

CSCW Internet Tools and Environments for Distance Education

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

Abstract: This paper describes collaborative Internet environments for distance education. The Internet offers various possibilities for asynchronous as well as synchronous collaboration. This paper focuses on synchronous platforms which range from systems for shared applications to video conference systems. Beside platforms which are designed for research purposes (*Habanero*, *Groupkit* and *Share-Kit*), two commercial products are examined: *Netscape Conference* and *CU-SeeMe*. In addition to these platforms, a development of the University of Hagen is introduced: *DreamTeam*. 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. It is planned to use the DreamTeam environment for software courses and electronic lectures in the Virtual University project.

The DreamTeam project is heavily supported by the project VirtUE - a Virtual University for Europe, TEN-TELECOM contract 45493 .*

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. Various platforms exist nowadays, both research platforms as well as commercial products. This paper describes a collection of platforms, each of it focusing on special problem areas. The criteria to assess such platform are manifold. We identify the following main topics:

1.1 Service domain

There are different variations of synchronous collaboration. The main topics are *shared applications* and *computer mediated communication*.

Shared applications are applications which run in a collaborative environment and have to consider both, events from a local user as well as events from other users or systems. To develop such a co-operative application is a difficult and time-consuming task. Dommel et al. [DG97] distinguish three types of co-operative applications:

* The DreamTeam platform as well as the documentation and a slide show can be accessed under http://carmen.fernuni-hagen.de/dreamteam/dreamteam_eng.html

- *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.

Using a platform which is designed according to 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 (see 1.2).

Systems for computer mediated communication enable audio or video conferences. Such systems have to handle different audio and video hardware, establish real-time network connections and have to use compressing/decompressing algorithms for audio and video data. Since most of these problems are more of technical nature, most of the systems are commercial products rather than research systems. Normally, video hardware is very expensive, thus not suitable for student systems, but recently low-cost cameras with mediocre video quality are available. For sound channels, the audio hardware of common PCs are adequate.

1.2 Development support

In most cases a system has to be extended in order to fit to the special task. Some systems offer an application programming interface (API) to develop own software. Normally an API description is only available for non-commercial systems.

Some systems are specially designed to encourage own developments and are delivered with programming libraries which contain building blocks for basic problems as well as group specific elements.

Normally, it is required to use a special programming language for a certain platform. The influence of the programming language should not be underestimated. Using a modern language (e.g. Java) can solve many problems which must be programmed explicitly in older languages, e.g. object serialization, exception handling, synchronisation monitors etc. From the viewpoint of platform independence, different levels can be identified:

- Platform independent syntax:
A program can be translated on several systems if a compiler is available. Examples are the programming languages C and C++, where syntax is standardised. Programs for different platforms must not use proprietary libraries e.g. for building user interfaces.
- Compatibility on binary level:
To achieve compatibility, it is necessary to have an interpreter or virtual machine on every system as well as a programming library version for every platform. Programming languages of this kind are script languages (*TCL-TK*, *Perl*) or languages which use a virtual processor, e.g. *Java* [SUNa].

The compatibility on binary level has many advantages. A program can be executed on several platforms without re-compilation. Most of our students are using PCs with a variety of different operating systems, so a platform independent development is extremely important.

1.3 Organisation

Co-operative platforms 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 are often 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. A simple variation of the de-centralised architecture is a point-to-point architecture. If only two systems are connected, many problems are avoided, but a limitation on two participants means a hard restriction.

2 Collaborative Internet platforms

In the following, a collection of platforms for synchronous collaboration is examined.

2.1 Habanero

Habanero [NCSAa] is a platform for synchronous collaboration. Habanero focuses on making Java applets available in a distributed environment. The applets must be available as source code and can in most cases be converted into a distributed applet (called *Hablet*) without too much effort.

Habanero is entirely written in Java, thus executable on many systems. Unfortunately, some shared demo applications contain native code.

The architecture of Habanero is centralised. A server application has to run on a well-known server in order to enable collaboration. Once the Habanero server is started, a user can start the client application. The next figure shows a typical Habanero desktop:

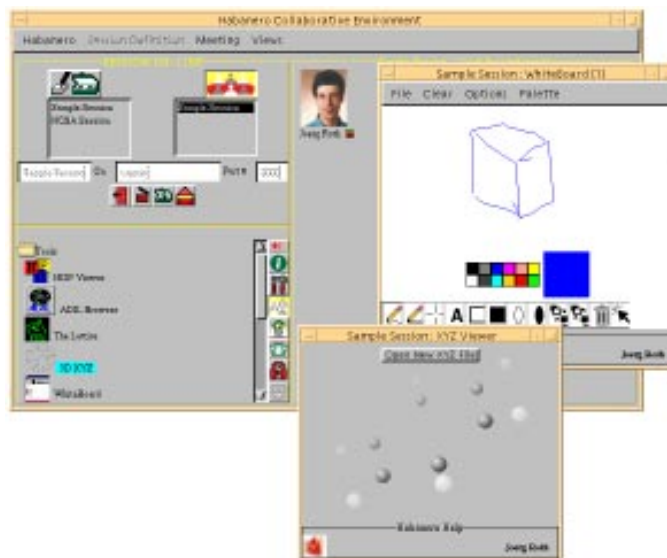


Figure 1: A Habanero desktop

In the client environment a participant can create a user profile entry including a picture. The basic idea of Habanero is to broadcast all user actions to every participant's system. An existing applet can be converted to a Habanero shared application by making a few source code changes. This process is called "Habanerizing a Java Applet" [NCSAb].

Habanero offers two event handling concepts:

- *Simple Event Handling*: all user events are made public without any restriction
- *Advanced Event Handling*: Some events can be handled by the local application and will not be made public to others (*private events*)

With the last mechanism it is possible to limit the distribution as necessary, e.g., if a users draws a line in a shared whiteboard, it is not necessary to publish an event for every single mouse step.

All user events which in Java cause a call of the applet's *action* method, are redirected to a distributing mechanism. Unfortunately this concept is restricted to older Java versions, since the Java event concept has fundamentally changed with version 1.1. Newer Java applets have several event handling methods, thus the generic distributing mechanism of Habanero fails. It is not clear, if the Habanero environment or Habanero Hablets can be migrated to the new event model standards.

Since Java's object serialization concept [SUNb] for the first time is available in version 1.1, it cannot be used in Habanero. With object serialization, it is possible to send an arbitrary object through the network and to rebuild it on another system. Since this is unavailable in Habanero, event data are restricted to *string* type.

Habanero offers a set of predefined shared applications, e.g. a shared whiteboard, a chat tool, a tool to examine molecule structure. For collaboratively browsing the web, the Mosaic browser is integrated into the environment [NCSAc]. A real-time audio tool enables audio conferencing. Since Java has no audio library until now, this tool contains native code and is not available on all operating system platforms.

2.2 Groupkit

Groupkit ([GK], [RG96]) is a development package for distributed application. It is based on TCL-TK. A program library contains basic services for standard problems, covering *session management*, *communication* and *distributed user interfaces*. Group and user profiles are stored and can be accessed from an application.

The next figure shows a Groupkit desktop.

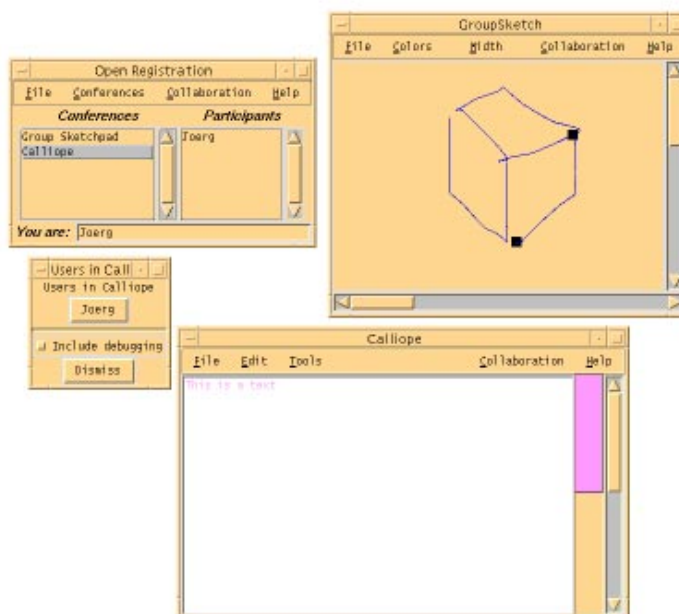


Figure 2: A Groupkit desktop

The organisation of Groupkit is in main parts de-centralised. Only for the contact phase, a centralised *registrar* is involved. A registrar is a well-known server which holds and make

public network addresses of potential session members. When a session is established, it continues in a de-centralised way without the registrar.

Groupkit offers group specific dialog elements [GRG96], e.g.

- telepointer
- miniature view: shows a document with reduced scale in another window
- radar view: same as miniature view, the views of other users are represented as small rectangles and other mouse pointers are visible
- multi-user scrollbar: a scrollbar which represents the view settings of all participants
- what you see is what I do (wysiwid): shows the nearer area around the pointers of other users.

An application developer can simply integrate this elements in a shared application without concerning communication aspects between users.

The Groupkit environment contains several demo applications e.g. multi-user text editors, sketch tools and brainstorming tools. For collaborative web browsing, a shared web browser is available, called *Groupweb* [GR96].

2.3 Share-Kit

Share-Kit ([TUB], [Jah95], [Edl93]) is a Unix based platform for the development of shared applications. The programming language for Share-Kit is C. The organisation of Share-Kit is centralised, i.e. a server application must run on a well-known server.

A Share-Kit application runs without a designated front-end. A user simply starts an application from a command line and passes the server address via command line parameter. Share-Kit includes neither session nor user management.

The following figure shows a Share-Kit sample application.

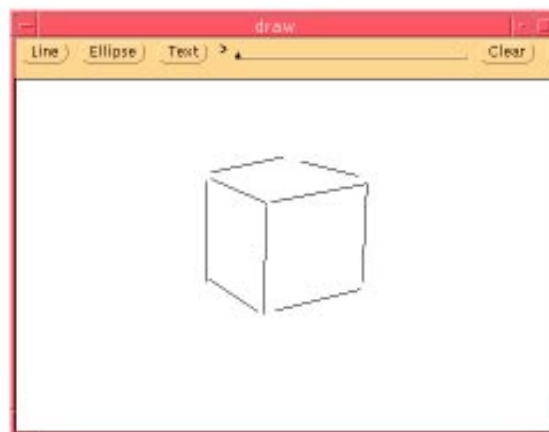


Figure 3: The Share-Kit sample application "Draw"

Share-Kit publishes user events via Multicast-RPC. Multicast procedure stubs are generated by a C pre-processor.

2.4 Netscape Conference

Netscape Conference is part of the *Netscape Communicator* package. The Communicator essentially offers services for web browsing (*Netscape Navigator*). In addition, services like mail reading, newsgroups etc. are provided. The conference tool is included in the Communicator package to enable synchronous conferences.

Netscape Conference contains:

- a tool for audio conferencing
- a shared whiteboard
- a chat tool

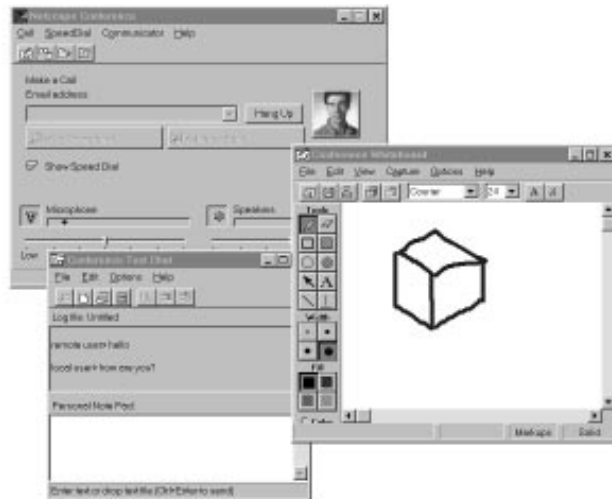


Figure 4: Netscape Conference

With help of the Navigator, shared WWW sessions are possible. It is not intended to integrate own shared applications nor is any programming library available.

2.5 CU-SeeMe



Figure 5: CU-SeeMe

The main task of CU-SeeMe [CU] is to enable video conferences. Only low-cost equipment is required (e.g. a black-and-white camera which is connected via printer port to the PC). For the audio channel, a simple sound card with microphone and speakers is sufficient.

For conferences with two members, only CU-SeeMe clients are required. For conferences with more participants, a *reflector* is necessary. A reflector is a server application which receives video- and audio data and broadcasts it to all participants.

CU-SeeMe contains in addition to the audio and video channel a shared whiteboard and a textual chat tool. It is not possible to extend this set by own developments.

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

Organisation

As described earlier, a centralised organisation leads to simpler program architectures, but group servers are often performance bottlenecks. 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.

Communication channels

On request, DreamTeam supports a variety of communication channels:

Connection channels are the fastest channels available; they are used for real time communication with minimal delay. They are realised via permanent sockets, which are opened when a session starts or when a user joins a session. They stay open until the session finishes or the user leaves a session. Connection channels are handled by highest priority tasks.

Session channels are handled by medium priority tasks; they are used for the distribution of session profiles, session status changes, as well as for login and logout protocols. Session channels are handled in the background, they are opened and closed on request for single transactions.

Transfer channels are operated in the background with the lowest priority. They are, for example, used for file transfer and, like session channels, are opened and closed on request. In the future it will be necessary to extend this spectrum of channels by very fast channels for video and audio transfer.

Group rendezvous

A de-centralised group rendezvous must work without a „well-known server“. If no central server keeps the group state, it is difficult for a new participant to reach his group. The problem becomes even more complex, if Internet addresses change between dial-ins. To

solve this problem, we divide all group members into two categories, the first category containing all participants with fixed network addresses (e.g. the teaching staff), the second one containing the members with variable addresses (e.g. students). Members of the second category are represented by sets of potential addresses. The address or the set of addresses is stored in the user profile. When a participant wants to join a group, his system first has to build a list of all other systems currently online. For this, the rendezvous component queries all known systems beginning with the fixed addresses. A query is successful, if another system is already online (i.e. the network access has been established and DreamTeam has been started). This other system has already built an own address list which it now transfers to the newcomer. The newcomer can now ask the other systems to update their local address lists. This procedure ensures that, after a certain delay, all systems know the state and address of all other systems in their group. Even in the case of variable addresses, a broadcast is avoided if at least one participant with a fixed address is online. In the worst case (all group members have variable addresses), a broadcast is necessary. Nevertheless, the network load for this broadcast is acceptable, because only addresses stored in the user profile have to be tested.

3.2 The development environment

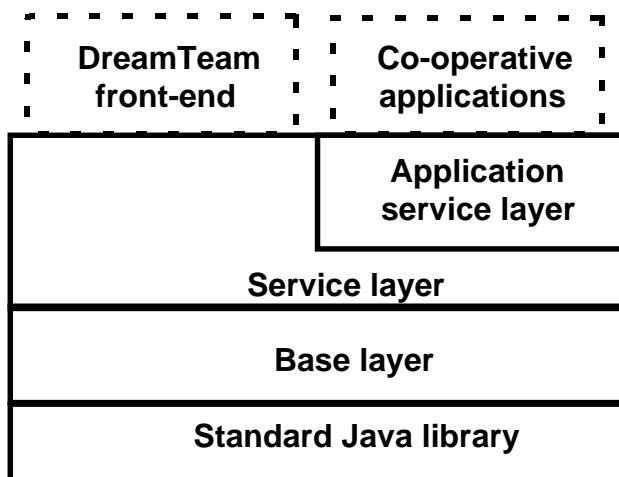


Figure 6: Layers of the class library

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 (figure 6).

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 [Roth97a] and [Roth97b]. Using this class library, the DreamTeam front-end (see 3.3) as well as several co-operative applications have been developed so far, a co-operative web browser called *DreamView* is described in more detail in [Roth97c].

3.3 The runtime environment

Besides providing services to shared applications, the runtime environment offers a set of functions to the end user. The following figure shows a typical DreamTeam desktop.

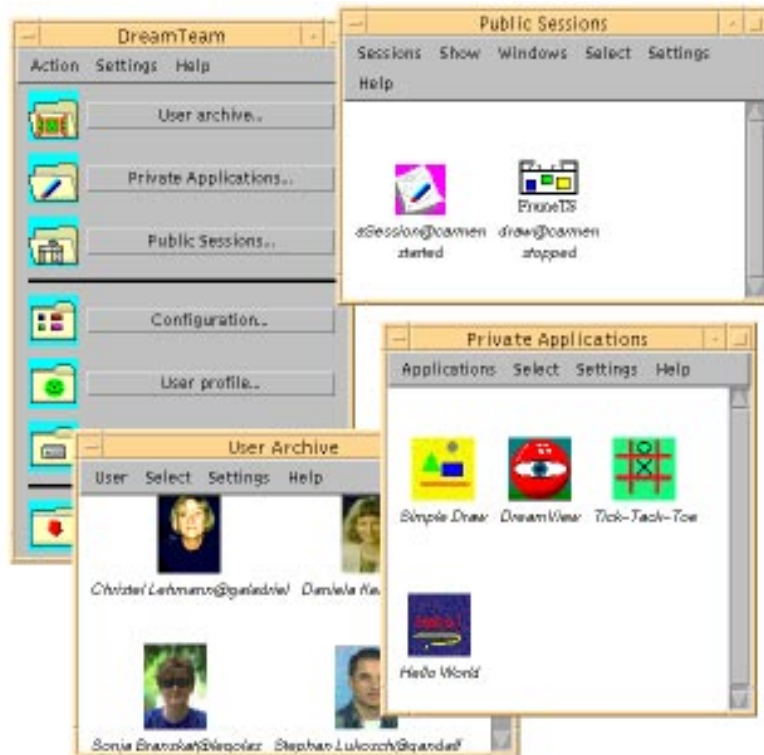


Figure 7: The DreamTeam working environment

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.

3.5 Future developments

The current service domain of DreamTeam is to enable shared applications. For direct communication between session members, at least an audio channel is necessary. At the moment, a separate tool must be started to enable audio conferences. From the viewpoint of end-user, it is more suitable to have audio capabilities integrated in the DreamTeam environment. For this, it is planned to implement audio services as far as the underlying programming language (Java) gives access to the audio hardware.

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

4 Comparison

4.1 Service domain

	Habanero	Groupkit	Share-Kit	Netscape Conference	CU-SeeMe	DreamTeam
Shared Applications	yes	yes	yes	only chat and whiteboard	only chat and whiteboard	yes
Collaborative Web browsing	Mosaic	Groupweb	no	Navigator	no	DreamView
Audio conferences	yes	no	no	yes	yes	no
Video conferences	no	no	no	no	yes	no

4.2 Development support

	Habanero	Groupkit	Share-Kit	Netscape Conference	CU-SeeMe	DreamTeam
Own developments provided	yes	yes	yes	no	no	yes
development environment	yes	yes	yes	no	no	yes
programming language	Java (only 1.02)	TCL-TK	C	-	-	Java
System platforms	platform independent	Unix, Windows	Unix	Unix, Windows	Unix, Windows	platform independent
simulation environment	no	no	no	no	no	yes

4.3 Organisation

	Habanero	Groupkit	Share-Kit	Netscape Conference	CU-SeeMe	DreamTeam
During rendezvous	centralised	centralised	centralised	point-to-point or centralised	point-to-point or centralised	de-centralised
During session	centralised	de-centralised	centralised	point-to-point	point-to-point or centralised	de-centralised

5 References

- [CU] *Enhanced CU-SeeMe Home Page*
<http://www.cu-seeme.com>
- [DG97] Dommel H.-P., Garcia-Luna-Aceves J. J.,
Floor control for multimedia conferencing and collaboration,
Multimedia Systems, Vol. 5, 1997, 23-38
- [Edl93] Edlich S.,
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
- [GK] *Groupkit Home Page*
<http://www.cpsc.ucalgary.ca/projects/grouplab/groupkit/>
- [GR96] Greenberg S., Roseman M.,
GroupWeb: A WWW Browser as Real Time Groupware,
Proceedings of the ACM Conference on Human Factors in Computing Systems, ACM Press, Apr. 1996, 271-272
- [GRG96] Gutwin C., Roseman M., Greenberg S.,
A Usability Study of Awareness Widgets in a Shared Workspace Groupware System,
Proceedings of the ACM Conference on Computer Supported Cooperative Work, ACM Press, Nov. 1996, 258-267
- [Jah95] Jahn P.,
Getting started with Share-Kit,
<ftp://ftp.cs.tu-berlin.de/pub/local/kbs/share-kit/share-kit.tar.gz>, 1995
- [NCSAa] *NCSA Habanero Homepage*
<http://www.ncsa.uiuc.edu/SDG/Software/Habanero/HabaneroHome.html>
- [NCSAb] Ian C.,
Peppery Patterns - How to turn Java applets into Habanero hablets
<http://www.ncsa.uiuc.edu/SDG/Software/Habanero/API/Tutorial/pepperyPatterns/>
- [NCSAc] *WWW Shared Session*
<http://www.ncsa.uiuc.edu/SDG/Software/Habanero/Tools/WWWShare/index.html>
- [RG96] Roseman M., Greenberg S.,
Building Real-Time Groupware with GroupKit, A Groupware Toolkit,
ACM Transactions on Computer-Human Interaction, Vol. 3, No. 1, Mar. 1996, 66-106
- [Roth97a] Roth J.,
„DreamTeam“ - eine synchrone CSCW-Plattform für heterogene Umgebungen,
Informatik Bericht 218, Fernuniversität Hagen, May 1997
- [Roth97b] Roth J.,
How to write shared applications with „DreamTeam“,
Internal technical Reference, Fernuniversität Hagen, Dec. 1997
- [Roth97c] Roth J.,
„DreamView“ - ein kooperativer Webbrowser für eine synchrone CSCW-Umgebung,
Informatik Bericht 218, Fernuniversität Hagen, Nov. 1997
- [SUNa] *JavaSoft Home Page*
<http://java.sun.com>

- [SUNb] *Object Serialization*
<http://java.sun.com/products/jdk/1.1/docs/guide/serialization/index.html>
- [TUB] *Share-Kit Home Page*
<http://www.cs.tu-berlin.de/cs/research/english/projects/share-kit.html>