

„DreamTeam“ - a Synchronous CSCW Environment for Distance Education

Jörg Roth, Claus Unger

University of Hagen, Department for Computer Science, 58084 Hagen, Germany
Joerg.Roth@Fernuni-Hagen.de, Claus.Unger@Fernuni-Hagen.de

Abstract: This paper presents a platform for the development and execution of synchronous co-operative shared applications in a distributed education environment. Even though several co-operative environments are existing nowadays, specific problems in real student environments are not treated satisfactorily. Students often have to deal with unstable network connections and low bandwidths; they are using different hardware and software platforms; their computers are off-line most of the time.

The DreamTeam platform addresses the special needs of distance education environments. It comprises a development environment, a runtime environment and a simulation environment; it is based upon the concept of a de-centralised architecture. It minimises the development cycle for co-operative applications and encourages rapid prototyping.

DreamView, a co-operative Web browser developed in the DreamTeam environment, demonstrates the strengths of the DreamTeam environment.

1 Introduction

Co-operative (CSCW) applications are playing a major role in distance education, e.g. for group discussions, jointly working on electronic courses, jointly visiting Web pages, remote laboratories, video conferencing, joint program development, co-operative publishing, etc. To develop such a co-operative application is a difficult and time-consuming task.

Dommel et al. [Dommel et al. 97] distinguish three types of co-operative applications:

- *Collaboration unaware applications* offer no collaboration services themselves; they are single user applications, running in a shared environment.
- *Collaboration aware applications* are developed for co-operative environments, but their services for collaboration are hard-coded.
- *Collaboration transparent applications* provide their services for collaboration by using high level services of a standard collaboration environment.

Taking the last approach, a developer can concentrate on the application specific details of his application and can use the collaboration oriented services from the standard collaboration environment.

In the following, we describe DreamTeam, a Java oriented development and runtime environment for synchronous, collaboration transparent applications for distance education. As an example for such an application we further describe DreamView, a co-operative Web browser, which has been developed within the DreamTeam environment. In the following, the terms co-operative application or groupware are used as shorthands for a collaboration transparent application running in a shared distributed environment.

2 Co-operative Applications for Distance Education

Co-operative applications for distance education have to take into account our students' special needs and requirements.

User interface: Especially for remote students it should be as easy as possible to install and run a co-operative application. Different applications should provide consistent interfaces, at least for all aspects of installation and general collaboration.

System platforms: Most of our students are using PCs with a variety of different operating systems. A co-operative application should make as few assumptions as possible about the system platform with regard to hardware requirements (screen resolution, system performance etc.) and software requirements (operating system, version, installed software etc.). PCs tend to be unstable. A co-operative application should be tolerant against local breakdowns and should easily allow to restart a system and rejoin an existing group.

Network: Most of our students are using the Internet for communication, mainly through dial-up modem connections, connecting them to the university either directly or via a service provider. These connections are exposed to network delays, bandwidth degradations, as well as sudden, unexpected disconnections. A co-operative application should handle such situations itself without requesting any additional actions from the user, e.g. by reconnecting to the active group. Usually, students are off-line most of the time and their Internet addresses may change between consecutive dial-ins. It is the task of a rendezvous component to find out which students are actually available for a meeting and how they can be reached via the Internet.

3 The DreamTeam Environment

The DreamTeam environment allows the developer to develop co-operative applications like single user applications, without struggling with network details, synchronisation algorithms, etc. The environment consists of three components, a development environment, a runtime environment and a simulation environment. In the following, the basic DreamTeam concept as well as the three components are described in more detail.

3.1 The Concept

Architecture: Co-operative applications can either be implemented in a centralised or in a de-centralised way. In a centralised architecture, all messages are routed via a single group server, thus group events and data accesses can easily be synchronised. On the other hand, group servers often are performance bottlenecks, a shut down of a group server shuts down the session; all messages, even those between clients residing in the same host, have to be sent through the group server.

In a de-centralised architecture, availability and bandwidth problems can be avoided, but synchronisation and serialisation have to be handled by higher level protocols. On the other hand, a de-centralised architecture allows different kinds of communication channels, e.g. via ISDN connections.

The actual version of DreamTeam supports de-centralised architectures without the need for any central server.

Session management: In a de-centralised architecture, session management has to be handled in a de-centralised way too. The originator of a session defines a session profile and generates the session. As soon as he has started the session, other users can join and leave the session. When the originator himself leaves the session, the session can continue with the remaining members but no new members can join in.

Information distribution: To distribute information among session members, DreamTeam uses a special kind of Multicast Remote Procedure Calls, through which a local system can call a procedure at all participating remote systems. Complex data structures can be serialised, transferred as procedure parameters and rebuilt at the target site. All objects defining a DreamTeam session can be serialised; thus even for a late-comer the actual session state can easily be generated.

3.2 The Development Environment

The development environment provides a big object oriented class library based upon the standard Java library. It contains 190 classes with more than 2500 methods and is steadily expanded for new kinds of co-operative applications. It is structured into three layers.

The *base layer* offers general classes for user interface design, general data structures, general network services and other utilities, which are not groupware specific but are missing in the standard Java library.

The *service layer* contains groupware-specific classes. For, e.g., session management, data sharing, file transfer, distributed dialogues and special groupware widgets (telepointers, tracking of document views etc.).

The *application service layer* defines the only access to the service layer for application programmers. It includes security checks for all method calls, and thus protects the runtime system. Invalid calls are documented in a message log file. Details can be found in [Roth 97] and [Roth 98].

Using this class library, the DreamTeam front-end (see [The Runtime Environment]) as well as several co-operative applications have been developed so far, a co-operative Web browser will be described in more detail in [„DreamView“ - a Co-operative Web Browser].

3.3 The Runtime Environment

Besides providing services to shared applications, the runtime environment offers a set of functions to the end user. [Fig. 1] shows a typical DreamTeam desktop.



Figure 1. The DreamTeam working environment

The upper left main window allows to

- configure host and user profiles;
- retrieve user profiles of participants of previous sessions (lower left window);
- define and edit session profiles;
- start and stop sessions (icons representing open sessions are shown in the upper right window);
- start private applications (icons are shown in the lower right window);
- view users currently connected to a session.

3.4 The Simulation Environment

The DreamTeam simulation system allows the developer to test his distributed application on a single computer, and to simulate network effects like delays, reduced bandwidths, network failures, etc. To simulate the network, the participating applications are connected to each other via a special program (network simulator) rather than a real network. The simulator transmits every data block to the corresponding receiver. To produce delays and bandwidth effects, data blocks can be stored inside the simulator for a specified time before they are delivered to the receivers. To simulate network failures, the simulation can interrupt the data traffic for a certain time. A user can interactively control the simulator by a user interface, which also provides statistical functions.

4 „DreamView“ - a Co-operative Web Browser

4.1 The Concept

DreamView is the first substantial co-operative application developed in the DreamTeam environment. Beside providing a useful tool for distributed education, DreamView has been developed to test and validate the DreamTeam design concept.

DreamView allows a group of users to co-operatively browse the World-Wide Web. The reasons for implementing a co-operative Web browser are manifold. In addition to browsing remote documents, Web browsers can be used to browse local HTML documents; manuals and course materials can be published as HTML trees and distributed on CD-ROMs. In the following, three possible usage scenarios for DreamView are sketched:

A co-operative Web browser as a background tool: In a group session, it would be helpful to have a Web browser as a background tool. For example, a group is developing a program with a shared editor. Suddenly, a specific algorithm is needed. A Web browser is activated and the group visits a site with the corresponding course material. The participants read the specific chapter, find the algorithm, and continue working on their main task.

A co-operative Web browser as a document presentation tool: A Web browser can present formatted documents. In this case, the Web browser is the main tool in a co-operative session. It can be used, for example, to discuss software design documents, graphs and sketches etc. In contrast to the first scenario, internal group documents rather than external documents are discussed and can be annotated by all group members.

A browser as a presentation tool for computer mediated lectures: A co-operative Web browser can be used to give lectures to a student group. The teacher prepares a set of HTML pages. The students put their browsers into tracking mode. Each page and section is synchronised with the teacher's view. In addition, the teacher can use a telepointer to point to relevant items.



Figure 2. The DreamView browser environment

The browser represents itself as a normal Web browser [Fig. 2]. The largest area is used by the page contents. In contrast to other Web browsers, DreamView provides a separate area for annotations. In the annotation area, textual comments which can be anchored to the page items, can be stored. If necessary, it is possible to hide the annotation area in order to enlarge the page area.

4.2 Group-specific Services

To observe the activities of other users, a special window can be opened (see right bottom window in [Fig. 2]). It presents a radar view of the page area of another user. Each group member can open such a window for each other group member. If desired, a user can jump immediately to the shown page location. In addition, two views can be synchronised (follow mode) in order to follow the same page sections. This mode is helpful for lectures.

To point to specific page elements, a telepointer can be installed. Every user can make his mouse cursor public to other participants.

A problem arises when transferring position information between users: because each user has his or her own workspace settings (e.g. screen resolution, window sizes or fonts), it is not meaningful to transfer platform dependent absolute co-ordinates (e.g. pixel positions). In the case of telepointers, e.g., a textual mapping is more useful than a mapping based upon cartesian co-ordinates. For each mouse position, the browser determines the referenced item (e.g. a word or image). Then it transfers a reference to this item as well as the relative position of the telepointer inside the bounding box of this item to other users. The recipient can now restore the corresponding position, relative to his individual screen settings. If someone points to a specific word, a letter inside a word, or a detail inside an image, on every participant's desktop the telepointer points to the same corresponding item.

4.3 Development Costs

DreamView consists of the following parts:

- the main program, providing the browser window and configuration services;
- classes for the administration of annotations;
- classes for the HTTP connection, for parsing HTML pages and for loading images;
- classes for processing page contents and generating graphical output.

The following tables show the amount of code produced for each of these parts:

Area	Classes	Lines	Co-operative aspect	Lines	Realised in part
Main program, user interface	5	1568	Basic sharing functions, initialisation	297	Main program
Annotations	3	685	Telepointers	148	page processing and graphical output
HTTP and HTML parsing	5	2075	following links and page sections	71	page processing and graphical output
page processing and graphical output	5	2914	distributed data processing of annotations	31	Main program

Table 1. Development costs

The right table shows the amount of code related to collaboration aspects.

As a result,

7,6% of the source code is used to realise co-operative functions;

92,4% of the source code is used to realise functions of a single user browser.

This result 'proves' that the uncompromising usage of the DreamTeam platform allows a developer of shared applications to concentrate on the application itself rather than on re-implementing co-operative services.

5 Related Work

This section relates DreamTeam to other synchronous co-operative platforms.

The *Groupkit* system [Roseman et al. 96] is a package for implementing shared applications under TCL-TK. A library offers services for session management, communication and shared dialogue management. It is mostly based upon a de-centralised architecture, only for the group rendezvous a central server is needed. A co-operative Web server called *Groupweb* is built on the Groupkit platform. In addition to telepointers and co-operative browsing, it allows the annotation of Web pages. In contrast to DreamView, a page can only be annotated as a whole. Annotations cannot be anchored to page items.

Habanero [NCSAa] has completely been implemented in Java. Its architecture is centralised, i.e. requires a Habanero server application in order to enable group activities. A co-operative Web browser is available, called *WWW shared session* [NCSAb]. Actually, this browser is the Mosaic browser, which is controlled via a data channel. Thus, group specific services (e.g. telepointers) are not available.

The *Rendezvous* system [Hill et al. 93] provides a distribution mechanism based upon X-windows events. This approach is completely different to our approach, since its emphasis lies on distributing windows contents and user events via the X-protocol. It is difficult to apply this approach to non-X-windows environments.

Share-Kit [Edlich 93] is a Unix-based platform which provides multicast RPC for C programs. Neither session management nor group specific widgets are included in this platform.

Dolphin [Streiz et al. 94] is a co-operative hypermedia system. The emphasis lies on the co-operative editing of hypermedia documents. It is written in Smalltalk and provides one hardcoded application, a shared hypermedia editor.

Compared to other platforms, DreamTeam offers a straight forward implementation of shared application and encourages rapid prototyping. A simulation environment helps to find out weak points and ensures stability before delivering applications to the end users. The de-centralised structure of DreamTeam avoids bottleneck problems with central servers.

6 Conclusion and Future Work

DreamTeam is a platform for synchronous co-operative applications. It consists of a runtime environment, a development environment and a simulation environment. The platform is designed to minimise the development cycle for co-operative applications and to encourage rapid prototyping. The implementation of the co-operative Web browser DreamView indicates the validity of our approach. Only small code segments had to be added to the code for a single user browser to make the application available in a co-operative environment.

The development of the DreamTeam environment is an ongoing process. Each new kind of shared applications requires new services in the service or base layer. In order to define a 'complete' set of services, we are planning to realise a variety of different co-operative applications. In addition to the co-operative Web browser, a chat tool, a drawing tool, a two player game and a co-operative, object oriented graphical editor have been implemented so far. Presently, a sophisticated, co-operative text editor is being implemented.

In order to reduce the time for implementing new co-operative applications, we are working on the definition of a high-level programming language for co-operative application, which supports the development of fully collaboration transparent applications with a minimum of extra source code.

7 References

[Dommel et al. 97] Dommel, H.-P., & Garcia-Luna-Aceves, J.J. (1997), Floor control for multimedia conferencing and collaboration, *Multimedia Systems*, Vol. 5, 1997, 23-38

[Edlich 93] Edlich S. (1993), Software Cooperation with the Share-Kit: Influences of Semantic Levels on the Working Efficiency, *Vienna Conference on Human Computer Interaction VHCI '93*, Vienna, Austria, Sept. 20-22, 1993, 225-234

[Hill et al. 93] Hill R.D., & Brinck T., & Patterson J. F., & Rohall S. L., & Wilner W. T. (1993), *Rendezvous Language*, *Communications of the ACM*, Vol. 36, No. 1, Jan. 1993, 62-67

[NCSAa] NCSA Habanero Homepage, <http://www.ncsa.uiuc.edu/SDG/Software/Habanero/HabaneroHome.html>

[NCSAb] WWW Shared Session, <http://www.ncsa.uiuc.edu/SDG/Software/Habanero/Tools/WWWShare/index.html>

[Roseman et al. 96] Roseman, M., & Greenberg, S. (1996), Building Real-Time Groupware with GroupKit, A Groupware Toolkit, *ACM Transactions on Computer-Human Interaction*, Vol. 3, No. 1, Mar. 1996, 66-106

[Roth 97] Roth J. (1997), „DreamTeam“ - eine synchrone CSCW-Plattform für heterogene Umgebungen, *Internal Report 218*, Fernuniversität Hagen, Mai 1997

[Roth 98] Roth J. (1998), How to write shared applications with „DreamTeam“, *Internal Technical Report*, Fernuniversität Hagen, Jan. 1998

[Streiz et al. 94] Streitz, N.A., & Geißler, J., & Haake, J.M., & Hol J. (1994), DOLPHIN: Integrated Meeting Support across LiveBoards, Local and Remote Desktop Environments, *Proceedings of the ACM '94 Conference on Computer Supported Cooperative Work (CSCW '94)*, Chapel Hill, North Carolina, Oct. 22-26, 1994, 345-358