D-26121 Oldenburg

Abstract

Due to the growth of the Internet and the fact that more and more people have access to networked computers research on computer supported collaborative learning (CSCL) has gained importance. However development of CSCL environments is expensive. Developers need to consider e.g. distributed computing and team-building and content presentation. Even more difficult is the development of collaborative multimedia teaching/learning software. To reduce development time and costs models and tools are needed that support developers of CSCL applications. This paper describes MoMI, a model for multiuser interactions. With the help of MoMI developers may specify how various users take part in an interaction. The main focus of MoMI is on collaborative interactions in synchronous applications, i.e. interactions where more than one user has to take part in the interaction. MoMI specifies e.g. that an interaction is accepted by the application after all users pressed the same button. Another specification may be that a selection between different films is taken if at least half of the users agree on the same film. There are various other examples for multi-user interactions.
In this paper single user and multiuser interactions are compared, characteristics of multiuser interactions are discussed and a model for specifying multiuser interactions is presented.

Keywords: CSCL, multiuser interaction, synchronous multiuser application

1 Introduction

The wide distribution of networked computers in private households combined with the growth of the Internet helped to establish the World Wide Web as an important source of information for everyday users. At the same time research in cooperative and collaborative learning showed that these new concepts of learning have some positive effects on the learning process and on learners. Bringing together these two aspects research in CSCL (Computer Supported Collaborative Learning) started.
One of the drawbacks of CSCL software is the fact that developing CSCL environments is rather expensive due to the complexity of the software. CSCL developers need to consider the distribution of the software, networking facilities, social aspects of working in groups, user interface design and last but not least the contents that are to be learned.
One major aspect of CSCL environments is the way people work together, communicate and know about each other. Today CSCL environments consist mainly of teaching material, newsgroups and chats for the learners to discuss their ideas and tools like shared whiteboards or shared web browsers [WPM00].
On the other hand more and more single user multimedia learning applications are developed like e.g. virtual laboratories ([BBD$^+$98]) or just applets illustrating algorithms ([Fer]). Those applications rarely support multiple users. One scenario for a multiuser multimedia learning application could be a simulation of a plastic surgery. Every user taking part in the application has a special role to fulfill (surgeon, nurse, ...). Depending on the role the user can interact with the application (e.g. the nurse is not allowed to use the scalpel on the patient but only to hand the scalpel to the surgeon). This kind of application would not only help the surgeon to get used to his/her part of the application moreover surgeon and nurse would learn how they have to cooperate in order to complete the surgery. Another example for a multimedia multiuser application is a language laboratory. Each pupil in the laboratory is working on the assignment for himself. The teacher can watch one pupil at a time and then either decide to help the pupil and contact him directly e.g. via chat, phone, etc or decide that the pupil does not need

[*]Email: haber@informatik.uni-oldenburg.de

help and move on to watch the next pupil. Development of multiuser multimedia learning systems is desirable but it is very difficult due to the complexity caused by multimedia and multiuser aspects.
This paper introduces MoMI, a **Mo**del for **M**ultiuser **I**nteractions, which is intended to support the development of such multiuser multimedia learning systems. MoMi specifies multiuser interactions, i.e. interactions where more than just one user take part in the interaction. The paper is structured as follows: The first section discusses multiuser applications in general and the ways users cooperate with one another via the application. Then a short comparison between single and multiuser interactions is given. The next section discusses how single user interactions can be described as IfCA rules. These rules are later enhanced to support simple multiuser interactions. Based on the simple multiuser interactions complex multiuser interactions are developed. The paper closes with a discourse on how MoMI can be used in the development of synchronous cooperative applications and a short summary.

2 Multiuser applications

There is a wide variety of multiuser applications. They differ especially in the ways people work together. We distinguish the following three application classes:

- shared applications
- multiuser applications with one model
- multiuser applications with different models

The differences between these application classes shall be described with the help of the Model-View-Controller (MVC) paradigm. There the user input, the modeling of the external world and the visual feedback to the user are separated and handled by model, view and controller objects. The model object manages one or more data elements that are to be presented at the user interface. The controller object interprets inputs from the user (e.g. from mouse and keyboard) and maps these user actions into commands that are sent to the model and/or view object to effect the appropriate changes. The view object is responsible for the presentation of the dates at the user interface. Figure 1 shows the differences between the three application classes according to the MVC paradigm.

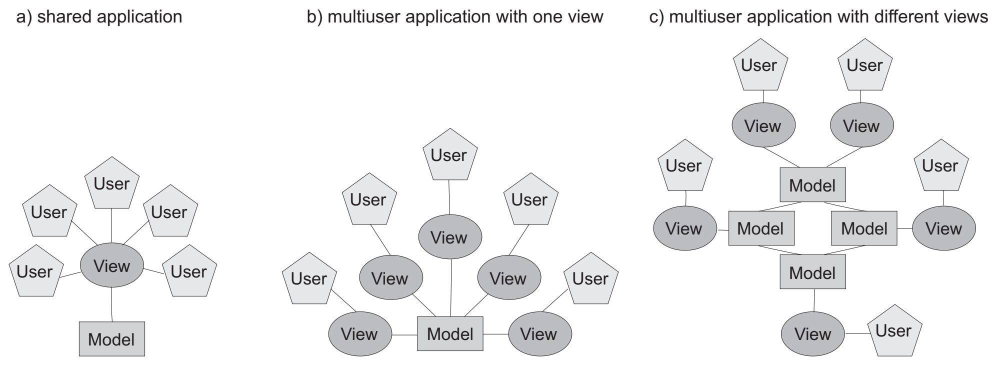
a) shared application
b) multiuser application with one view
c) multiuser application with different views
User
View
Model

Figure 1: Multiuser application classes

The first class is the one easiest to develop. Shared applications are single user applications made multiuser with the help of special tools like shared X. As the application itself is a single user application there is no differentiation between users, all user inputs are taken as input from one user and whatever action a user interaction causes is the same for all users. User inputs can be synchronized e.g. with the help of floor control mechanisms. The screen presentation of the application is identical for all users (WYSIWIS). The application sharing softwares responsibilities include maintaining consistent shared views, managing floor control between participants wishing to interact with the application, registering participants. Using the MVC paradigm as in figure 1a) a shared application has one model that is used for all users and only one view for all users. Therefore the presentation of the information on the users screens is identical for all users. For more information on shared applications see [Gre90]. As every application can be used as a shared application without any alteration to the application itself no special multiuser models or tools are needed to develop those applications.

More difficult to develop are multiuser applications with one model as shown in figure 1b). Such an application implies that all users still have the same model, but the different users may have different views of the application, i.e. no strong WYSIWIS is expected. In our definition all view objects are parameterized objects of the same view class. That way the application may focus on details for a user with high network connectivity and be not that detailed for users with poor connectivity. Another example is that some users have added functionality due to their role in a team like e.g. the teacher in a learning application. Besides those small alterations the application is quite the same for all users and all users have similar views of the application. The views may still differ because of network latency but they are synchronized through the application.
The last group of multiuser applications are applications where each user may have a different model and a different view on the application as figure 1c) shows. Users are coupled loosely, i.e. each user works for himself. As the models of the different users are coupled the actions of a user may influence other users applications at some point within the application but they don't have to. The whole functionality of the application may differ for different users. The MoMI model introduced in this paper focuses on multiuser applications with one model.

3 MoMI - a model for multiuser interactions

The first step in the direction of a model for multiuser interactions is a close examination of multiuser interactions. The following section compares single user interactions to multiuser interactions.

3.1 Comparing single user and multiuser interactions

In the field of multimedia an interaction is defined mainly as the way a user may influence the application flow [Bol94]. An interaction is characterized by the interaction task, interaction technique, interaction form, interaction device and the effect of the interaction. According to [FvDFH98] an interaction task is the entry of a unit of information by the user. Interaction tasks classify the fundamental types of information entered with the interaction techniques. Interaction tasks are defined by what the user accomplishes, not how it is accomplished. The four basic interaction tasks are select, position, quantify and text. Select allows the user to choose from a set of choices, position allows the user to specify a position in screen or world coordinates, quantify allows the user to specify a numeric value and text provides unvalidated data entry and output messages to the user.
Interaction techniques as defined in [FvDFH98] are ways to use input devices to enter information into the computer like pressing a mouse button. According to [Bol94] interaction forms are the software part that inputs dates into the computer (e.g. masks and user interface components). The input device is a piece of computer hardware that allows a user to interact with a computer like mouse and keyboard. The interaction effect specifies the reaction the user expects from the application. The characteristics of multiuser interactions are quite similar to those of single user interactions. Still there are a few differences. Table 1 outlines the differences between single user interactions and multiuser interactions.

	Interaction	**multiuser Interaction**
interaction task	position text select quantify	multiuser position multiuser text multiuser select multiuser quantify communication
interaction technique	mouse down mouse move press keyboard …	mouse down mouse move press keyboard …
interaction form	masks UI components …	masks multiuser UI components …
interaction devices	mouse keyboard …	mouse keyboard …
interaction effect	reaction of the application	reaction of the application (local) reaction of the application (global)

Table 1: Comparison between single user and multiuser interaction

In a single user application an interaction can only be started by one user: *the* user. In a multiuser application an

interaction can be started by one out of many users, or it may be started by a given number of users. Even the identity of the user may determine whether the interaction is started or not.
The main interaction tasks for multiuser interactions are still position, text, select and quantify, but the meaning of each of these tasks has changed. This will be explained by a short example. Selecting an object in a single user application is a well known task. The object is selected by one user on the screen. In a multiuser application it is to be considered which users have to select the object. Is it sufficient for one user to select the object, for a group of users, or do all users have the task to select the object? Therefore it is not only a select, but a multiuser select where the users whose task the selection is have to be specified. The same applies to the other interaction tasks (text, position, quantify) which are enhanced to multiuser text, multiuser position, multiuser quantify in a multiuser environment. As there are multiple users taking part in a multiuser application the interaction task may even be the communication between users. Interaction techniques are the same in multiuser applications as in single user applications. Assuming that each user is still sitting in front of his/her own computer, no new interaction techniques are needed. The same applies to interaction forms. However to make the development of multiuser applications easier it would be desirable to have multiuser UI components that encapsulate e.g. the communication between the components on the different user computers. Interaction devices do not change in multiuser applications. The main interaction devices are still mouse and keyboard. The effect of an interaction in a single user environment is a reaction by the application. In multiuser applications the effect is a reaction by the application as well, but this reaction may affect different users so that it is not only a local reaction but it can be a global reaction concerning an arbitrary set of users. One way to model single user interactions is in the form of IfCA rules.

3.2 Modeling a single user interaction as IfCA-rule

Interactions in single user applications may be modeled as **E**vent **C**ondition **A**ction rules (ECA rules), i.e. on occurrence of an event a condition is evaluated and depending on the evaluation an action is triggered. The event can be caused by the application itself or by the user. [Göt95] differentiates between so called base events and complex events. Base events are caused by atomar actions and are not decomposable. There are five types of base events:

Mouse events are initiated by a user pressing or releasing a mouse button or moving the mouse.
Keyboard events are events triggered by typing at the keyboard.
Time events are events initiated by a system clock. They may relate to absolute times like e.g. 4.00 a.m. or relative times like "half an hour after starting the application".
Application events are events caused by an application without any user interaction. The events may be caused by the application itself or by another application.
Variable events are triggered on changing the value of a variable. Therefore they may but need not be caused by a user through a user interaction. There is no need for a user interaction to take place in order to initiate a variable event.

According to this classification selecting an object on the screen with the mouse is a complex event because it consists of pressing the mouse button and afterwards releasing it again. ECA rules allow for specifying base events or complex events (sequences of events as generated by user interface components) as the event in the ECA rule. In this paper they are called "interaction form event" or short IF and the rules are not called ECA rules but IfCA rules to emphasize the use of complex events caused by interaction forms.

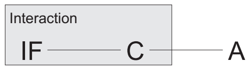

Figure 2: Relationship between IfCA rule and interactions

A condition in an IfCA rule is a boolean expression on the values of attributes within the application or concerning the application environment. A condition may restrict an interaction on the IP-address of the computer the application is running on, the time the interaction takes place or in the case of a user filling text into a text field it may make restrictions on the text the user typed. Conditions can be combined using the logical operators AND or OR thus allowing for complex conditions. The actions an IfCA rule can describe are manifold. The action can e.g. change the value of an attribute of an object within the application, start the display of a video film, or create a new event. There are no restrictions to the actions an IfCA rule can specify. Figure 2 shows the relationship between IfCA rules and interactions. An interaction consists of the IF and condition part of the IfCA rule. The action itself is not part of the interaction. So far we only considered single user interactions. The following section discusses using IfCA rules for modeling multiuser interactions.

3.3 Modeling multiuser interactions

3.3.1 General considerations

Ere the MoMI model for modeling multiuser interactions is introduced some general considerations need to be discussed. The first item to be considered is when an interaction should be aborted. This question is important for single user interactions as well, but it gains importance in multiuser interactions as multiuser interactions may take some time if e.g. every user has to take part in the interaction. Another situation may be an application where the users have to vote. What happens, if not all users want to take part in the voting? Suppose one user doesn't make a choice at all. For situations like this it must be possible to give time restrictions for a multiuser interaction. Moreover it is to be modeled how the application continues after an interaction is aborted. There are three main possibilities.

- The interaction is dropped altogether. It is not reinitialized and the user actions concerning this interaction are ignored. The application proceeds just as if this multiuser interaction did not exist at all. Local actions have to be rolled back and the status of the application has to be restored to the status at the time when the interaction started.
- The interaction is reinitialized, i.e. the possibility for the interaction is there again just as if the interaction was started at that very moment. The actions so far by the different users in order to fulfill the interaction condition are ignored and nothing else happens.
- The interaction is dropped and the application proceeds at a spot defined for the abortion of the interaction. This may be a spot specially designed to handle the aborted interaction or just any spot within the application.

A multiuser interaction can be aborted for two reasons:

Time restriction: For each multiuser interaction the interaction modeler has to decide whether time restrictions shall be applied. If there are to be time restrictions they have to be specified. The first way to specify a time restriction is relative to the start of the multiuser interaction, that is the time the multiuser interaction object is initialized. The second way to give time restrictions is relative to the first IF within this multiuser interaction, i.e. the moment the first IF relevant for the multiuser interaction occurs time is running.

Impossible condition: An interaction should be abortable if the condition of the IfCA rule describing the interaction cannot be fulfilled. This may be the case if e.g. the condition specifies that at least five users have to click on a button but there are only four users taking part in the application. For each multiuser interaction the modeler may specify if an interaction shall be aborted if the interaction condition cannot be fulfilled or if this fact shall be ignored in the hope that the condition may be fulfilled later.

The modeler has to specify for each multiuser interaction if the interaction can be aborted and where to proceed with the application after an interaction is aborted. There are always the three different ways discussed above to proceed with the application. On the other hand the modeler may specify that the interaction is never aborted at all but stays alive and all IFs are considered until the condition of the IfCA rule is fulfilled.

3.3.2 Modeling simple multiuser interactions as IfCA rules

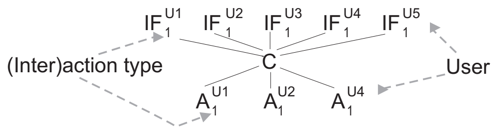

Figure 3: IF^nCA^n rule

The IfCA rules described in 3.2 are not sufficient to describe multiuser interactions. As discussed in section 3.1 a multiuser interaction may only take place if a given number of users were actively involved in the interaction. This situation cannot be described by one IF. Rather a number of IFs are needed, one for every user involved in the start of the interaction. Consider the example of a multiple choice selection of films. The users may decide on the film that is shown. After every user made his/her selection that film is taken which had the most supporters. The interaction in this example is finished after every user made the choice and the application decided on the basis of the user choices. In this case the application will wait for a selection IF of every user, evaluate the condition after each incoming IF and only after all users have chosen start the action (the film). As the film will start on all user screens, the action is multiplied as well as there is not one action, but one action per user. The IfCA rule is therefore enhanced to an IF^nCA^n rule where the IFs are identical. They differ only in the user causing the IF as figure 3 shows.

There is also a slight difference between the conditions for single user IfCA rules and multiuser IfCA rules. In a single user application the condition may be specified regarding the following criteria:

Attributes: A condition on attributes compares the value of an attribute of an application object to a given value or a given range of values.

Time: A condition concerning time may restrict the interaction to a time interval during which the interaction has to take place. The time interval can be specified absolute (e.g. after 8.00 a.m) or relative (e.g. after another interaction took place).

Location: The location of the IF specifies where (on which computer) the interaction takes place. This kind of condition is interesting for security relevant interactions that are allowed on special computers only.

For multiuser interactions supplementary condition types are needed:

Access rights: With the help of access rights an interaction may be restricted to special users or user groups. IF caused by users not belonging to a certain group are ignored.

Number of users: Not only the identity of the user may be important for the interaction. Another criteria may be the number of users causing an IF. This condition allows developers to specify interactions that are started only if a given number of users is involved in the interaction. The number can be specified explicitly (e.g. five users) or implicitly (e.g. half the users) and may be combined with the access rights (e.g. three users of group 1).

In a multiuser IfCA rule condition the different condition types may be combined using logical operators. That way you may specify rather complex interactions. Take a program administrating system resources as an example. Only the system administrator is allowed to start the backup of the system and only after 6.00 p.m. whereas all users are allowed to start other processes.
As some time may be taken from the first user causing an IF to the last user IF needed to start the interaction there should be a mechanism to include feedback to the users about the current status of the interaction. [tH98] differentiates feedback and feedthrough where feedback is the reaction of an application caused by an IF to the user causing the IF and feedthrough is the reaction of the same application to all other users (not causing the IF). In order to support feedback and feedthrough the IfCA rule has to be enhanced again. Each IF has methods attached to it, one defining feedback and one defining feedthrough. If neither feedback nor feedthrough is desired the methods can be ignored. After the integration of feedback and feedthrough the rule has the form $(IF \times Fb \times Ft)\,^nCA^n$ where Fb is the feedback function and Ft the function for feedthrough. Simple multiuser interactions can be represented as shown in figure 4.

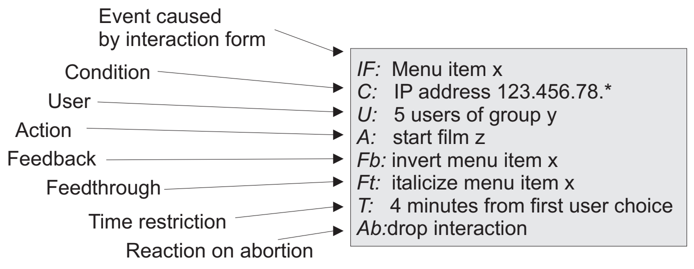

Figure 4: $(IF \times Fb \times Ft)^nCA^n$ rule

As all the IFs in an $(IF \times Fb \times Ft)^nCA^n$ rule are identical only simple multiuser interactions can be specified. The rules allow only for interactions where each user has the same part in the interaction or does not cause any IF important to that interaction at all. The next section will show how more complex multiuser interactions can be modeled.

3.3.3 Modeling complex multiuser interactions

A complex multiuser interaction is constructed out of a number of simple multiuser interactions and other complex multiuser interactions using linking operators. Thus the specifier has to decide on interactions already specified and the way he wants them combined. The following linking operators are allowed:

AND: Combining multiuser interactions with the AND operator causes the whole interaction to take place only if *all* the part-interactions took place. There is no restriction on the order the part-interactions have to take place in.

OR: The OR operator combines simple and/or complex multiuser interactions in a way that fulfills the interaction the moment *at least one* of the part-interactions is fulfilled.

SEQ: The SEQ (Sequence) operator lists the part-interactions in the order in which they have to occur. *All* part-interactions have to be fulfilled *in the order given* to fulfill the combined multiuser interaction.

In multiuser applications it may be better to specify feedback according to part-interactions and not according to IFs. Therefore MoMI allows for defining feedback together with the logical operators. The feedback is valid for *all* users and takes effect when the interaction condition specified with the logical operator evaluates to true. Nevertheless this feature should be used prudently as too much feedback may confuse the user. Therefore specification of a feedback method for combined multiuser interactions is optional. Graphically a complex multiuser interaction can be represented as a tree. The leafs represent simple multiuser interactions and the inner nodes of the tree represent the operators used for combining multiuser interactions. Each inner node (including the root node) has two to n sons. Figure 5 shows such a multiuser interaction tree.

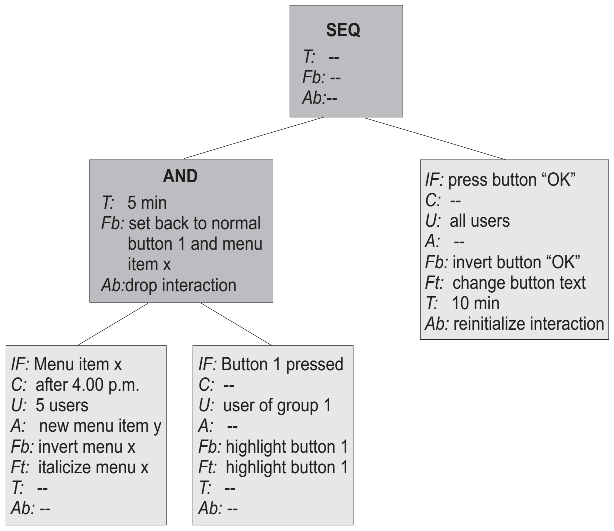

Figure 5: Multiuser interaction tree

In fact this tree describes a rather complex multiuser interaction. Because of space limitations the tree is not a complete specification rather it indicates the whole interaction. For a complete specification e.g. the time specification need to be given in more detail (when does the time counter start? After the initialisation of the interaction or after the first IF?). In order for the interaction to fulfill a number of IFs have to occur. The whole interaction is a sequence which means that the son interactions have to be in a special order. In our example first the AND interaction has to take place and only after the AND tree is completed the other subtree will be considered. The AND tree demands 5 users to choose menu item x. As feedback the menu item is inverted and as feedthrough it is italicized. Moreover 1 user of group 1 has to press Button 1. Both part interactions have to take place within 5 minutes otherwise the whole interaction is dropped (time restriction with the AND operator). After the AND interaction is fulfilled the presentation of button 1 and menu item x is set back to "normal". Then according to the right subtree all users have to confirm the choice by pressing the OK button within 10 minutes.

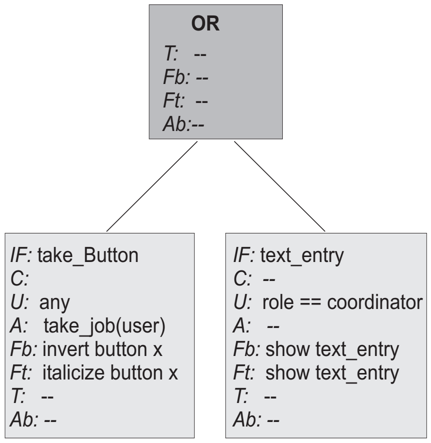

Figure 6: Assigning jobs

This example sounds rather academic and it seems hardly likely that any interaction of this kind will ever be needed in real life, but interaction trees are sensible for all multiuser interactions where various people have to cause different events for the interaction to take place. And even if these interactions are usually not that complicated it makes sense to use a special notation for them. Take the real life situation shown in figure 6 as an example.

In this example there are a few unpopular jobs to be assigned to the users of a group. Each user can either offer to take a job (click on the take_Button) *or* just sit still in the hope of not getting a job. The group coordinator on the other hand may assign jobs to the users (the event text_entry caused by the coordinator triggers the action assign_task). One important fact to be mentioned here is the structure of an interaction tree. It is not necessarily a binary tree, rather it is a n-ary tree where each node has at least two sons.

3.4 Using the MoMI-Model

There is no sense in developing models if there are no situations where the model is actually used. This use may either be to discuss and understand the situation the model describes or to support the development of an application. The MoMI model may be used for both scenarios. It helps to understand the ways in which people work together to cause a multiuser interaction. On the other hand it helps developing multiuser interactions in cooperative applications. How the MoMI can be used for theses tasks will be discussed in this section.
MoMI is not very useful for multiuser applications with different models as MoMI is intended for applications where users work together closely. In applications with different models multiuser interactions where different users cooperate in the interaction are rare but if there are some they can be modeled using MoMI. Still other tools and models need to be introduced to support applications of this application class.
MoMI should be used to develop multiuser applications with one model as those applications stress tight synchronous cooperation of users. To develop such an application you may take a model for developing single user multimedia presentation applications like the IMRA model [Bol95] and combine it with MoMI. The IMRA model models an application as a set of objects and relationships between the objects. Exchanging the interaction objects in the IMRA model with MoMI multiuser interaction objects a multiuser multimedia presentation application can be modeled quite easily.

4 Summary

MoMI is a model for modeling multiuser interactions for synchronous, cooperative applications. The model is suitable for complex interactions where different users cooperate in order to trigger the interaction. The focus of MoMI is on modeling the cooperation between users.
For the utilization of MoMI tools are needed like a MoMI editor for building interaction trees. Other helpful tools would be a class library offering the basic functionality needed for interaction trees, a simulation component where the interaction the interaction tree specifies can be simulated or a code generation unit where code is generated out of the interaction tree, offering hooks for developers to integrate feedback and feedthrough.
Right now there are no tools supporting MoMI but we intend to implement an editor in the near future. More tools will hopefully follow later on.

References

[BBD+98] Dietrich Boles, Eckhard Boles, Peter Dawabi, Marco Schlattmann, Claudia Trunk, and Frank Wigger. Objektorientierte Multimedia-Softwareentwicklung: Vom UML-Modell zur Director-Anwendung am Beispiel virtueller naturwissenschaftlich-technischer Labore. In *Tagungsband zum Workshop "Multimedia-Systeme" im Rahmen der GI-Jahrestagung 1998*, pages 31–51, 1998.

[Bol94] Dietrich Boles. Das IMRA-Modell - Integration von Interaktionen in das Autorenwerkzeug FMAD, 1994. Technischer Bericht der Carl von Ossietzky Universität Oldenburg.

[Bol95] Dietrich Boles. Das IMRA-Modell - Modellierung interaktiver multimedialer Präsentationen. In Rainer Kuhlen and Marc Rittberger, editors, *Hypertext-Information Retrieval-Multimedia(HIM'95)*, pages 61–75. Universitätsverlag Konstanz, 1995.

[Fer] Luis Fernandes. The Abacus: The Art of Calculating with Beads. Available under the URL: http://java.sun.com/applets/archive/beta/Abacus/index.html.

[FvDFH98] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 2nd edition edition, 1998.

[Göt95] Rainer Götze. *Dialogmodellierung für multimediale Benutzerschnittstellen*, volume 14 of *Teubner-Texte zur Informatik*. B.G. Teubner, Leipzig, 1995.

[Gre90] Saul Greenberg. Sharing Views and Interactions with Single-User Applications. In *Proceedings of the ACM/IEEE Conference on Office Information Systems*, pages 227–237, 1990.

[tH98] G.H. ter Hofte. *Working apart together : Foundations for component groupware*, volume 001 of *Telematica Instituut Fundamental Research Series*. Telematica Instituut, Enschede, Netherlands, 1998.

[WPM00] Martin Wessner, Hans-Rüdiger Pfister, and Yongwu Miao. Umgebungen für computerunterstütztes kooperatives Lernen in der Schule. *informatica didactica*, 2000.