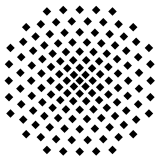

Universität Stuttgart
Fakultät Informatik

# Application sharing in teaching context with wireless networks

**Authors:**
Dr. C. Burger
Dipl.-Inf. S. Papakosta
Prof. Dr. K. Rothermel

**CR-Classification:**
K.3
H.4.1

**Keywords:**
Application sharing
Teaching
Context awareness

**Application sharing in teaching context with wireless networks**

*C. Burger, S. Papakosta, K. Rothermel*

**Technical Report 2001/07**
December 2001

Institute of Parallel and Distributed High-Performance Systems (IPVR)
Department of Computer Science University of Stuttgart
Breitwiesenstr. 20 - 22
D-70565 Stuttgart
Germany

# Application sharing in teaching context with wireless networks

Cora Burger, Stella Papakosta and Kurt Rothermel

Institute of Parallel and Distributed
High-Performance Systems (IPVR)
University of Stuttgart
Breitwiesenstr. 20-22
70565 Stuttgart, Germany
{caburger, papakosi, rothermel}@informatik.uni-stuttgart.de

Technical Report 2001/07
Department of Computer Science
University of Stuttgart
October 2001

**Abstract**

The success of teaching is depending on a couple of factors: on how far students are involved into lectures, on the material, its completeness and on co-learning of students. Involvement of students into lectures means, being able to follow the thoughts of the teacher, ask questions and make comments. The material must be presented in a suitable form and essential parts of it have to be available during the whole learning process, for preparing participation in lectures and exercises as well as for exams. For more effective learning and training of social abilities, working in groups of co-learners has to be encouraged.

Mobile and ubiquitous computing offer new possibilities to achieve these goals by increasing the awareness in class and supporting an active participation of students. By promoting existing concepts and enabling new ways of application sharing, the project SASCIA (System architecture supporting cooperative and interactive applications) aims at developing a framework for multiple applications to support teaching in collocated, remote and hybrid scenarios. Its core is composed of components to capture and distribute context information about sessions, participants and those applications that are used during a lecture or encounter among students. A configurable floor control was designed to cope with a wide spectrum of applications and learning situations. For some cases, even a control for semantic consistency can be necessary. In combination with a suitable user and session management, a whiteboard for annotations and a recording facility to support latecomers as well as subsequent replay, these components are providing the required functionality.

As a consequence, SASCIA offers remote control and viewing facilities to all participants during lectures and co-learning sessions. That means, usage of a lot of facilities like applications, pointing and annotating in application windows are available to everyone by leaving teachers the right to interrupt at any time. All public annotations as well as private ones can be stored and replayed afterwards. By offering state information about everybody in class with proper regard of privacy concerns, participants are enabled to address each other more personally. This holds especially for teachers getting direct feedback on their presentation. For discussions without a teacher being present, SASCIA provides collaborative ways of experiencing teachware and a voting mechanism to support ad hoc encounters and peer relations among participants.

As demonstrated by measurement values with a first prototype, the performance of the whole system is acceptable. Further evaluation and experiments in real class situations are planned for the next semesters.

## 1 Introduction

Education has become one of the most crucial factors of our community regarding industry as well as all other areas of daily life. To achieve high quality in teaching, students have to be inspired to actively deal with a topic. Hence, suitable facilities are needed and an appropriate atmosphere has to be created to let students experience the material by themselves, discuss and test different aspects, ask questions and make comments. In short, a constructive rather than an instructive manner of teaching is the preferred one. Moreover, learning in groups has to be encouraged to increase its effectiveness and to train social abilities.

A first support for collocated scenarios with all participants being present in the same room was given with desktop computers connected by a fixed network and equipped with a conference system for application sharing. Because of the enourmous costs and the inflexibility of this approach, it was realized in special environments and for small groups of

learners only (cf. [1]). By transmitting audio and video from lectures to distributed groups of students, remote scenarios are supported (e. g. [11]; [41]). This is useful for instance in sparsely populated regions, for handicapped people or for life long learning. In short, those people are gaining profit from tele-lectures who prefer staying in a certain environment due to problems with traveling or just to save time and cost.

But tele-presence is not well accepted due to its lack from mediating the feeling of really being together. Moreover, tele-lectures have proofed rather poor with regard to interactivity and students becoming actively involved [26]. Even new mechanisms and policies for floor control to achieve more flexibility in gaining access rights to microphone and keyboard, did not change this situation tremendously (cf. [42]). Nevertheless, tele-lectures have triggered the recording of audio and video in combination with slides, a mechanism that is used successfully to make available material to learners [12], [57]. By adding further facilities, even annotations as made on the slides by the teacher during the lecture can be captured and offered afterwards (e. g. [1], [6], [9], [27], [60]).

In contrast to human interaction, there is a much higher motivation to perform laboratory experiments from remote. While it can provoke the most intensive experience for learners, to be close to an experiment and really watch the whole process, there are some serious reasons against it. Usually, equipment for experiments is quite expensive but nevertheless rather seldom in use. Hence, the possibility of sharing the same equipment in the form of tele-experiments among different, geographically separated institutions, increases the selection of available experiments for teachers and students. Furthermore, it is not always feasible and can even bear risks to be near, e. g. in chemical or nuclear plants. As a consequence, some monolithic solutions have been designed already to enable tele-experiments in a special teaching area for single users as well as for collaboration (e. g. [25], [17]).

With more wide spread usage of mobile devices and the upcoming technology of wireless networks, new possibilities are arising for collocated and remote ways of teaching. Currently, a lot of work is underway to bring wireless equipment to the campus and find out its potential for teaching. A first step is provided by making available services like electronic mail, access to information and course material as well as notification throughout the whole campus [20]. More sophisticated features are offered by the following work:

- [66] let each student experiment on its own with tools like compilers during the lecture.
- [9] enable teacher and learners to make public and private notes for later retrieval.
- [2] provide a facility for remote pointing in application windows for a scenario among an expert and somebody working in the field. This possibility can be applied to teaching situations as well as shown by [48].
- [56] and [33] examine further possibilities like the electronic pendant of hand raising and direct feedback to teachers via handheld computers during the lecture.

From a more general point of view, the last two cases have in common that state information about applications and people as well as its visibility to participants are involved. In literature, this fact is called *context awareness* (e. g. [24] and [39]): Either an individual is informed about its own state and its environment or other people are getting such information. This holds for the electronic as well as for the physical world. For instance, people can be informed about activities of other persons in an electronic shared workspace where members are allowed to bring in or modify folders and files for shared access [4]. Even in the physical case, more and more types of context information are becoming available like e. g. position, temperature, and intensity of light or noise. To this end, either cameras or suitable sensors can be used (e. g. [25], [59]).

We want to proceed one step further by studying all these aspects in a systematic way. Context awareness will be combined with all other facilities into one framework that allows to share and control typical applications among teacher and students or a group of students during the same session in the most powerful and flexible way. Such typical applications in the area of teaching include

- tools for presentation (e. g. Microsoft PowerPoint, StarOffice or Acrobat Reader),
- engineering or simulation environments (e. g. products like JBuilder, Rational Rose, Ptolemy, Matlab) or even
- real experiments like e. g. remote control of input to three tanks or of a railway model [8].

In the following, we start with collecting requirements arising in this special context (section 2). At first sight, the well known concept of conferencing and sharing tools with standardized protocols ([28], [31]) should suffice to solve our requirements. But by evaluating some existing systems against our criteria, section 3 demonstrates that they have not been designed to meet teaching requirements completely. Hence, the need for additional and specific features came up.

As a consequence, a model was built for all involved components and their context information being relevant for application sharing in teaching. From that, the new framework SASCIA (System Architecture Supporting Cooperative and Interactive Applications) was derived that directly copes with teaching situations (cf. overview in section 5). It is based on suitable components for user and session management that are maintaining context information and make use of it for special purposes (section 6). For each application to be integrated, SASCIA is containing an application frame, that con-

sists of a context information capturing mechanism, a configurable floor and consistency control and a shared whiteboard for public and individual annotation (section 7). A first step towards subsequent replay is provided by a mechanism to track teaching material including all annotations and store it into databases (section 8). Up to now, some first applications have been integrated into this framework successfully. While the notorious shared whiteboard is already part of each application frame to enable note taking, it can be used as well on its own as an application for drawing and writing in shared documents. Moreover, we describe the collaborative usage of interactive animation applets to teach communication protocols (section 9). After detailing implementation aspects and measurement results obtained with the first prototype (section 10), we conclude this report by outlining some directions for future work in section 12.

## 2 Scenarios and requirements

As laid out in the introduction, two different scenarios, one with teacher and the other one without, are motivating our work.

### 2.1 Scenario 1: Lectures

The first scenario concerns a scheduled lecture with teacher and students (cf. fig. 1). We assume most of them to be collocated in the same room. Hence, this performance can be identified by name of lecture, teacher or room. All participants have to authenticate to proof that he or she is allowed to attend resp. act as a teacher. Typically, students are not always in time due to latency of trains, traffic jam or other reasons. That means, some can enter after the start of the lecture or leave in advance. Some can not attend at all because of other activities taking place in parallel, sickness or something else. Depending on the reason, students may want to participate remotely from the beginning in real time, maybe switch to the collocated case some time later on or at least replay the lecture afterwards at a suitable moment. Hence, the number and presence of participants can vary during such kinds of teaching sessions. Moreover, devices with totally different characteristics can be used for participation. Besides available resources, input devices and operating system, even the quality of the connection can differ [5].

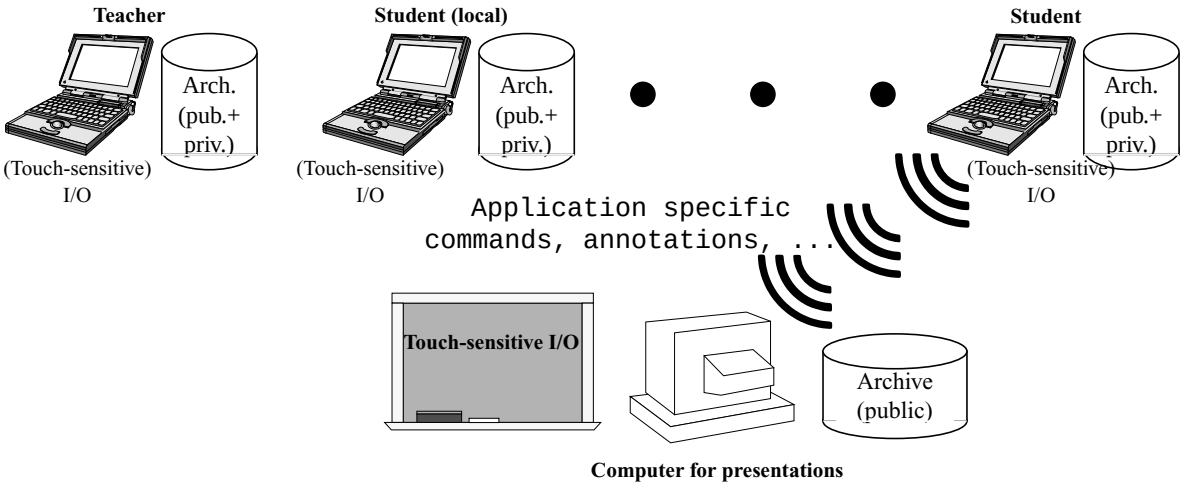


**Figure 1:** Scenario 1 - lecture in wireless environment

Traditionally, teachers of all disciplines have been writing on a blackboard, showing physical slides or experiments or have been discussing a topic with their students. All these media have been used in a combined way. Nowadays, they are relying more and more on a designated computer that is equipped with a projector and suitable input/output facilities, e. g. a physical whiteboard or graphic tablet. This opens up many possibilities like using a tool for presentation (e. g. Microsoft PowerPoint, StarOffice or Acrobat Reader) or demonstrate some application like an engineering or simulation environment (e. g. products like JBuilder, Rational Rose, Ptolemy, MatLab). In contrast to former times, it is even simpler to play videos. Standing in front of a touch-sensitive board, one can write and draw either on blank screen (ideal instrument to support discussions) or on projected application windows as well as erase former annotations. Last but not least, even tele-experiments come into sight, where the teacher is controlling some equipment directly from the auditory as discussed already in the introduction. To this end, one must know exactly about the current state of all components involved into the experiment.

Thinking of students carrying mobile and handheld devices with them, opens up new possibilities for them as well, thus increasing interactivity during lectures. At first, students are enabled to download material. Moreover, they should be allowed to not only perform private note taking but as well point and annotate on public windows as projected on the screen. If available, a touch-sensitive input device like a special notebook or tablet would be the matter of choice. For subsequent learning and preparation of exams, it is useful to record all these data by carefully obeying the correct temporal ordering and durations. Due to copyright and license restrictions, reasonable precautions must be taken regarding download as well as recording of material. Last but not least, even control of presentations and further kinds of applications by students should be possible but treated with care. Depending on the nature of an application, its control can require special abilities resp. result in more or less serious problems in case of misuse (e. g. a tele-experiment with expensive equipment). In contrast to this, there is limited harm and much benefit provided by a control of the loudspeaker of the room via the microphone of a mobile device because this feature can make a student understandable even in a very large auditory or from remote.

While originally the teacher will be having the right for such tasks, she should be able to grant it wholly or in parts to a student on request. Having controlled an application by more than one person at the same time increases the need for context awareness, i. e. complete information about the state of all components and actions of other participants. To enable the teacher to get aware of questions, comments or general feedback of students and react accordingly, a mechanism is needed to identify such events. Depending on the size of room and crowd, there is a risk to overlook raised hands and be unconscious regarding the understanding of students. In remote settings, this problem is getting even more serious. Moreover, when addressing students one cannot exclude the possibility that they want to stay anonymous. Hence, they should be identified either by real name or by seat number. In any case, it would be advantageous to have available a map containing all participating persons with their position and current state (hand raised, floor owner, attentivity etc.). This feature can help teachers as well as students. Especially in a large auditory, learners may want to find their friends for further talks after the lecture. In some cases, it can even proof useful to select other participants and discuss topics in a manner that imitates whispering. This can lead to perturbation, therefore the teacher has to decide whether to allow or prohibit this functionality.

In general, teachers must be able to guarantee a proper performance of the lecture and control students' actions, especially revoke access rights from them at any time. In the worst case, even removal of students from the session should be possible. Furthermore, wireless connection to the rest of the world bears the risk of seducing students to do different things instead of following the lecture. Thus, some mechanism is required to prevent them from too much distraction. Finally, all the functionality as described above should be provided without too much further effort, e. g. for setup of supporting tools (cf. [9]).

### 2.2 Scenario 2: Learning and performing experiments without guide

The second scenario takes place in a group of students without any supervising person being present. These students are discussing some topic (exercises, exams) and their encounter can arise in a spontaneous or planned manner with local and remote people. For instance the team A in fig. 2 is constituted of two collocated students at the left and one remote student at the right hand side. Perhaps the participants have to produce some common result, maybe in competition to other groups. Hence, others must be prevented from spying (student of team B in fig. 2). Such meetings need not take place in a room with a designated computer and whiteboard at hand. Instead, students can sit elsewhere inside or outside buildings without any such equipment.

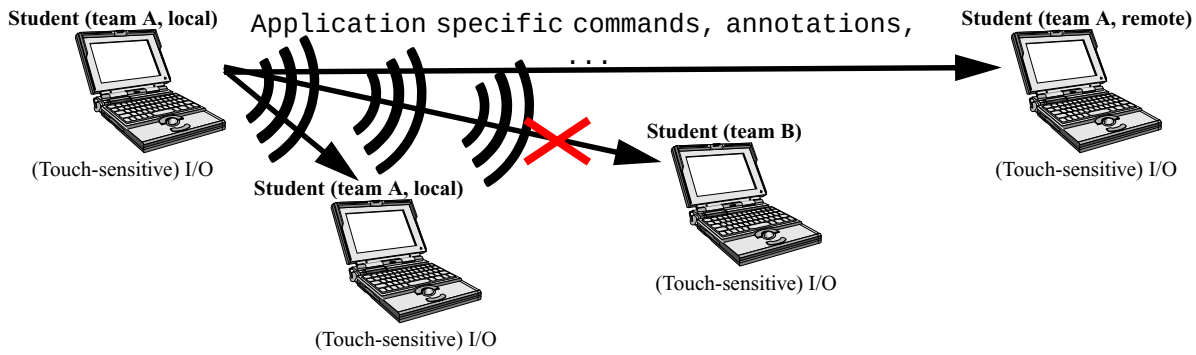


**Figure 2:** Scenario 2 - student encounter in wireless environment

But nevertheless, they are equipped with mobile devices and share applications via wireless connection. After authentication, they can even access file servers and tele-experiments on the campus. Since all participants are of same level, the problem can arise whom to assign the right to control the session or remove disturbing elements. Depending on the social abilities of participants, some kind of group decision support can be worth while. Analogous to scenario 1, further facilities like support for latecomers, public and private annotations to shared applications as well as recording and replay should be available.

### 2.3 Requirements

It can be seen, that both scenarios are posing a number of different requirements that can be classified into specific and general ones. Specific requirements are the following:

- Awareness in class regarding participants, their identification, seat number, role, level of attentiveness and understanding, hand raising.
- Visibility of relevant application information to be able to control them in a proper way.
- Openness of the system, i. e. the possibility to plug-in applications from different teaching disciplines, going hand in hand with the requirement to integrate further services, especially those providing awareness information.
- During a session, a whole set of such applications should be usable enabling participants to switch among them.
- Different user groups and hierarchies (teacher, students) have to be coped with, resulting in different rights (see below) and alternate interaction styles: moderated session or scenario with peers, whispering allowed or disabled.
- Depending on role: right to control session and participants.
- The following rights are available for an authorized person (teacher, elected session chair) and granted to others either in an isolated or combined way:
  remote control of audio channel,
  pointing and public annotation in application windows (during lectures, these windows are shown on the projector),
  remote control of applications including their equipment.
- Private annotation.
- Access to recorded material including public annotation by taking into account copyright restrictions.
- Recording in such a way that real time replay of the whole presentation including audio, public and private annotation is possible afterwards.
- Prevention from distraction must be provided.

The inclusion of video streams has not been mentioned explicitely due to the fact that it is similar to audio. Moreover, cameras mean an expensive equipment and require extra effort in usage. Hence, they are worth while for special cases only like the performance of some experiment but not for teaching scenarios in general.

General requirements concern implementation aspects and the runtime environment of a system with the functionality as described above:

- Easy and quick set up of sessions to not bore teachers and learners but let them concentrate on the essential. Support for latecomers.
- Need for authentication and possibility to restrict membership.
- For usage at universities, the system must be independent from platform and available for free.
- Sufficient stability and security must be provided.
- A consistent user interface should be offered.
- User interfaces should be adapted to different natural languages with minimal effort. This is underlined by the fact, that international studies are taking place at the same institution than national ones.

In the following we examine, whether existing systems do cope with the catalogue of these requirements.

## 3 Examination of existing products and related work

Before developing a new system from scratch, we investigated existing systems with respect to the requirements as collected in the last section and the effort arising for an eventual extension. In the following we briefly sketch systems being available for free (at least for non-commercial usage) and summarize results for the most significant ones in table 1 and [2] (cf. the treatment in [50]).

According to ITU standards [31], ***NetMeeting*** [47] from Microsoft is offering audio, video and whiteboard functionality as well as application sharing. NetMeeting clients can call either a central server or a client already being connected to a server. Names or internet addresses of other hosts are either known directly or identified via the Microsoft Internet Locator Service based on LDAP (Lightweight Directory Access Protocol) but not completely compatible to this standard [45].

Potential receivers have to decide whether to connect automatically in case of an incoming call (optionally with password) or demand a confirmation by the user via a popup message. Especially for lectures, the latter behavior is intolerable. In any case, the resulting network like structure causes problems if those people want to leave in advance who are placed in the middle. In contrast to this, latecomers are supported insofar as they can join at any time after start. Authentication of participants can be guaranteed by means of the Windows Certificate Store. To achieve awareness in class, the list of all participants is available. Moreover, participants can chat among each other but without any possibility for a teacher to stop them.

| | **NetMeeting** | **MBone** | **JETS2000** | **VNC** |
|---|---|---|---|---|
| Available applications | Whiteboard, Chat<br>ITU T.126, ITU T.128<br>Windows applications | Whiteboards, Editor | Whiteboard, Chat | All kinds of applications |
| Formats as supported by whiteboard | proprietary & application specific | text, ps, pdf, gif, jpeg, SGML | text, gif, jpeg, slides in html, drawings, videos (ITU H.263), VRML | A lot of encodings for different formats |
| Hierarchy of participants | (+) | - | + | - |
| Interaction styles | moderated only<br>chat in subgroups<br>(optionally encrypted) | MACS: various policies | moderated only<br>enable/disable chat in subgroups | - |
| Identification, basic awareness | participants list with names<br>location fix | MACS: icons, available resources<br>mlb: participants list with names | participants list with names | - |
| Further awareness support | - | mlb: hand raising, feedback | voting | (application only) |
| Removal of participants | + | - | (-)<br>exclude from watching & acting | - |
| Pointing & public annotation in application window | Whiteboard: both<br>Arbitrary application window: pointing only | Whiteboards (mlb, MACS): both | Public annotation in whiteboard (still image required), public chat, no pointing | + |
| Private annotation | - | - | - | - |
| Data handling | ITU T.127 | - | +/- | |
| Copyright | + | - | + | - |
| Recording, replay | - | + | - | - |
| Latecomers<br>Early leavers | +<br>- | + | +/- | - |
| Firewall:<br>inside<br>outside | <br>+<br>+ | <br>+<br>+ | <br>+<br>? | |

**Table 1:** Comparison of most significant existing products (specific requirements)

The initiator of a session can remove participants and control the whiteboard but a priori has no further rights. Each arbitrary participant having started an application may share control with others (compatible with ITU T.128) and revoke the floor at any time. In class, this can lead to numerous kinds of perturbations (an even more dangerous functionality concerns the one of releasing the desktop, i. e. completely allowing control over a computer for remote access; this feature is applicable in special cases only). While copyright restrictions are obeyed by transmitting bitmaps, the network load is increased tremendously. Application sharing can be used in combination with a physical whiteboard, but in this case annotation is not possible any longer. This holds as well for the whiteboard (based on ITU T.126) that provides drawing,

textual input and pointing as well as individual switching among pages. Due to the fact, that application sharing and whiteboard are having separate windows, annotations can not be performed on application windows in general. Hence, annotations are restricted to the whiteboard and the proprietary format as supported by it. Pointers of different users should differentiate in colour, but this feature does not work quite well. Moreover, its usefulness in classes of large size is questionable. Private annotation and recording are not supported at all. Because unicast is used for data transfer (compatible with ITU T.127), problems with bandwidth can arise for more than 20 participants and especially in the mobile case. What is even more, the configuration of the operating systems has to be modified to allow more than 20 tcp/ip connections (if not using the tree like structure as mentioned above).

A software development kit (SDK) is available that allows the integration of new tools. As demonstrated by the inclusion of a pointer into video, this kit is not easy to handle [2]. Moreover, an API (application programmer's interface) is not offered for internal components, especially floor control is not extensible. Existing user interfaces have been built in a consistent manner. The stability of the whole system seems to be acceptable. Whereas inside firewalls, the whole functionality is at hand, problems regarding audio and video apply to the case of conferences across firewalls due to dynamically negotiated ports.

As a whole, NetMeeting is a common tool in business environments where participants can be assumed to behave properly. A very serious disadvantage concerns its restriction to Windows platforms.

Like NetMeeting, ***Groove*** [15] simply relies on Windows operating systems. It provides a list of several synchronous features like voice over Internet, textual chat, shared viewing of documents and pictures, co-browsing and co-editing of Word documents as well as asynchronous collaboration support. Its drawbacks regarding the usage for teaching are similar to the ones of NetMeeting.

Based on ***MBone*** (Multicast Backbone, [43]), a number of tools are available for conferencing in distributed manner without a central server. We investigated the Session Directory sdr 3.0 and the UCL Shared Whiteboard wbd 1.0ucl4 as well as more sophisticated whiteboards like mlb [44] and MACS [5]. With sdr, sessions can be announced with different degrees of visibility: local, region, world. Participants can join an announced session at any time during the performance thus supporting latecomers as well. The UCL Shared Whiteboard wbd even transmits existing drawings to such persons (for mlb this functionality is planned). Authentication based on PGP (pretty good privacy) is provided for session management but not for whiteboards. Due to its main purpose of being a multimedia conferencing system with audio and video windows included, mere usage of the shared whiteboard wbd lacks the visibility of other participants for team awareness. MACS overcomes this problem by a special visualization module to show all participants including their configuration (desktop, mobile device, available equipment) in 3D manner. Another solution is provided by mlb that offers a list of participants and tools for pointing, hand raising and feed back.

MACS offers different policies for granting access rights but wbd and mlb restrict to a simple floor control mechanism based on passing the floor among participants. All three whiteboards allow import of a couple of file formats (e. g. text, postscript, pdf, gif, jpeg) but no real application with the exception of Java applets. With wbd on Windows NT4.0, we encountered serious problems when trying to import a postscript file with landscape orientation. Only public annotation is possible but private one is excluded. Whereas recording can be performed in various ways (e. g. [27], [60]), data transfer is not supported.

The source code of most MBone tools as well as compiled versions for Unix and Windows are available. Hence, extensions in functionality and adaptation to further platforms can be performed. Setting up the tools requires a lot of effort and experience. When having found a configuration that works, the system is running sufficiently stable inside and through firewalls. The usage of MBone tools is wide spread, especially in the academic community.

***JETS*** Version 2.0 and JETS2000 Alpha 1 (Java Enabled Telecollaboration System, [33]) is developed at the Multimedia Communications Research Laboratory of the University of Ottawa. It is based on the concept of a Web server and client applets running in browsers. Its main focus is on the functionality of a whiteboard and appropriate control of access rights (no access at all, mere watching, watching and acting) for students by a teacher. Integration of components for audio and video is still under construction. Besides common formats, the whiteboard even enables import and control of videos and VRML files. To some extent, the problem of copyrights can be coped with by using bitmaps with snapshots instead of the document itself. Like NetMeeting, JETS2000 is supporting latecomers insofar, as login is available at any time, but transfer of missed material is not included. Participants have to authenticate with name (or nickname) and password as distributed by the teacher. Especially in classes of large size, this means a tremendous effort. Usage of different web sites, determines the role of a participant: chair person or listener. The chair person can modify access rights of other participants to the whiteboard and all its functionality, whereas listeners have to request for it. Current access rights of a participant are indicated by colours (red, yellow, green). Besides modifying access rights, the chair person can decide about enabling or disabling private chat among participants.

The system provides an application programming interface (API) for further development, e. g. for the integration of applications into the whiteboard. It mainly consists of two components, one for a broadcast channel to participants and one to be able to lock an application thus providing exclusive access. Unfortunately there are some restrictions. For instance, core components like session management do not offer an API. Some further shortcomings apply as well. The whiteboard allows drawing only but no textual input nor pointing or storing of such drawings. The possibility of private annotation is not mentioned. There are many different windows involved into the system that partially lack usability. Furthermore, the system is based on dynamically negotiated tcp ports. Hence, problems can arise when using firewalls.

Due to be written in Java, JETS2000 is independent from platform. But it is rather sensible regarding the Java version. For instance, there have been serious stability and performance problems with all browsers we used for testing the system (Internet Explorer, Netscape on Windows NT 4.0 and on SunOS).

***VNC*** (Virtual Network Computing, [55]) mainly aims at making available control of devices from anywhere in a single user scenario. It is built in a client-server based structure, where the server is implemented on the framebuffer level and clients are acting as viewers. By allowing more than one viewer simultaneously, multiple users are enabled to share applications. Due to the main goal, only authentication of clients is supported without any further administration of participants and their context. Moreover, recording has not been mentioned. VNC is an open source project, where material can be downloaded. VNC servers are available for X Server in the UNIX world and on Windows as well thus providing platform independence for these systems. Parts being written in Java and the possibility of a Web based user interface even extend this independence and enable further development.

Originally, ***JaTek*** (Java Based Teleteaching Kit, [32]) was developed at the Technical University of Dresden. Among other components, it is containing the modules JaWos (Java Based Workgroup Support) and Classroom that aim at supporting groups of geographically distributed students by means of a shared whiteboard. The system is based on Java and therefore independent from platform. In the meantime, it became a product and essential parts are not available for free any longer.

| | **NetMeeting** | **MBone** | **JETS2000** | **VNC** |
|---|---|---|---|---|
| Set-up | call:<br>automatically accepted<br>or to be confirmed | +<br>(announcement - join) | | |
| Openness | SDK (restricted) | + | + (API) | API |
| Platform independency | -<br>(Windows) | (+)<br>(Unix, Windows) | + (Java based) | +<br>(Java based parts) |
| Availability | + | + | + | + (open source) |
| Stability | + | + | (Browser) | |
| Authentication | certificate | sdr: pgp<br>wbd: - | password | password<br>(challenge response) |
| User interface | + | - | - | |
| Firewall | + | + | +/- | |

**Table 2:** Comparison of most significant existing products (general requirements)

It can be concluded that neither of these systems is covering the whole list of requirements. Most seriously, annotation functionality is offered for whiteboards only. In the best case, such a whiteboard allows to import slides or videos but does not support arbitrary application windows. Moreover, none of the systems provides private annotation being needed heavily in teaching context. Last but not least, the control of access rights is not solved in a satisfying way. Hence, we decided to design a new system.

## 4 Model for application sharing in teaching

When studying application sharing in teaching, three main components become apparent: sessions, participants and applications. They can be characterized by static and dynamic properties and relations among them. The most important static attribute of each component is its *identification*. In the following, our focus is on dynamic behaviour. At each point in time, a component and its relations exhibit a certain state with regard to attributes. While these states are changing, a history is arising. Moreover, persons and things can be subject to a predefined schedule. As a whole, one has to regard the

evolution of states from past via presence to future. All this information about individuals and their environment, the so-called context information, can be meaningful to other participants to a more or less extent. Hence, we have to take a closer look on context data that are relevant from our three types of sources as mentioned above.

A *session* is determined by the following information:

- History: past sessions being related somehow, progress of the current session in time and content, i. e. the overall results achieved so far.
- State of sessions being related to this session and taking place in parallel.
- Number, identification and state of each participant.
- Number, identification and state of each shared application.
- Schedule.

Teaching sessions cannot be seen as isolated performances but depend on each other with regard to their content and participants. Each single part contributes to the overall learning process by delivering a subset of knowledge to mostly the same people. A session starts at some designated point in time and is making progress. Participants are discussing topics and presenting slides or experiments to mediate facts and relationships. To this end, they are using appropriate applications. Though not a regular case, the possibility exists that several sessions are taking place in parallel and have to be synchronized according to some predefined behavior. For instance, a large class can be split into small groups for an exercise and joined afterwards to continue plenary discussion. If they are mutually aware of their state, groups are getting the chance to adapt their speed and reduce the overall waiting time. A similar benefit comes with the possibility of joining sessions at an appropriate moment. This functionality can be useful for multiple interesting lectures taking place in parallel or for life long learning, i. e. for switching between sessions and activities (cf. [18]). In general, the sequence of treated topics can be predefined in a more or less strict way. Moreover, the session can be planned with a definite completion time or be open-ended.

Some relevant context information about *participants* of a session has been mentioned already in Section 2. The complete list looks as follows:

- History: knowledge level and abilities
- Current state: Location, role, desire to ask/comment, degree of attentiveness, understanding, equipment and resources
- Schedule
- Privacy concerns

Teachers and students are motivated differently with regard to their interest in a certain participant. This is one example for the need of roles; additional ones are application specific as described in the next section. Teachers have to take into account the history and current state of at least the majority of learners to be able to adapt their presentation adequately. This holds even for the equipment of learners due to the fact that the most wonderful multimedia slide is not worthwhile for people participating with small wearable devices (cf. [5]). Moreover, teachers should not overlook a student's intention to contribute actively, e.g. by raising his hand. As described in the first scenario, finding teams of co-learners motivates the interest of students in each other. For them, knowledge level, location and schedule are the most relevant context information. But application specific roles can be of interest as well. In either case, it must be totally up to each participant, whether to publish context information to everybody, to a subset of participants or not at all. If information is visible to others, it is useful in collocated scenarios already, especially in a large auditory. For remote participants, it plays an even more essential role. The most suitable way to present context information about participants is a map containing all participating persons with their position and additional information at appropriate level of detail.

For applications, only a few general statements are possible. Very often, more than one *application object* is involved. Similar to sessions and participants, such application objects should provide the following context information:

- History
- Current state
- Schedule
- Relations to other objects and to participants

Depending on the type of application, its objects either belong to the real or the electronic world or to both. As such, they are subject to a certain life cycle starting in the past and moving toward future in a planned or ad-hoc way. Objects can be interrelated in temporal, spatial or other fashion. Relations to participants define roles with regard to a certain object, i .e.. the right to modify or view its state. All these abstract considerations will be concretized when coming to examples in section 8.1. Before that, the realization of the model for application sharing in teaching is examined.

## 5 Architecture

Basically, two alternate architectures are available for application sharing: one based on the client-server model and the other one on peers. When considering the lecture scenario taking place in a designated room it is probable to have at hand a computer and projector for presentations because more and more rooms at universities are going to be equipped with such devices. Hence, for this case it is obvious to have the server for application sharing running on this mostly powerful presentation computer whereas mobile devices with possibly less resources host the client part. Regarding however the scenario of encounters among students that can take place anywhere and maybe without any infrastructure at hand, a peer like fashion would be more adequate. According to [36], encounters of rather short (seconds, minutes) and of longer duration can be differentiated. To support the first one of these cases, they offer the development enviroment Proem. For teaching purposes however, meetings of a certain duration longer than just a few minutes are more relevant. Therefore, the tool kit Jxta [37] being focused on this case will be considered more closely.

In a first step, we developed a client-server-based system [51] by keeping in mind the general architecture of conferencing and application sharing systems (cf. [52]). But the peer case will be tackled with in the near future [49]. As shown in figure 1, clients and server are physically connected by a network, that can be of fixed or wireless nature. A suitable communication mechanism based on this network is providing logical connections between all components involved. We are assuming a channel-based metaphor where each participating component can register to get events forwarded to the channel by others. By encapsulating the communication mechanism, different systems can be used and have to be specified for each instantiation.

Before diving into details, we briefly introduce the functionality of all components. With a server running on some host, authenticated users can set up sessions via the *session administration* components. A public announcement of a session can be performed by inserting it into the *session directory.* Via *participation* components, one can join a session either by selecting the appropriate session name in the session directory or by directly specifying the server and session name. Moreover, in combination with the *user directory*, these components care for proper authentication of participants. Components to capture *context information* about participants, applications as well as the session as a whole increase the awareness in class and serve further purposes.

After all these preparations, the actual presentation or meeting activities can take place. To this end, participants are using applications. Each of them is integrated into the system by instantiating and specializing a generic *application frame*. It consists of a number of components and provides functionality as follows:

- A configurable flo*or control* component checks access rights of participants for input and control of the application.
- *Consistency control* cares for a semantic check of the feasibility of actions.
- Application specific *context* is captured and evaluated.
- It enables public and private *annotation*.
- It embeds the communication system.
- It provides *archiving* of all relevant data including context information into databases for later retrieval. Its usage can be internally or externally. For instance, recorded information can be needed by the component for context handling to derive missing data. External usage means retrieval by students.

In the following sections, we describe each of these components including our example applications more detailed.

## 6 Session and user related aspects

Components for user and session management as well as participation are straightforward and similar to those of existing systems. Hence, we sketch them briefly by emphasizing our new features (cf. fig. 4). For further details please refer to [62].

### 6.1 Session administration

According to the client-server based architecture, the first thing to do is to start the *session administration server* (either locally or remote) that cares for an superordinate management of all session activities. When invoking this server, a number of configuration data can be appended, e. g. addresses of directories, a list of all applications being available on this host for sharing purposes, the selected communication system and default values for all possible parameters (see below). Upon start, the session administration server initiates a channel of the communication system for usage by clients that must be authenticated against data of the user directory. Thus, sessions can be announced and started via this channel at any time and from any place. To this end, the *session administration client* is used to specify the following parameters:

- A unique identification of the session, e. g. by name of lecture and teacher, room, date and time of begin.
- Optionally a meaningful description.
- Those applications, that are started initially.

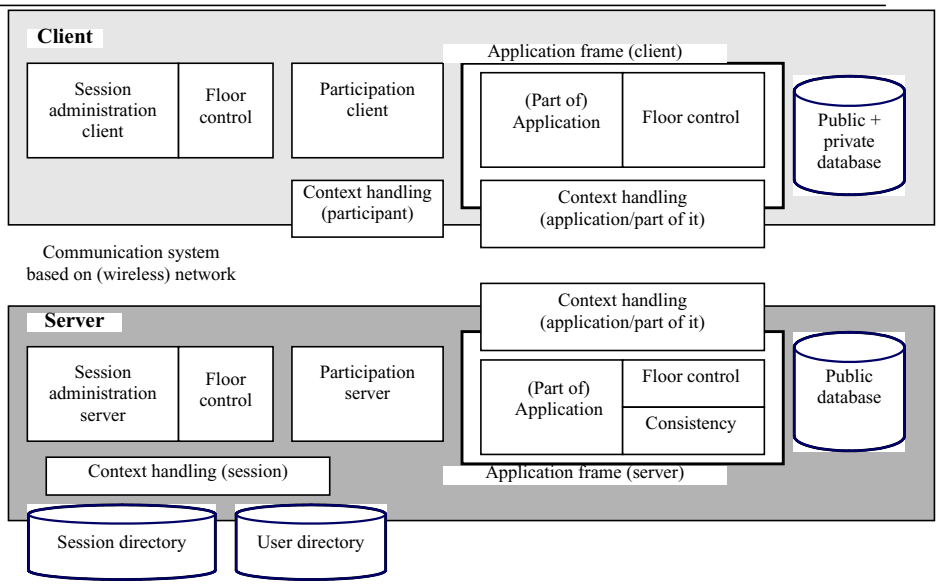


**Figure 3:** SASCIA architecture

- For each application the style of floor control (e. g. with/without chair person, see below).
  Among others this applies to session administration during this session itself.
  Different sessions managed by the same administration server can feature unequal styles.
- Maybe for each application: needed documents including their copyright restrictions.
- Optionally names for public archives.
- Specification of the degree of visibility (announced to the world, restricted) and the address of a session directory (optionally).
- Optionally the lists of admitted and of excluded users.

For each parameter not being specified explicitely, a default value is used. During a running session, each of these parameters can be modified, additional applications can be activated and others deactivated at any time. Even removal of originally admitted participants is possible. Such operations for session control can be performed by the floor holder for session administration only. By default, either the person with highest priority (see characteristics of participants as contained in user directory) or the one who starts a session or first joins it automatically becomes floor holder for this task if not having the majority vote against him. Like with ordinary applications, this floor can be passed to any other participant.

### 6.2 Session directory and context

Depending on the two scenarios as described above, a session can be planned in advance for an open auditory like a lecture or it takes place in a more spontaneous and maybe closed manner like encounters, mainly among students. In accordance with that, announcement and start of a session are either separate events or performed directly one after the other. In any case, all parameters of the session are manifesting a new entry in a session directory. Hence, one can check for duplicate specification of sessions with same originator and date. Depending on the intended visibility, the whole information about a session is either publicly available to all participation clients or restricted somehow (cf. functionality of [43]). This can be achieved by a hierarchy of multiple such session directories, e. g. one being local to the administration server and further ones being central to whole sets of servers. The highest amount of visibility is reached at the root of this hierarchy. Additionally, a directory at any level has to guarantee that unauthorized users do not learn about a certain session. Authorization is determined based on the user's characteristics (see user directory below).

Besides the announcement of a session, the session directory collects all dynamic data thus maintaining the *session context*. It includes the following to fit the above introduced model:

- time of start and end of the whole session,
- joining and leaving, resulting list of participants (including the context of each participant, see below),
- activation and deactivation of applications,
- application specific information (see section 8.1), e. g. number of slide currently presented or current topic.

This information provides awareness in class. As described in Section 4, people can profit from it in various ways.

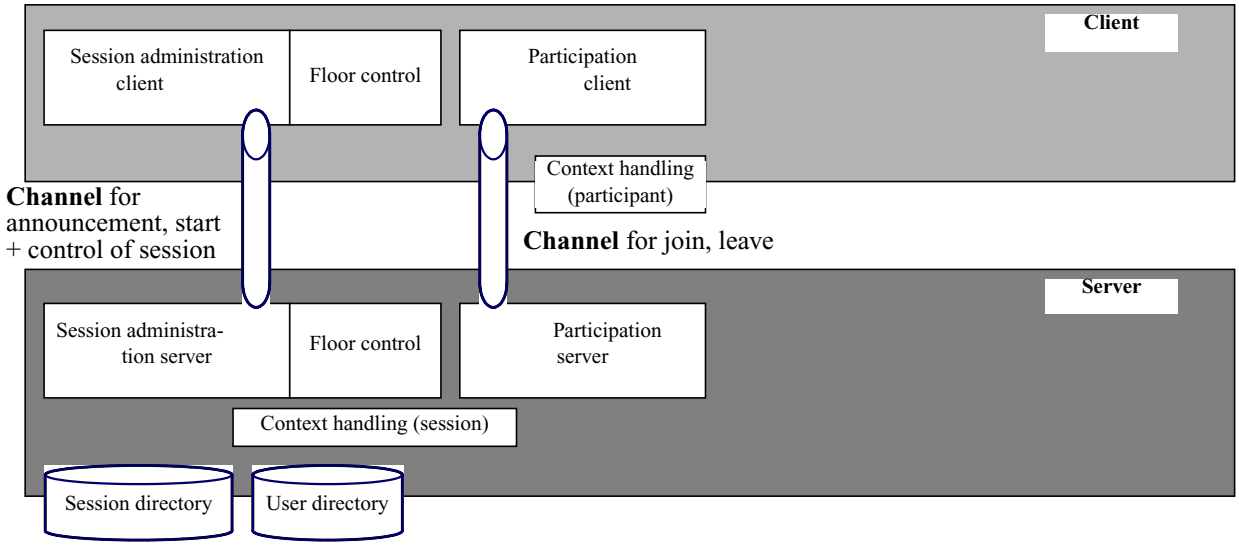


**Figure 4:** Session and user related aspects in SASCIA

### 6.3 Participation

For each session, a *participation server* and all relevant application frames are started that offer their own sets of channels for clients. Based on the *participation client*, a user can transmit a join request to this server. This happens either by selecting the appropriate name of an announced session in the session directory or by directly specifying the server and session name. The following further data are required and have to be appended to the join request:

- identification (name or pseudonym) and authentication data,
- optionally perspected role,
- current location in class, e. g. seat number as captured by some positioning system or specified manually (see discussion in section 6.4).

After checking authentication and all characteristics of this person as contained in the user directory against the session parameters as described in section 6.2, a person either is admitted to the corresponding session or rejected. In the first case, she becomes a participant and her participation client receives all necessary information to join a communication channel and start application frames. After that, she can share applications according to her role and access rights, perform annotations, manually modify her context (see dynamical data in the user directory) and leave the session by again using her participation client. As a whole, participation components are responsible for those operations during a session that are concerning one user only thus not underlying any floor control restrictions.

### 6.4 User directories and participant context

As mentioned already, all relevant characteristics of users are maintained by the *user directory*. On one hand, it manages attributes staying static for some period of time like the following:

- *Authentication* data like user name and password or certificate to support the session administration and participation server.
- The *hierarchical level*, e. g. professor, staff, student, guest for floor control components (see below).
- *Privacy* demands regarding all kinds of information, especially dynamic ones.

These data are inserted by the system administrator and the user itself and modified rather seldom. On the other hand, dynamic information is contained in the user directory. It defines the current participant's context to increase the overall awareness among people in general as well as in class by obeying privacy regards. Like for the static part, most current values of these attributes can be fed into the system manually. But as discussed in the following, this is not always the way preferred or feasible. Hence, automatic procedures are required but unfortunately lead to further problems (cf. Table 3).

The *role* of a person is the only information being easily to determine. Whether a person is allowed to act as a teacher can be derived from the hierarchical level. Application specific roles are either assigned by a teacher or agreed upon by the group in a voting process (cf. Section 7.2 and 9). In either case, they are defined manually and inserted accordingly.

Further parameters of participants are more or less difficult to capture. In the meantime, a lot of systems have been built to find out about the *location* of a person and achieve location awareness [38]. An overview is presented in [19] who clas-

sify according to criteria like cost as well as accuracy and precision that reach from some hundred meters down to millimeters. Due to high cost and effort of these systems, they are not suited to fit the needs at universities. Above all, this attribute remains the same during the whole session or at least for a certain period of time. Hence, the SASCIA system currently relies on manual input of the seat number. The same applies to the desire to contribute, despite the fact, that *hand raising* should appear more often. But the only alternative apparently being available consists in using cameras and image understanding. [44] apply manual input for students' feedback to the teacher's presentation speed. This is one hint for *attentiveness* and *understanding* of students (further information regarding the level of concentration can be derived e. g. by the examination of eye movement as reported in [14]).

| | **Intended frequency of changes** | **Manually** | | **Automatically** | |
|---|---|---|---|---|---|
| | | **Mechanism** | **Evaluation** | **Mechanism** | **Evaluation** |
| **Location, seat number** | **Low** | Directly / selection | Acceptable | Expensive equipment | Not worth currently |
| **Desire to contribute, hand raising** | **High** | Button click | Acceptable | Camera + image understanding | To be examined |
| **Level of attentiveness** | **Low** | - | - | (Eye movement) | To be examined |
| **Level of understanding** | **Low** | Feedback slider | To be examined | Currently not available | - |
| **Level of knowledge and abilities** | **High** | Self estimation | Not reliable | Test | To be examined |

**Table 3:** Comparison of mechanisms for capturing context of participants

In contrast to these parameters, the *knowledge* level should not be specified manually due to the fact that self estimation is questionable. In a first approach, participants of an unguided experiment are classified by means of a short initial test. More extensive information can be collected by permanent control and observation.

Looking at existing infrastructures at universities, a user database is at hand but generally it is just containing authentication data without any further attributes. Due to national laws for privacy and individual security, this restriction can be mandatory. While it is useful to profit from functionality being available already, the lack of additional attributes is a serious one. We solved this problem by introducing a hierarchy of user directories with different levels of information. Each presentation computer hosting a server for application sharing is equipped with a local user directory containing all necessary data about those users that most frequently are working with this server. This includes a guest account for external people being invited for a presentation. If a person is not found in this directory, a central user directory is called that relies on the existing infrastructure for authentication.

## 7 Application frame and control components

### 7.1 Application frame

An application frame including components for start and data distribution, for access control, annotation, archiving and context handling is needed for each application that is shared among participants. Almost all components of such frames must be invoked both on the client and on the server side. In contrast to this, it is not necessary and sometimes even undesired that the application as a whole is running everywhere but can be split among server and client sides (cf. the example of simulation applets in Section 9). As a consequence, the application frame client and server each is starting its correct portion of the application in combination with a shared whiteboard to enable public and private annotation. Moreover, the frame components are invoking a floor control component with parameters suited to this application and a specific control component to observe semantic consistency of all input as made by participants (at server side only).

After the instantiation phase, clients and server care for mediating commands and data between their application parts, the communication system and components for archiving as described in the next section. In any case, they are letting the floor control components check whether the initiating person is authorized for access to the application or to public annotation. Moreover, copyright and license restrictions can be obeyed by transmitting snapshots instead of actual application data. All data are treated type-dependent, e. g. the transmission of audio and video streams happens via compressed frames.

While the components for control of floor and semantic consistency are described more detailed in this section, we postpone the treatment of the shared whiteboard to section 9 due to the dual role of this component as part of the application frame and an application of its own.

### 7.2 Floor control

Components for *floor control* are a first step toward consistency of control and data. They are needed for the session management as well as for each application that is shared among the participants (cf. [50], [3]). Depending on the setting, participants are in a hierarchical or peer relationship. Different applications as well as these two settings are posing different requirements on floor control. To cope with the whole spectrum, a configurable component has been designed. It exhibits the following parameters:

- Floor size
- Sub-floors to enable a partitioning of the floor.
- Session style: moderated or voting to decide about granting or revoking the floor.

We introduce the *size* of a floor to be a number greater or equal to one. It stands for the number of participants that is granted the floor simultaneously. Thus, strict as well as relaxed conditions for sharing applications can be realized. In a lecture, rather the first case will be the one preferred because either the teacher or a student is allowed to perform actions or annotations in an application window. For a discussion among students, simultaneous acting can be more advantageous to some extent.

If not set to one, the floor size should be specified as fraction of the group size to automatically adapt it if the number of participants changes. Otherwise two kinds of side effects might appear, either the unintended case of not letting everybody get the floor simultaneously in case of latecomers or the mostly uncritical one of multiple assignment of the floor to the same person after early leaves.

The concept of *sub-floors* is motivated as follows. When thinking of simulations or games, participants can play different roles when using the same application. Hence, a partitioning of the whole floor makes sense to control input to each of these roles separately by a designated sub-floor. The principle behind coincides with the assignement of floors to single resources as introduced by [21], but is more concrete and closer to the application level. We further proof the usefulness of this mechanism in the context of our second example application, shared usage of simulation applets for learning communication protocols.

If a participant wants to perform an action and currently does not possess the floor or sub-floor, he has to request it. In a lecture or more generally a *moderated* setting, the chair person is informed about the request. To be close to real life, this person can be shown a map of the classroom with all those participants marked that are competing for the floor. Based on this map, she can decide whom to grant the floor at first and how to proceed.

In a setting with peers, granting the floor can be supported by *voting*. To this end, each participant is informed about queued floor requests and votes for each of them. The participant with the majority of votes gets the floor. In case of deadlock, either the team is informed about the situation or the system determines one of the candidates automatically. This behavior is subject to configuration.

### 7.3 Control of semantic consistency

Even in case of input being in concordance with access rights of a participant, the semantics of this action can be faulty in the current context. While learning by making errors and observing the effect of these errors is a very effective one for students, it depends on the application whether consequences of erroneous behaviour can be tolerated or not.

Regarding applications like talking to a microphone, gesturing toward a camera or using an electronic whiteboard, errors are difficult to specify and detect but are annoying for other participants in the worst case. A similar example is given by the mechanism to experiment with communication protocols as laid out in [7] and [54]. It consists of simulating and interacting with different protocol variants and watching the resulting behaviour with respect to correctness and performance. In such simulation environments being completely restricted to the electronic world, faulty usage can be tolerated as long as the availability of devices is not touched.

But when thinking of further applications, errors can even lead to serious problems. In contrast to mere electronic simulation environment, this holds for real world experiments. Taking examples like a railway model or chemical reactor, train crashes and explosions lead to damages and loss of expensive material. Hence, erroneous input must be recognized and prohibited before causing such effects. This applies already for the single user case but even more for the complicated version of multiple participants acting in different roles.

It can be concluded, that the specific rules of an application have to be specified. Only then, the input of participants can be checked automatically against these rules and be accepted or rejected accordingly.

## 8 Application specific context, recording and exception handling

### 8.1 Context handling

By the interaction of people with an application, state changes of application objects are provoked (if allowed by control components for floor and semantic consistency, see above). Thus, the *application specific context* is changed as well that is comprising internal states and those of the environment. To enable collaborative usage, participants must be informed about such changes where only a subset of them can be meaningful. Hence, events of state changes have to be captured, eventually evaluated and filtered as well as distributed. To make things more concrete, we study the nature of states and corresponding changes for the most popular applications in teaching: presentation tool, development or simulation environment, real experiment in the area of natural and engineering sciences.

By means of *presentation tools*, a sequence of slides is shown. Thus, a state of this application comprises the list of those slides being treated already as well as the number of the current one. With each slide, annotations of different type are coupled closely as appended during the performance. An annotation can be of written, oral or visual form. As a consequence, even drawings, keywords and gesturing for explanatory purposes are belonging to states of the presentation application. Moreover, synchronization points are of interest, e. g. switching to or returning from another application.

A *development environment* is responsible for the creation, modification and completion of specifications (e. g. UML diagram, interface declaration, programme module) and relations among them. Hence, these specifications are constituting application objects. Their current state is one part of the state of the application as a whole. In a similar way, a *simulation environment* consists of a number of objects, each one running through a series of possible states during a specific simulation process. For instance in communication protocols, simulation objects refer to nodes, channels and messages [7]. The state of a node is comprising states of all its internal parameters, where relevant events are sending and receiving of messages of a certain type. At each point of time, a channel exhibits a certain bandwidth, throughput and error rate. Consequently, a message is transmitted correctly or with error states (e. g. lost, destroyed or multiplied).

Regarding *real experiments*, the chemical or physical state of all objects and elements involved are of relevance. Whereas states of all other applications as mentioned above are directly available in electronic form, those of real experiments have to be measured first and transferred to the electronic world. Taking for example a railway model, current position, direction and speed of trains have to be captured by a suitable mechanism [8].

### 8.2 Recording

It can be seen, that in all cases an interface is needed between an application and its application frame to deliver application specific context to the frame for distribution to participants. Like the context of sessions and participants, the one of applications has to maintained. Notably for applications, the history is of interest to help in deriving context information that is difficult to detect otherwise. Besides this, recording all application-dependent events as well as data and annotations during lectures or discussions, serves three purposes: informing latecomers about past events and data, retransmitting in case of errors and storing the whole procedure for future replay in real-time. Due to these requirements and the one to record information about all kinds of applications, a special data format has been designed consisting of the following *meta data* (similar to [22]):

- identification of the originator of an operation,
- type of data,
- global time stamp plus index,
- visibility: public or private

and the data themselves.

Thinking of later retrieval of such data in various ways, the best thing to do is to integrate a database for archiving. A first design decision concerns the number and placement of such databases. Having enough resources at hand, databases for all public data on the server and on each client as well as databases for private annotations on each client provide the maximum of flexibility and fault tolerance. Hence, we have chosen this approach for the prototype but will consider alternate solutions later on to be able to adapt to smaller devices with only scarce resources. For each application, a separate set of databases is created.

To be independent from the actual database product, suitable components (see *DBAppServerImpl* and *DBAppClientImpl* in fig. 5 and 6) are needed for capsulation. To even achieve independence from the implementation of these components, only their interface is relevant for applications. To distribute public data to clients, two separate channels are used, one for the regular and one for exceptional cases. The latter is similar to the late-join channel in [63]. Names of these channels are determined automatically by deriving them from the corresponding application name, e. g. <session name>regWB and <session name>latWB for a whiteboard application.

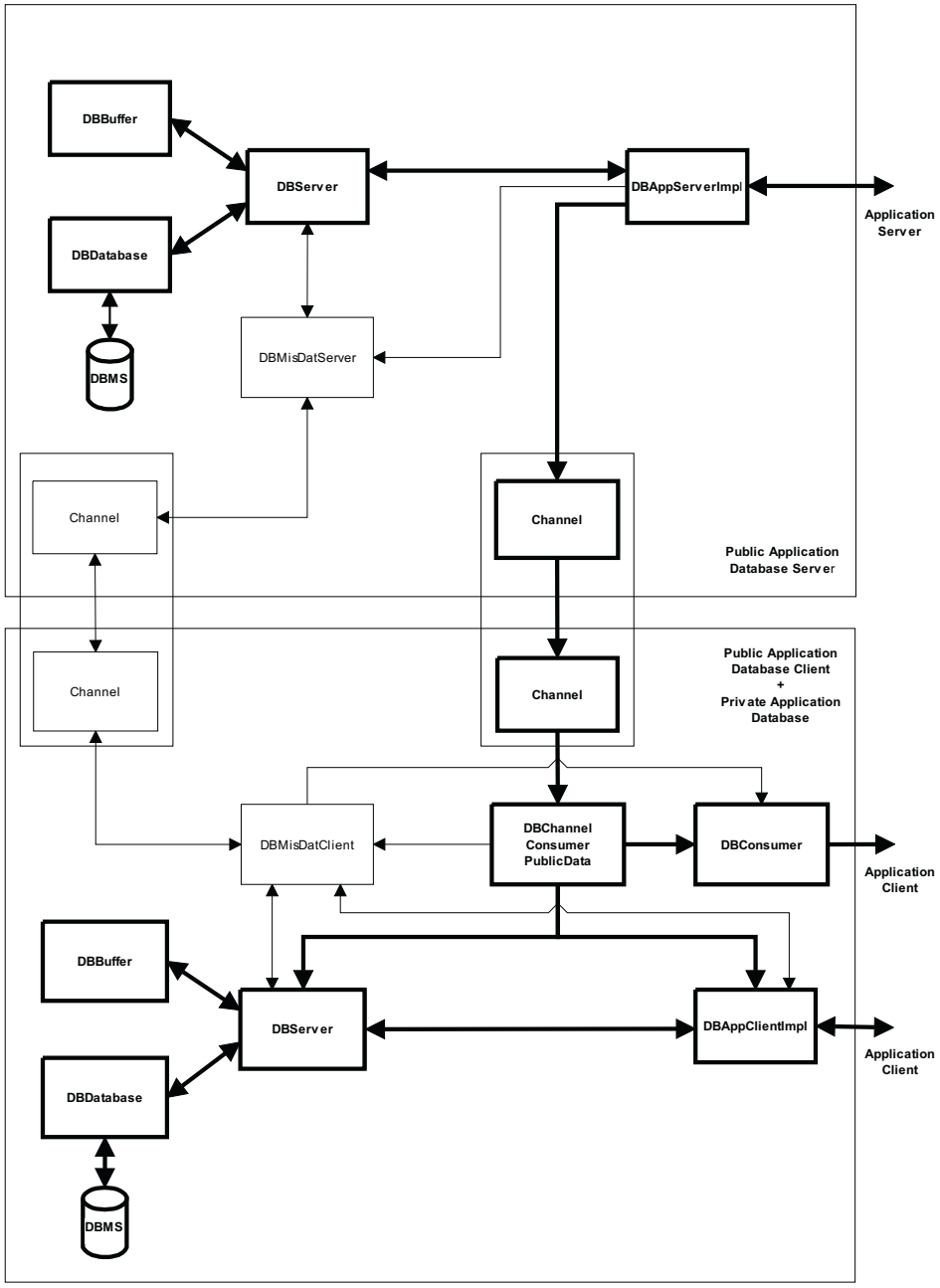


**Figure 5:** Database usage for regular case

As shown in fig. 5, the channel for the regular case is instantiated on the server by the *Application Server*. When receiving data including originator and type from the application frame on the server, this component appends the remaining meta data like time stamp, index and public visibility. Time stamps are needed to be able to replay in correct temporal order. The index is returned to the application frame and serves both as acknowledgement and reference to the archived data record. Using the afore mentioned channel for the regular case, the whole record of public data is then propagated to all clients being registered at this channel. At the same time, it is passed to a component named *DBServer* that stores it into the database. Due to performance reasons (see section 11), two steps are needed. At first, the record is stored in main memory before really writing it to the database in an asynchronous way.

On each client side, a consumer object (*DBChannelConsumerPublicData*) has to register at this channel. Thus, it receives all data transmitted through this channel. It passes them to its local DBServer that stores all data into its local database in the same way as the corresponding component on the server (see above). Moreover, meta data contained in the data record are propagated to DBConsumer and DBAppClientImpl. *DBConsumer* informs the application frame run-

ning on the client about the arrival of new data via a suitable event containing all relevant meta data. This enables the application to read new data at an appropriate moment.

*DBAppClientImpl* is responsible for private note taking that is performed without involving the server. To this end, it is receiving private annotation by the application frame. For proper ordering of public and private data, a special numbering scheme is used. Indices are built in the form of `m.n` where `m` signifies the highest index as appeared in public data and received by DBChannelConsumerPublicData. `n` denotes the current index of private data. Public data themselves are having a zero in the private index. As soon as the next public data arrive, the index of private data continues with `m+1.n+1`. Let us consider e. g. a private data record arriving between public entries having 1.0 and 2.0. Hence, its index is 1.1. Two subsequent private records between public data 2.0 and 3.0 will then get 2.2 and 2.3. Having determined the proper indices, DBAppClientImpl passes private data and corresponding meta data to DBServer for final database storage.

### 8.3 Database usage for exception handling

It has been mentioned already, that components for archiving are used not only for later replay but as well for the case of data records missing at one of the clients' databases. This arises in case of latecomers, application abort and transmission errors. As shown in fig. 6, *DBMisDatServer* and *DBMisDatClient* are instantiated at system start to care for the appropriate treatment. They can communicate via a special channel (left hand side in fig. 6) being offered by the server and joined by clients. Each one initiates and maintains a *DBList* object containing the list of indices of public data that are existing in the local database. At the server, DBAppServerImpl is responsible for any update of DBList. On the client side, this object is shared between DBMisDatClient and DBChannelConsumerPublicData (see above) who both can receive data from the server. In either case, DBList is reflecting the actual content of the local database thus avoiding permanent requests to get it.

Hence, missing data can be detected whenever DBList is exchanged among DBMisDatClient and DBMisDatServer by comparing the received object with the corresponding local one. Sending DBList from client to server actually means a request to retransmit any missing data, whereas in the opposite direction the client has to check its local data for completeness and possibly claim for them afterwards. As a whole, DBMisDat transfers its DBList to the server in the following cases:

- When joining and leaving a session or after an application crash.
- If an application has requested a data record from the local database, that is not existing there thus causing a DBNotFoundException.
- After receiving a DBList from the server containing more indices than its local DBList.

The first case is good for supporting latecomers and achieving a complete reception of data, even in face of application problems. If an application request for data leads to a DBNotFoundException, access to the database is blocked until the corresponding record has been arrived from the server. If this record is not even existing at the server, the application is informed by means of a DBNoSuchDataException.

Hand in hand with sending DBList, a *DBPublicDataEvent* is created at the client side and passed via DBConsumer to the application. This information enables the application to react accordingly by either blocking until all data have been arrived or by continuing its execution. Correspondingly, a concluding DBPublicDataEvent notifies about the completion of all activities regarding request for missing data and retransmission. Hence, the application is informed permanently about the state of any such activities.

After having described the behaviour at the client side, the server has to be studied more detailed. Its activities are straightforward. Whenever DBMisDatServer detects missing data from the comparison of DBList objects, it requests the corresponding records from its local DBServer and packs them into objects of type *DBMisData*. Such objects are retransmitted to all participants through the special channel in decreasing order of indices, i. e. starting with the highest index number. Each such sequence of retransmissions is concluded by propagating the server's DBList object to clients to enable them to check for further missing data. Due to this feature and the fact, that all clients are registered at this channel, subsequent arrivals of DBList from different clients can be neglected until the current one has been answered completely.

## 9 Example applications

### 9.1 Example application 1: Whiteboard

Besides a simple chat facility, two further applications have been integrated into SASCIA yet. As mentioned already, the electronic whiteboard is playing a dual role. Besides being part of application frames, it can be used for loading or unloading of one or more documents as well as for drawing, writing, public annotation and pointing. To be allowed to

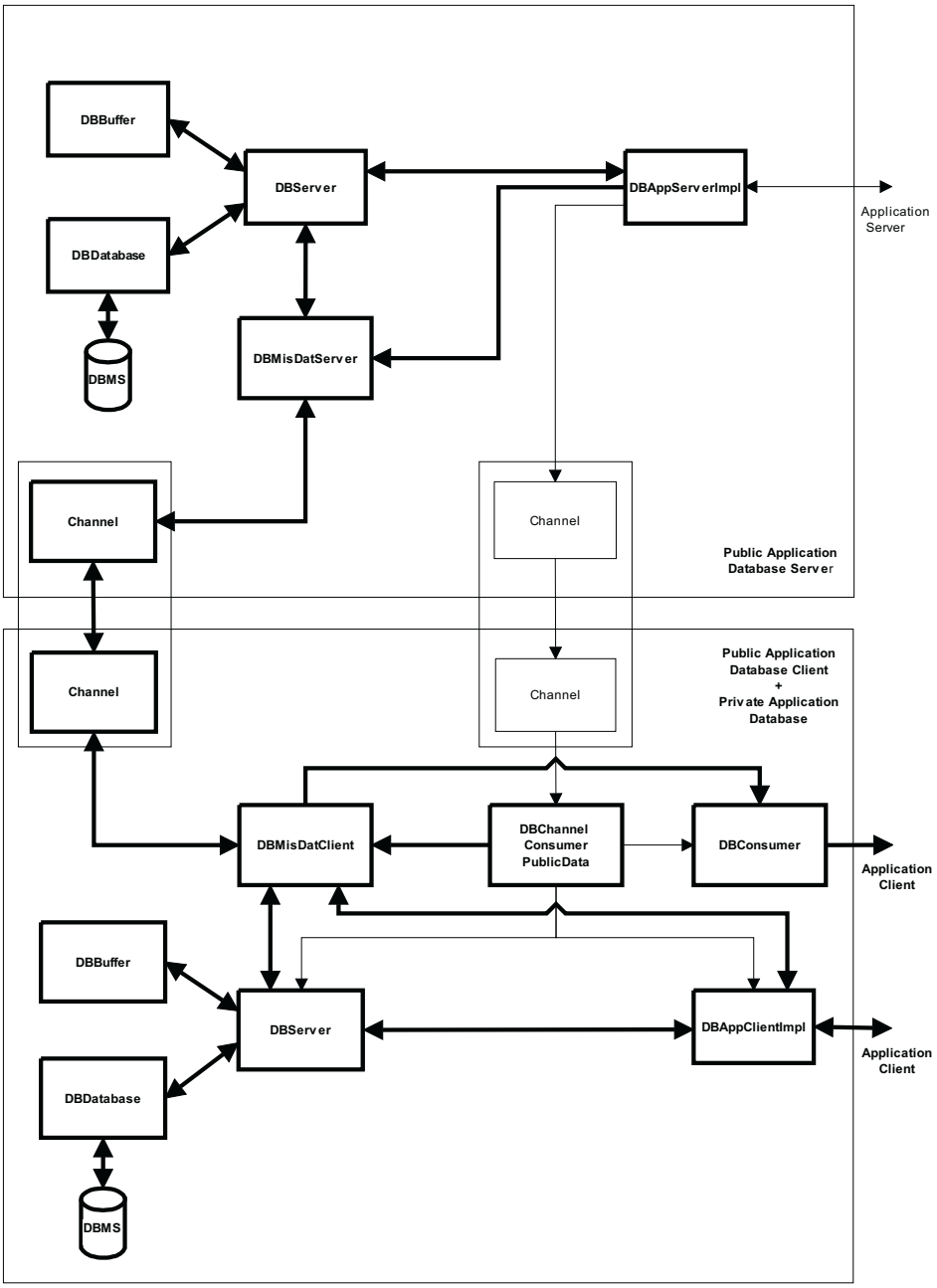


**Figure 6:** Database usage for exception handling

invoke such operations, participants must be floor holder. According to the concepts of the last section, a participant either is possessing the floor (e. g. the teacher) or has to request it. Fig. 7 presents the teacher's view who can grant and revoke the floor to/from students. As shown at the upper left of fig. 7, we extended this basic concept by a second mode for being able to apply private annotation as well. This case does not require any possession of the floor. The following tools are available with the whiteboard: pencil, sponge, marker for highlighting, text, pointer. For each of these tools, position, size and colour can be chosen. Usage of tools and input of data can be performed with devices like mouse, keyboard or touch-sensitive surface.

For each pointing device, a locator object has to be implemented. By imitating the popular mouse interface, this locator object offers operations like press, release and drag. The corresponding events are converted into specific operations and passed to an object for drawing resp. writing as well as to the local DBServer . Such events can happen on a client in private or in public mode or on the server. Regarding public cases, DBServer transmits events to its counterpart on the

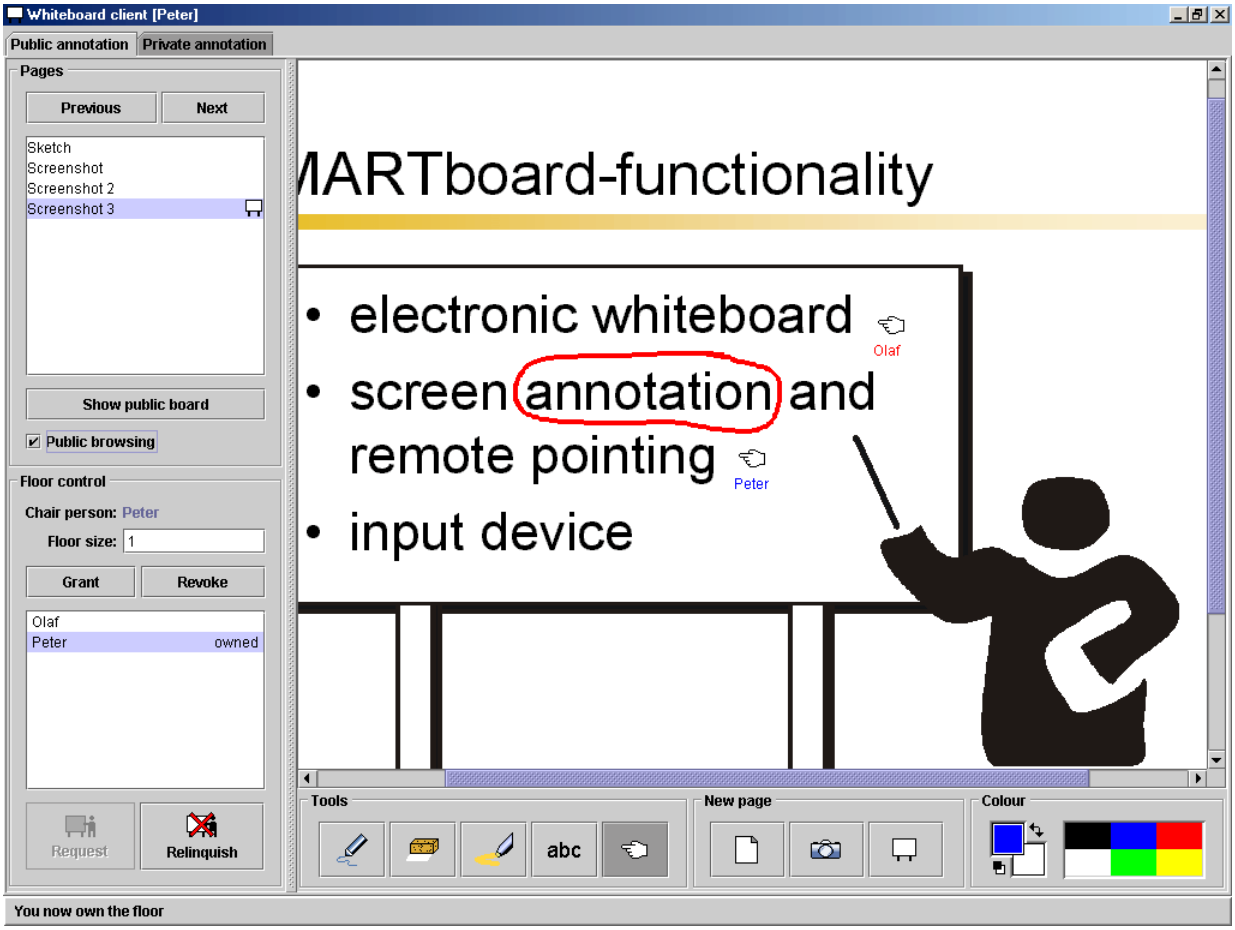


**Figure 7:** Snapshot of whiteboard with facility for public and private annotation

server or all clients. In any case, it stores them into the appropriate local database. A more detailed description of the architecture can be found in [51].

### 9.2 Example application 2: Animation and Simulation applets

As a third example, a special sort of animation and simulation applets was integrated. Such applets have been designed especially to facilitate teaching in the area of communication protocols where students very often are confronted with rather complex, highly dynamic and concurrent processes (e. g. [7]). Think for instance of a protocol for the electronic market with buyer and seller trying to install a reliable and secure connection where they can trust each other. It is not obvious, how parameters can prevent a man-in-the-middle attack. In a first step, a simulation applet enables a single learner to play with the protocol, try out different input sequences as well as correct and faulty protocol variants. Hence, she can watch the resulting behavior with respect to correctness and performance. Moreover, a teacher can demonstrate the whole process to students.

In a second step, collaborative usage of applets has been realized [3]. Depending on the protocol in question, simulated nodes are behaving according to a certain role. For instance, in the mutual authentication scenario, three roles are involved (besides a moderator): buyer, seller and attacker. The floor control as described above cares for offering a sub-floor for each of these roles, thus enabling participants to play together in a controlled manner (see fig. 8). Moreover, they can apply public and private annotation on snapshots of the applet at each point of time.

It should be noted, that these simulation applets have been constructed according to the MVC-scheme (Model, View, Control). This enables the afore mentioned possibility to partition applications. Whereas the model and control parts are residing on the server side, each client has only a view part running locally. After each simulation step, the server notifies clients about the new context information. Hence, the component for view can provide context awareness to participants by showing the current state of each simulated object.

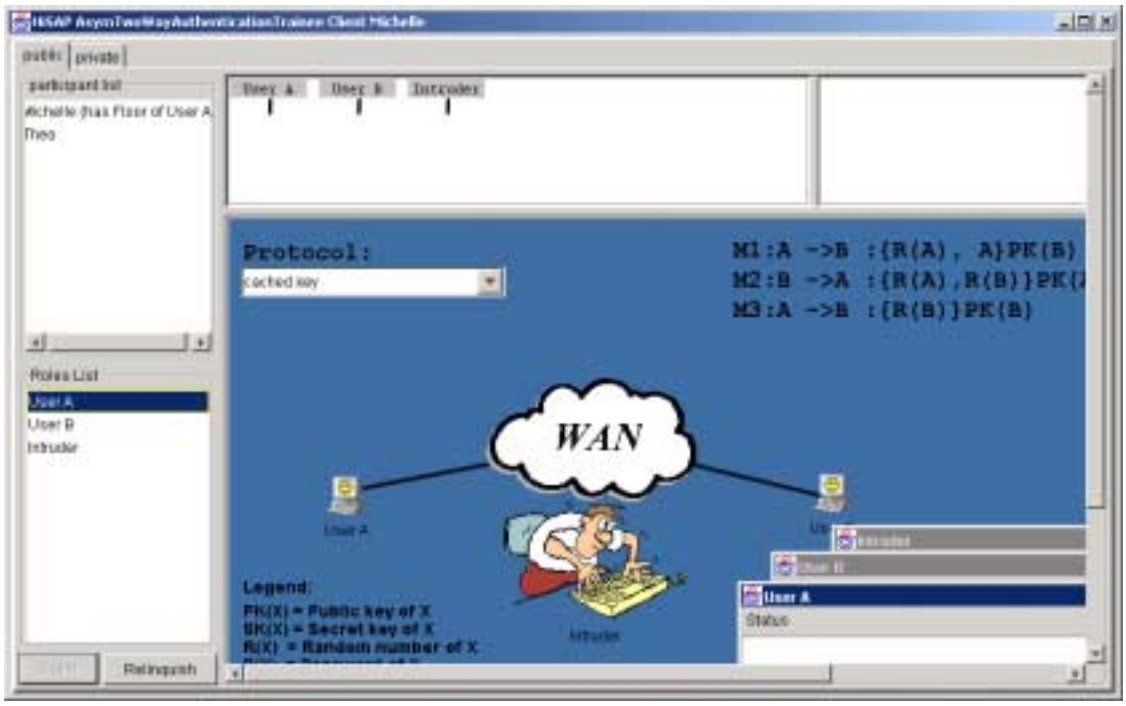


**Figure 8:** Shared usage of animation and simulation applet

## 10 Implementation aspects

To achieve platform independence, the system was implemented in the JAVA programming language. Existing tool kits and products were evaluated for stability reasons and reduced implementation effort.

### 10.1 Underlying communication mechanism

Regarding the logical connection between clients and server, we used JSDT (Java Shared Data Toolkit, [34]) Version 2.0 by Sun as underlying communication mechanism. It is consisting of a collection of Java objects and functions to support development of interactive and cooperative applications. Its main focus is on dissemination of messages and data to a designated group of receivers. To provide an interface of higher abstraction level, it is independent from and encapsulates the underlying transport mechanism in so-called communication objects. Among other features, JSDT supports the channel metaphor as described earlier and needed by the components of our framework.

The following communications objects are available with the distribution:

- tcp/ip sockets for communication via tcp (transmission control protocol for internet protocol) and udp (user datagram protocol),
- http (hypertext transfer protocol) tunelled sockets to cope with firewalls and
- lrmp (lightweight reliable multicast protocol).

The latter can be used to benefit from multicast facilities as offered in wireless networks.

Either the application or the user are choosing one of these communication objects at runtime or when starting the first connection. This does not exclude the usage of further objects but will be neccessary in special cases only, e. g. in the context of firewalls.

JSDT is offering mechanisms to structure a communication into three different basic components: session, channel and bytearray. The concept of a session is used for data exchange among communication participants. A session is started and managed by a server object. Via the address of this server (its ip address, port and the chosen communciation object), clients can participate in a session. The concept of channel enables data transfer to one special member or a group of them in a synchronous or asynchronous way. Bytearrays define a distributed storage area that is synchronized automatically.

Further components and methods are available in JSDT to manage and control the data transfer itself. Examples are tokens, manageable objects and client authentification. Tokens care for synchronizing actions belonging to the same session. They can be requested, released and passed among clients. Hence, they are an ideal concept to realize floor control mechanisms for applications and coordination of shared access to resources. By means of manageable objects, clients

can be privileged for special actions on this object. For example, creation, removal and entering a session, channel, byte-array or token can be privileged actions. Before performing such actions, clients have to authenticate.

It can be seen, that JSDT is offering sufficient concepts to realize the application sharing system as described above. It should be noted, that similar results can be obtained with GroupKit [16]. But this system is based on TCL/TK and focused on Unix operating systems.

### 10.2 Underlying database

Another design decision concerns the database to be used for archiving. From the requirements of section 2, the following evaluation criteria have been derived:

- Independent from platform.
- Available for free for non commercial usage.
- Sufficient stability.
- Minimal hardware requirements (regarding mobile and wearable devices)
- JDBC interface for conformance with standard [13].
- Short response times.
- Cope with large and with many records.
- Easy to install and to configure.

When applying these criteria to the systems MySQL 3.23.30 [35], [46], Ovrimos SQL Server 3.0 [10], [53] and InstantDB 3.26 [29], we obtained the results as shown in table 4 (+: valid +/-: partially -: does not hold ?: unknown). It can be seen, that all three systems provide comparable behaviour in almost all criteria with the exception of easy installation and configuration. Because these aspects are crucial in a teaching environment, we decided to integrate InstantDB 3.26 into the SASCIA prototype (see [58] for further details).

| | **MySQL 3.23.30** | **Ovrimos SQL Server 3.0** | **InstantDB 3.26** |
|---|---|---|---|
| Independent from platform | +/- | +/- | + |
| Available for free (non commercial usage) | + | +/- | +/- |
| Stability sufficient | + | ? | + |
| Minimal hardware requirements | + | + | + |
| JDBC interface | + | + | + |
| Short response times | + | + | +/- |
| Large | + | + | + |
| Many | + | + | + |
| Installation easy | - | - | + |
| Configuration easy | - | - | + |

**Table 4:** Comparison of existing databases

## 11 Measurement results

The prototypical implementation is based on a wireless network of Lucent [40], workstations running Sun Solaris, notebooks with some version of Microsoft Windows, a physical whiteboard called SMARTBoard from Smart Technologies, Inc. [61]. Further hardware to be included are a Digitizer Board [30] and tablet [64].

Before testing the system in class, a series of measurements has been performed. We estimated the most critical parameters to be the time for establishing connections between clients and servers, the time to start application frames and the time for storing and retrieving data to resp. from the database. Moreover, the multicast facility of JSDT was tested.

| | **Time for application start (on client / on server)** | **Time for establishment of client-server-connection** |
|---|---|---|
| Local case | 172 ms / 55 ms | 297 ms |
| Remote case | 312 ms / 55 ms | 398 ms |

**Table 5:** Performance of session management and underlying communication system

For the experiments, we used a server residing on a PC (AMD Athlon 900 MHz, 128 MB) and a client being placed on a notebook (Intel Pentium III 1000 MHz, 256 MB) connected by a 2 Mbit/s wireless LAN, a 10 Mbit/s connection between

a bridge and switch and another 100 Mbit/s connection via a second switch to the PC. As shown in table 5, connection establishment and start of application frames takes additional 100 to 140 more ms for the remote case in comparison to the one with client and server on the same node. For clients, times are less because more work has to be performed on the server and the server used was a rather slow one. As a whole, values are not the best ones but acceptable [62].

Regarding the performance of databases, different variants have been tested. The encapsulated database InstantDB is offering two modi, one with storing directly to disk ('without fastUpdate') and another one with keeping data in memory and writing them to disk subsequently ('fastUpdate'). Fig. 9 demonstrates that this factor does not provide good results. For better performance, a further buffer was introduced to receive data before passing them to the database. But even this variant does not perform rather well when compared to a solution solely relying on memory without any database at all. Nevertheless, values for the variant with buffer seem to be acceptable [58].

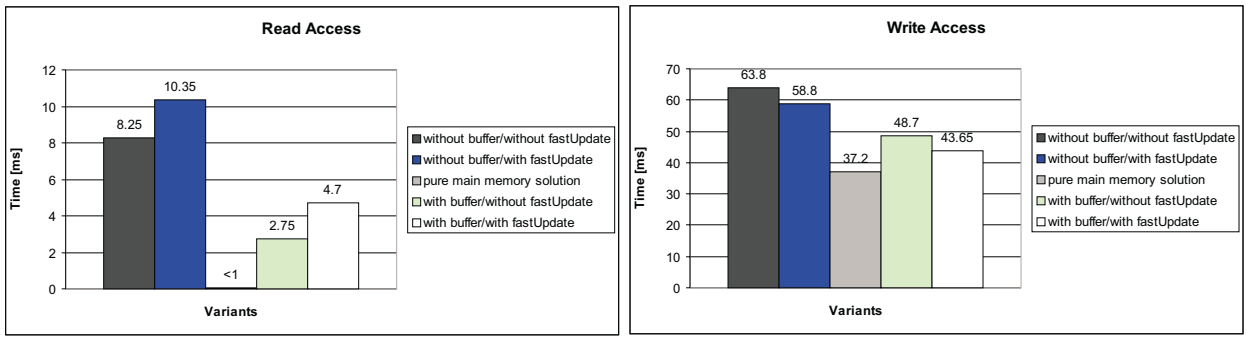


**Figure 9:** Access time for reading and writing

When comparing the communication behaviour of the system with two different transport media, tcp/ip and lrmp, it showed up that the variant with multicast encountered serious problems for a packet size above 8 KB whereas the unicast variant performed quite well (see [51] for more details).

## 12 Summary, conclusion

We presented some extensions to existing concepts of application sharing systems that cope with special requirements of collocated teaching. Context information is used for a couple of purposes (e. g. presenting a map to the teacher to facilitate identification of students, especially of those raising their hand). A configurable floor control is covering different kinds of applications and different session modi (moderated, voting, role-based sharing). Public and private data can be captured to make them available for subsequent replay.

The prototypical realization is containing a whiteboard for annotation, for shared presentation and collaborative usage of simulation applets to facilitate the learning of communication protocols. It is planned to integrate further applications like more simulation environments and real life engineering experiments. For such cases, context information about applications and observation of semantic consistency are getting the most important components. While it is rather easy to obtain context information about simulation objects, this does not hold for participants and application objects in real experiments (e. g. trains in a railway model or elements in a chemical reactor, see [8]). For instance, in the current version we rely on manual input of the geographical location or seat number of participants. It has to be investigated, how information can be captured automatically by means of sensors and cameras, how such data can be combined with the recorded history to achieve an almost complete picture of the whole session, its participants and all its application objects.

Further plans concern the extension of the system to integrate remote participants and small devices like PDAs. Moreover, an architecture based on peers [49] and a mechanism to notify absent persons about the treatment of interesting topics to enable them to join at the appropriate moment [18] will be examined. To facilitate the integration of arbitrary applications, it is planned to combine VNC [55] with our system.

As a whole, SASCIA will contribute to better interactivity among teachers and students as well as increase the effiency of teaching.

## Acknowledgments

The work of the first author is funded by the Margerete von Wrangell scholarship of Baden-Württemberg, Germany.

Our special gratitude is to the team of those students that have designed and implemented the framework and components as well as performed measurements: Xue Bai, Peter Oberparleiter, Andreas Schmid and Marcus Sommer. Moreover, we want to thank Eric Bullinger and Thomas Wagner for fruitful discussions about the application of our system to teaching in engineering areas.

## References

[1] Abowd, G. D.. Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment. IBM Systems Journal, Special issue on Pervasive Computing, Volume 38, Number 4, pp. 508-530, October 1999. Also available at http://www.cc.gatech.edu/fce/eclass/pubs/index.html

[2] Bauer, M., Kortuem, G., Segall, Z.. "Where Are You Pointing At?" A Study of Remote Collaboration in a Wearable Video-Conference System. Proceedings Third International Symposium on Wearable Computers (ISWC'99) , 18-19 October, 1999, San Francisco, California.

[3] Bai, X.. Gemeinsames Lernen von Kommunikationsprotokollen durch elektronisch unterstütztes Rollenspiel. Diplomarbeit Nr. 1930, Fakultät Informatik, Universität Stuttgart, October 2001

[4] R. Bentley, T. Horstmann, J. Trevor. The World Wide Web as Enabling Technology for CSCW: The Case of BSCW. Computer Supported Cooperative Work: The Journal of Collaborative Computing 6: 111-134, 1997

[5] Brand, O., Mahalek, W., Sturzebecher, D., Zitterbart, M.. MACS - A Modular Collaboration Environment. Proc. IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA '99), Nassau, Bahamas, October 1999

[6] Brotherton, J. A., Abowd, G. D., Truong, K. N.. Supporting Capture and Access Interfaces for Informal and Opportunistic Meetings. GVU Technical Report Number: GIT-GVU-99-06, http://www.gvu.gatech.edu/gvu/reports/1999/abstracts/99-06.html

[7] Burger, C., Rothermel, K.. A framework to support teaching in the area of distributed systems. acm Journal on Educational Resources in Computing (JERIC), Volume 1 (2001), available at http://www.acm.org/pubs/contents/journals/jeric/2001-1/

[8] Burger, C., Bullinger, E., Papakosta, S., Wagner, T.. Context awareness for application sharing in teaching environment. To appear in proceedings of SSGRR2002w.

[9] Davis, R. C., et al.. NotePals: Lightweight Note Sharing by the group, for the group. Proc. CHI'99, May 1999, pp. 338-345

[10] Dicken, H., Hajir, N.. Daten aus Hellas - Browsergesteuert: Ovrimos SQL Server 3.0. iX, (3):68-70, March 2001

[11] Eckert, A. et al. 1997. A Distance Learning System for Higher Education Based on Telecommunications and Multimedia. Proceedings ED-MEDIA/ED-TELECOM'97, Calgary, June 1997.

[12] Effelsberg, W.. CBT to "Rechnernetze". http://www-mm.informatik.uni-mannheim.de/veranstaltungen/ss2001/rechnernetze/

[13] Fisher, M.. JDBC Database Access. The Java Tutorial. http://java.sum.com-docs-books-tutorial-jdbc-index.html

[14] L. Garber. Polish Students Win Computer Society Design Competition. IEEE Computer, August 2001, Vol. 34, No. 8, pp. 67-71

[15] Groove. http://www.groove.net

[16] GroupKit. http://www.cpsc.ucalgary.ca/projects/grouplab/groupkit/gk5doc/

[17] M. Guggisberg, P. Fornaro, T. Gyalog and H. Burkhart, An Interdisciplinary Virtual Laboratory on Nanoscience , Electronic Notes in Future Generation Computer Systems, Vol. 1 (2001). Also available at http://www.nanoworld.unibas.ch/zope/nano/articles/proceedings_speedup

[18] Haaf, M.. Integration of a personal agent to observe SASCIA sessions and notify its user in a rule-based manner. Diploma thesis, Faculty of Computer Science, University of Stuttgart, to appear May 2002

[19] Hightower, J. Borriello, G.. Location Systems for Ubiquitous Computing. IEEE Computer, August 2001, Vol. 34, No. 8, pp. 57-66

[20] Hills, A.. "Wireless Andrew", IEEE Spectrum, Vol. 36, No. 6., June 1999

[21] Hilt, V., Geyer, W.. A Model for Collaborative Services in Distributed Learning Environments. Proc. International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services 1997 (IDMS'97), Darmstadt, Germany, R. Steinmetz, L. Wolf (Eds.), LNCS 1309, Springer Verlag, Berlin, Germany, pp. 364-375, 1997

[22] Hilt, V. et al.. A Generic Scheme for the Recording of Interactive Media Streams. Proc. International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services 1999 (IDMS'99), Toulouse, France, M. Diaz et.al. (Eds.), LNCS 1718, Springer Verlag, Berlin, Germany, pp. 291-304, 1999

[23] Hodes, T. et al.. Shared Remote Control of a Video Conferencing Application: Motivation, Design, and Implementation. Proceedings of SPIE Multimedia Computing and Networking, San Jose, CA, January 1999, pp.17-28. http://www.openmash.org/resources/index.html

[24] Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K., Schwehm, M.. Next Century Challenges: Nexus - An Open Global Infrastructure for Spatial-Aware Applications. Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), Seattle, Washington, USA, August 15-20, 1999, T. Imielinski, M. Steenstrup, (Eds.), ACM Press, 1999, pp. 249-255

[25] Hopp, A., Schulz, D., Burgard, W., Cremers, A. B., Fellner, D.. Virtual reality visualization of distributed tele-experiments. In Proc. of the 24th Annual Conference of the IEEE Industrial Electronics Society (IECON), 1998. Also available at http://www.informatik.uni-bonn.de/~schulz/

[26] Horz, H., Buchholz, A., Hofer, M.. Neue Lehr-/Lernformen durch Teleteaching? PIK 23. Jahrgang 3/00, pp. 129-136

[27] Hürst, W., Maass, G., Müller, R., Ottmann, T.. The 'Authoring on the Fly' System for Automatic Presentation Recording, Proceedings of ACM CHI 2001 Conference on Human Factors in Computing Systems,Seattle, WA, USA, March/April 2001

[28] IETF (The Internet Engineering Task Force): http://www.ietf.org/rfc.html
RFC 2327 Session description protocol.
RFC 2543 Session initiation protocol
RFC 2974 Session announcement protocol
RFC 2865 Remote Authentication Dial In User Service (Radius)

[29] InstantDB. instantdb.enhydra.org

[30] intelliBoard http://www.mcr-gmbh.com/wboard.htm

[31] ITU (International Telecommunication Union) Recommendation: ITU Electronic Bookshop
T.120 (07/96) - Data protocols for multimedia conferencing
T.126 - Multipoint still image and annotation protocol
T.127 - Multipoint Binary File Transfer Protocol
T.128 - Multipoint application sharing protocol

[32] JATEK 00 (Java Based Teleteaching Kit). http://www.jatek-gmbh.de/JaTeK.htm???

[33] JETS 2000. A Java-Enabled TeleCollaboration System. http://www.mcrlab.uottawa.ca/jets/

[34] Java (TM) Shared Data Toolkit Home Page (JSDT). http://java.sun.com/products/java-media/jsdt/

[35] Kirsch, C.. Datendüse - MySQL: Freie Datenbank. iX, (6):52-56, June 2000

[36] Kortuem, G., et al.. When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad-hoc Networks. 2001 International Conference on Peer-to-Peer Computing (P2P2001), Aug. 2001, Linkpings, Sweden http://www.cs.uoregon.edu/research/wearables/papers.html

[37] Krishnan, N.. The Jxta solution to P2P. JavaWorld, October 2001. http://www.javaworld.com/javaworld/jw-10-2001/jw-1019-jxta.html

[38] Leonhardt, U.. Supporting Location-Awareness in Open Distributed Systems. PhD thesis, University of London, 1998

[39] P. Ljungstrand. Context-awareness in distributed communication systems. Workshop "The What, Who, Where, When, Why and How of Context-Awareness", Computer-Human Interaction (CHI) 2000, The Hague, The Netherlands, 2000. Also available at http://www.viktoria.se/play/publications/2000.html

[40] Lucent Technologies. http://www.wavelan.com

[41] Lund, K. et al. 2000. Requirements Analysis and Design for a Flexible Learning-on-Demand System. Proc. Integrated Design & Process Technology, Dallas, June 2000

[42] Malpani, R., Rowe, L.A.. Floor Control for Large-Scale MBone Seminars. Proc. of The Fifth Annual ACM Intl. Multimedia Conf., November 1997. http://www-mash.cs.berkeley.edu/frame/research/publications/

[43] MBone. http://bzvd.urz.tu-dresden.de/mbone/software.html

[44] mlb http://www.informatik.uni-mannheim.de/informatik/pi4/projects/mlb/

[45] MSRES Microsoft Corporation NetMeeting Resource Kit, Chapter 3: Finding People. Microsoft Website http://www.microsoft.com/windows/netmeeting/Corp/reskit/Chapter3/

[46] MySQL. http://www.mysql.com

[47] NetMeeting. http://www.microsoft.com/windows/netmeeting/

[48] Myers, B. A.. "Using Hand-Held Devices and PCs Together," Communications of the ACM.Volume 44, Issue 11. November, 2001. pp. 34 - 41. http://www-2.cs.cmu.edu/~pebbles/index.html#papersandtalks

[49] Nguyen-Salamanis, K.-L.. Design and realization of a peer-to-peer architecture for application sharing. Diploma thesis, Faculty of Computer Science, University of Stuttgart, to appear May 2002

[50] Oberparleiter, P.. A generic floor control for shared usage of smartboards during lectures (in german). Pre-Diploma thesis no. 1807, Fakultät Informatik, Universität Stuttgart, May 2001, available at http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL_view.pl?id=STUD-1807&inst=VS&mod=&engl=

[51] Oberparleiter, P.. A framework to share teaching applications with a prototypical realisation of an electronic whiteboard (in german). Diploma thesis, Faculty of Computer Science, University of Stuttgart, to appear January 2002

[52] T. Ohmori, K. Maeno, S. Sakat, H. Fukuoka, K. Watabe. Distributed Cooperative Control for Sharing Applications Based on Multiparty and Multimedia Desktop Conferencing System: MERMAID. In International Conference on Distributed Computing Systems, pp. 538 - 546, Juni 1992

[53] Ovrimos SQL. http://www.ovrimos.gr

[54] Papakosta, S., Burger, C.. Generating Interactive Protocol Simulations and Visualizations for Learning Environments. In: CaberNet: 4th Plenary Workshop. Vol. 2001

[55] T. Richardson, Q. Stafford-Fraser, K. R. Wood, A. Hopper, "Virtual Network Computing", IEEE Internet Computing, Vol.2 No.1, Jan/Feb 1998, pp33-38. http://www.uk.research.att.com/vnc/docs.html

[56] Scheele, N.. Enabling Interactive Education with Handheld Devices. Diplomarbeit, Fakultät für Mathematik und Informatik, Universität Mannheim, Mai 2001.

[57] Schlichter, J. Material to “Computergestützte Gruppenarbeit”. http://www11.in.tum.de/lehre/lectures/ss2000/cscw/index_4.html

[58] Schmid, A.. Prototypical realisation of recording for teaching purposes (in german). Pre-Diploma thesis no. 1816, Faculty of Computer Science, University of Stuttgart, August 2001, available at http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL_view.pl?id=STUD-1816&inst=VS&mod=&engl=

[59] A. Schmidt, M. Beigl, H.-W. Gellersen. There is more to Context than Location. Environment Sensing Technologies for Adaptive Mobile User Interfaces. International Workshop on Interactive Applications of Mobile Computing (IMC 98), November 1998, Rostock, Germany.

[60] Schuett, A., et al.. A Distributed Recording System for High Quality MBone Archives. Proceedings of the First International Workshop on Networked Group Communication, (NGC ’99), Pisa, Italy, November 1999. http://www.openmash.org/resources/index.html

[61] SMARTBoard. http://www.smarttech.com/smartboard/

[62] Sommer, M.. Prototypical realisation of session management for teaching purposes (in german). Pre-Diploma thesis no. 1817, Faculty of Computer Science, University of Stuttgart, August 2001, available at http://www.informatik.uni-stuttgart.de/cgi-bin/NCSTRL_view.pl?id=STUD-1817&inst=VS&mod=&engl=

[63] Vogel, J. et al.. A Generic Late Join Service for Distributed Interactive Media Proc. ACM Multimedia 2000, Los Angeles, California, USA, 2000

[64] WACOM

[65] WISS – Wireless Infrastructure for Students and Staff, University of Rostock, http://wiss.informatik.uni-rostock.de/local/

[66] Wolf, L.. DUKATH - The wireless network of the University of Karlsruhe (in german). PIK - Praxis der Informationsverarbeitung und Kommunikation, 23. Jahrgang 4/2000

All Web pages were visited December 2001 for the last time.