

Developing Synchronous Groupware for Learning Environments

Stephan Lukosch and Jörg Roth

University of Hagen, Department of Computer Science, 58084 Hagen, Germany
{Stephan.Lukosch, Joerg.Roth}@Fernuni-Hagen.de

Abstract Synchronous groupware brings together users, which are geographically distributed, but connected via a network. Usually, groupware is not developed from-the-scratch, as the development is difficult, time-consuming, and error-prone. An application's data must be shared to support interactions across distributed, collaborating users. Using a platform, which offers solutions for the recurring problems, the development is simplified. In this paper we describe DreamTeam and its extension DreamObjects. They allow the developer to develop co-operative applications like single user applications, without struggling with network details, synchronisation algorithms, etc. As an example we present the development of a distance learning environment, which has been developed using these platforms.

1 Introduction

Groupware systems are computer-based systems that support two or more users engaged in a common task, and that provide an interface to a shared environment [EGR91]. In this paper we focus on the development of synchronous groupware, which allows users at different places to interact at the same time.

When developing synchronous groupware for education scenarios, one has to take into account various students' needs and requirements:

- *User interfaces:* 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 applications should make as few assumptions as possible about the system platform with regard to hardware and software. 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 session.
- *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 disconnection. 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 [RU99] to find out which students are actually available for a meeting and how they actually can be reached via the Internet.

When developing groupware applications respecting these requirements, we noticed that shared data and maintaining data consistency are among the main obstacles for groupware development. In our opinion, groupware development should be almost as easy as the development of single-user applications. This idea lead to the development of *DreamObjects* [LU00] in coherence with *DreamTeam* [Roth00].

After a description of DreamTeam and DreamObjects we present a learning environment, which has been developed using these platforms.

2 DreamTeam

DreamTeam is a platform for synchronous collaboration, which offers predefined solutions for application developers as well as end-users. It consists of a development environment, a runtime environment and a simulation environment. The development environment mainly consists of a huge Java class library which contains groupware specific problem solutions as building blocks. The runtime environment provides an infrastructure with special groupware facilities. A front-end on top of the runtime environment allows end-users to control and configure the system. Finally, collaborative applications can be tested in the simulation environment, which allows to simulate network characteristics on a single computer.

A common paradigm for shared workspaces is WYSIWIS (What You See Is What I See) [SBF+87]: the strict WYSIWIS policy enforces identical workspace displays on every participating system, whereas a relaxed WYSIWIS policy allows private areas and tolerates slight layout differences. Since DreamTeam is designed for heterogeneous environments, it supports relaxed WYSIWIS.

For an end-user it may be meaningful to use a shared application in a private environment as well, without having to learn a new user interface. With minor additional effort, DreamTeam applications can be developed for both, shared as well as private environments. An application can ‘ask’ the environment in which mode it is currently running and can enable or disable specific functions.

To develop groupware applications in a more component-based manner, DreamTeam comes along with a component concept called *TeamComponents* [RU00b]. TeamComponents allow to efficiently develop synchronous groupware applications with the help of software components, each of which is responsible for a specific artefact, offers its own user interface and manages its own internal data. Using components helps to divide a software project into well defined parts. Well documented interfaces help to reduce integration efforts and improve software quality. TeamComponents can be developed separately from an application, thus the component developer and the application developer can be different people. A set of classes and interfaces guarantees the proper integration of components into an application, even if the corresponding component source codes are not available.

3 DreamObjects

DreamObjects simplifies the development of shared groupware objects and offers services to handle them at runtime. These services include, e.g., a flexible distribution of the shared objects and a configurable concurrency control scheme. For this we replace shared objects with what we call *substitutes*. This name stems from the substitution principle of object-oriented languages. Due to this principle substitutes offer the same interface as the shared object they are used for, and thus can easily be used as a placeholder. Substitutes are either generated automatically at runtime or by the developer from the command line. Inside a groupware application the developer uses the substitute in the same way as a local object.

A substitute generalises a shared object and overwrites some of the shared object’s methods to add additional functionality. To share data, a substitute distributes method calls and encodes the corresponding arguments. Moreover, a substitute stores additional object information, e.g. the used concurrency control scheme or the distribution mode.

Since the discussion about the best distribution mode for groupware is still going on [RU00a], our toolkit supports replicated as well as central objects. We favour the use of replicated objects, when high responsiveness is needed, e.g. in group editors. Central objects fit well,

when the object itself contains a significant amount of data, e.g. a video, and only small parts of the object must be transmitted to users, e.g. single frames.

To develop a replicated object the developer has to extend a basic replicated object class and for a central object the developer has to extend a basic central object class. To access a shared object at runtime it has to be registered in DreamObjects's object manager. For this the developer must provide the shared object's classname and a unique registration name. The object manager creates an instance of the shared object's substitute class and adds this instance to its object registry. After this, the object manager informs all other object managers about the newly registered object and returns the substitute. If the newly registered object is replicated, the object manager distributes a replica of the shared object. Otherwise only its reference is distributed.

There are no limitations to the complexity of a shared object. A shared object can include references to other shared objects, which may be necessary in complex collaborative application. In contrast to other approaches in object-oriented distributed programming [GF99] new distribution modes or techniques for handling shared data can easily be included in the toolkit. To include a new distribution mode, a new basic shared object class is simply added to the object manager's class library.

4 An Example Collaborative Learning Environment

Figure 1 shows a screenshot of a collaborative learning environment.

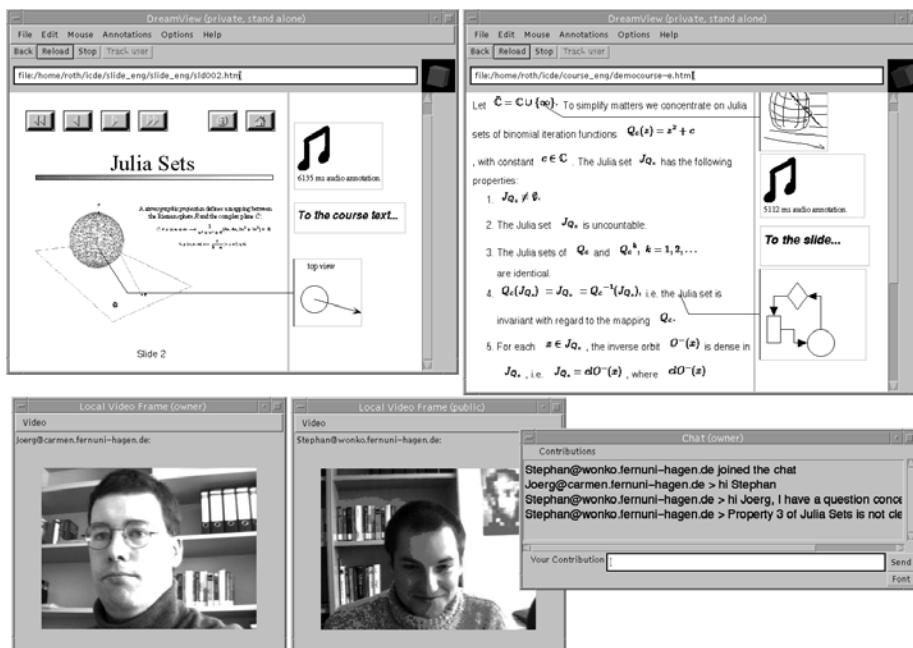


Figure 1: Collaborative Learning Environment

Each participant starts a DreamTeam session, which may contain the following collaborative components:

- DreamView (a collaborative Browser and Annotator),
- DreamChat,
- Portholes, and
- DreamAudio conference.

Depending on their connection bandwidth and their hardware equipment students may have to restrict their individual environments. As minimal support for a session, each student has to run the DreamView browser and the DreamChat tool. If there is enough bandwidth available, Portholes of all participants as well as a moderated DreamAudio conference can be added to individual environments. Since Java so far does not sufficiently support video applications, Portholes are not fully integrated in the DreamTeam environments: at each site, a camera regularly takes a picture of the participant and stores it to a local file, which is then distributed to all session participants via DreamTeam.

During a session, all participants can collaboratively work on a hyper-lecture [LRU99], can add annotations and use their individual, shared pointing devices (mouse pointers). One participant can dynamically take the role of a tutor or lecturer, run a slide show and control the audio conference. Other participants can join in via annotations as well as chat and audio contributions.

5 Future Work

Using DreamTeam and DreamObjects developing learning environments is almost as easy as developing a single-user application. However, there are still things which have to be improved. E.g., supporting the transition of work between varying degrees of asynchronous and synchronous collaboration during learning will allow students and also lectures to prepare themselves in better way.

6 References

- [EGR91] C.A. Ellis, S.J. Gibbs, and G.L. Rein: Groupware some issues and experiences, in *Communications of the ACM*, 34(1):38–58, January 1991.
- [GF99] R. Guerraoui and Mohamed E. Fayad: OO Distributed Programming Is Not Distributed OO Programming, in *Communications of the ACM*, 42(4):101–104, April 1999.
- [LRU99] S. Lukosch, J. Roth and C. Unger: Marrying on-campus teaching to distance teaching, in: *Proceedings of the 19th world conference on open learning and distance education*, Vienna, Austria, 20.-24. June 1999.
- [LU00] S. Lukosch and C. Unger: Flexible Management of Shared Groupware Objects, in *Proceedings of the Second International Network Conference (INC 2000)*, Plymouth, UK, 3.-6. July 2000, pp. 209-219.
- [Roth00] J. Roth: 'DreamTeam': A Platform for Synchronous Collaborative Applications, *AI & Society*, Vol. 14, No. 1, Springer Verlag London, March 2000, pp. 98-119.
- [RU99] J. Roth and C. Unger: Group Rendezvous in a Synchronous, Collaborative Environment, in R. Steinmetz (ed): *Kommunikation in Verteilten Systemen (KiVS'99)*, 11. ITG/VDE Fachtagung, Springer, 2.-5. March 1999, pp. 114-127.
- [RU00a] J. Roth and C. Unger: An extensible classification model for distribution architectures of synchronous groupware, in Dieng R. et al. (eds): *Fourth International Conference on the Design of Cooperative Systems*, Sophia Antipolis (France), 23.-26. May 2000, IOS Press, pp. 113-127
- [RU00b] J. Roth and C. Unger: Developing synchronous collaborative applications with TeamComponents, in Dieng R. et al. (eds): *Fourth International Conference on the Design of Cooperative Systems*, Sophia Antipolis (Frankreich), 23.-26. May 2000, IOS Press, 353-368
- [SBF+87] M. Stefik, D.G. Bobrow, G. Foster, S. Lanning, and D. Tatar: WYSIWIS revised: early experiences with multiuser interfaces, in *ACM Transactions on Office Information Systems*, Vol. 5, No. 2, Apr. 1987, 147-167