sTeam - Designing an Integrative Infrastructure for Web-Based Computer-Supported Cooperative Learning

Thorsten Hampel
Computers and Society
Heinz Nixdorf Institut
Fürstenallee 11, 33102 Paderborn, Germany
hampel@upb.de

Reinhard Keil-Slawik
Computers and Society
Heinz Nixdorf Institut
Fürstenallee 11, 33102 Paderborn, Germany
rks@upb.de

ABSTRACT

Learning is a socially embedded design process. But most of today's hypermedia systems fail to properly support both the design-related and the social aspects of learning. Authoring and web-publishing systems aim to support the authors' design process. Consequently, learners' activities are confined to selecting and reading. Based on some fundamental reflections on the role of technology in learning processes, we conclude that top priority must be given to the construction of infrastructures that support cooperative learning processes if we are to properly harness technology's potential.

We present a learner-centered – wholly web-based – approach for structuring information in teams (sTeam). The key concept in sTeam is virtual space. It draws together cooperation and communication, at the same time embodying the common external memory of a (virtual) learning group. The focus is not, then, on interactive systems for individual accessing of knowledge bases, but rather on the cooperative management and structuring of distributed knowledge bases.

Keywords

cooperative learning, web-based learning and teaching, learner-centered approaches, cooperation support, sTeam – structuring information in a team

1. INTRODUCTION

Mainstream discussions on the role of technology in teaching and learning center on two basic paradigms. Hypermedia systems aim to support individual learning processes, special emphasis being placed on new didactic qualities, attributed to the interactive combination of different media types such as text, graphics, audio, video, etc. The second paradigm embodies the notion of "delivering education", networking technology being used to distribute and access study materials as well as to establish communication channels between students and teachers. Although the idea of utilizing net services has strong collaborative connotations, the main argument in favor of networking is the temporal and spatial independence it offers and, consequently, independence from close collaboration with others.

Students can now learn individually at their own pace and their own chosen location.

WWW10, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

However, many studies evaluating the role of technology in learning processes have yielded conflicting results, indicating that there is no general, clear-cut connection between the effort required to produce high-quality multimedia educational materials and improvements in the learning process. The problem is to attribute certain benefits to a single variable – say, the specific technology used. Mostly, the results are a combination of different variables such as didactic style, educational strategy, technology deployment, appropriate selection of content and the personal qualities of teachers and students. (see [11], [13]) Thus, the widely accepted idea that learning can be improved by the individualization of learning processes using hypermedia and networking technology is not generally borne out by scientific research.

Instead of using technology to individualize learning processes, we have chosen the opposite approach – namely, using technology to support cooperative learning processes within the framework of traditional university education. Rather than trying to do away with the need for physical presence in the learning situation, we seek to support social processes in which students and teachers are physically present for a specific time at different places. Quite apart from practical considerations, theoretical research reveals that, ultimately, the social embedment of learning is a crucial factor in success.

Furthermore, our approach is based on the general assumption that technology can only solve technological problems, didactic problems requiring didactic solutions. Teaching activities cannot, then, be replaced by technological functions, and the embedding of technology into teaching and learning activities must be carefully studied.

To design a learning-supportive infrastructure, it is therefore necessary to identify which of the problems connected with the learning process are technological problems and which are the main technological functions for problem solution. In the following sections, we briefly sketch the theoretical framework enabling us to identify general technological functions that need to be implemented. However, empirical investigations are required to properly determine the specific implementation of these functions because it is the fine tuning of all variables, of technical means and didactic concepts, innovative ideas and available resources that ultimately determines how to set up an appropriate infrastructure. The sTeam approach focuses on the collaborative exploration and maintenance by students of a complex hypermedia knowledge base – a concept we call self-administration.

2. MEDIA AS EXTERNAL MEMORIES

Theoretical frameworks fail to provide a sufficient basis for the design of cooperative learning-supportive infrastructures. They do, however, help to define a general strategy for their development, especially when the overall development cycle spans several years. Such a framework must provide some guidance on how to specify goals, set priorities and formulate hypotheses to be validated in actual use, as well as helping to distinguish between technical and nontechnical problems and to analyze how they are interconnected.

Two general ideas might be useful to illustrate briefly the approach we have adopted. The first is based on an argument put forward by J.J. Gibson [7]. Gibson came to the conclusion that it is human information processing that allows us to distinguish between what is imagined and what is real. When we view a phenomenon from different perspectives, or when we modify what we perceive and interpret the respective changes, we gather information through our bodily activities. We cannot, as Gibson claims, gather information about a purely imagined phenomenon because the result of any purely mental modification or change in perspective is completely controlled by our mind. Thus, no difference between what is imagined and what is real can be detected. To carry this argument a step further, we could say that nothing can be learned about a certain phenomenon if we are unable to generate and interpret different patterns of perceivable responses. Learning, then, requires feedback through personal observation or social response; it is generally a combination of both. Hence, in order to gain information about a certain phenomenon, we must create an environment that can be perceived and manipulated (i.e. an experimental setting or symbolic description).

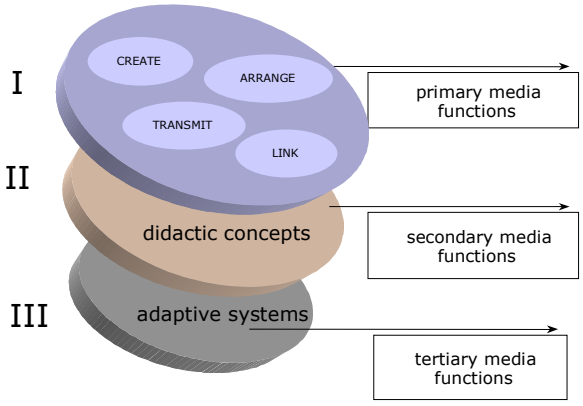

Figure 1: media functions

The second idea draws on the fact that the biological (genetic) basis for cognition has not changed for at least the last several thousand years of human evolution. Our cultural accomplishments do not reflect an improvement in our biological capacity for intelligent behavior, but rather our ability to create ever better means of expression to support productive thinking and to create a socio-symbolic universe that enables us to base our learning activities on the ideas and accomplishments of our ancestors (see [4]). In terms, then, of both the evolution of culture in general and the evolution of the individual's mental capacity, it can be said that learning is an inherently social activity based on the use of physical artifacts, semiotic artifacts such as drawings, calculations, verbal descriptions or formal notations also playing a crucial role. Of course, symbolic artifacts are of paramount importance: "...the use of signs appears to be indispensable to the use of reason." (see [16], p.135). However, the use of symbolic artifacts requires technology, so the crucial question is: Which specific technological functions adequately support learning?

To answer this question, we begin by distinguishing between the terms memory and storage. Human memory is dynamic, irreversible (one cannot deliberately forget or erase something), selective and subject to individual experiences and views. A storage is a repository generally used for depositing physical items for later retrieval. The storing and retrieving functions should not modify the items being stored (authenticity). The human memory, then, processes information, whereas computers store data.

Apart from the human brain and its memory functions, we talk of external memories in cases where the items in a storage are continuously interpreted and modified as a result of some individual or socio-intellectual process. A university library, for instance, is not just a storage (a repository for books and magazines), but a living body of knowledge available to a scientific community and continuously modified to reflect the current state of knowledge. To this extent, a library serves as an external memory.

The same holds for course material in an education context. During a course, students learn to explore the material, associating meaning with formulas and data, to assess and discuss the relevance of examples and to add their own material – which may be individually produced, copied from some other source or the result of collaborative work. The way they use the external memory reflects their current state of knowledge, changing as they become more knowledgeable. By the end of a course, participants will have acquired their own personal understanding, but they will also share a common understanding of the material commensurate with their active involvement in its collaborative use and redesign; a somewhat arbitrary repository of semiotic artifacts has thus evolved into an external memory. Items in a storage can be created, moved, arranged or deleted to express people's understanding and support their mental activities. However, the actual physical state of an external memory can only embody this knowledge to a limited extent because it fails to provide a detailed account of the knowledge's evolution and why it evolved in that particular way. This holds for textbooks, formal notations and software.

By way of conclusion, it may be said that technology supports learning processes by providing appropriate means to create and maintain storage that functions as an external memory.

3. ENHANCING LEARNING VS. REPLACING TEACHING

The concept of external memories constitutes an initial attempt to separate the technical (product) from the human (process). However, to design suitable external memories, we have to go a step further. We do so by distinguishing three classes of media functions and showing their implications for the interplay of technology and learning.

3.1 Primary media functions

Media are generally viewed in terms of communication, based in most cases on the classical transmitter/receiver or producer/ consumer model. The integration of different media types such as text, image and sound (multimedia) made possible by computers, together with the fusion of transmission channels and services (Internet), means that we must extend the concept of media. Media are no longer simply means of communication; they are – and have always been – both means of expression/ cognition and means of organization. Without media, our cultural achievements are inconceivable. Complex social processes based on the division of labor are just as reliant on media as are science and education. Media, for their part, require technology to create an objective, mostly symbolic world. To determine the potential and possibilities offered by the new media, we must first remind ourselves of what constitutes, in technical terms, the benefits they provide. Here, four basic categories can be defined for technical functions, whose implementation can help better support cognitive processes:

- CREATING: Media serve to create a perceptual space allowing conception and reality to be correlated by action and the relevant conclusions to be drawn. Scientific instruments, experimental apparatus, models and simulation programs are just as much instances of artifacts that perform this function as are symbolic descriptions, diagrams, images, formalisms and visualizations of large datasets.
- ARRANGING: To attain new insights, it is necessary to correlate different documents. Problem solving and learning invariably involve identifying differences and concurrences, combining different types of descriptions and representations or weighing statements from different sources against one another. To support these processes, the artifacts to be correlated must be brought into the field of perception, simultaneously if possible. Here, logical connections should be represented, if possible, by spatial connections as well, to enable them to be swiftly identified and processed.
- TRANSMITTING: Cultural learning achievements are social processes. Reducing learning to the individual processing of a document, for example, means failing to appreciate that this document is already being used, assessed and passed on in a specific social teaching/learning situation. Furthermore, contemporary university learning situations are often such that course materials are processed, exchanged or compared in groups. In such a context, learning is a disparate process taking place in a variety of forms in different places and at different times. The required material must be uniformly available at all learning locations, i.e. it must be transmitted there.
- LINKING: Arrangements embodying an important connection should be preserved after the act of arranging so that they do not need to be regenerated at a later point in order to continue working with them. Suitable links, for example, enable users – ideally in a single step – to refer to all documents that are of relevance for the respective meaning context. Equally important here is the fact that the meaning context may cover only part of the material and documents. For instance, an author may refer to the works of other authors by citing or paraphrasing them, linking together certain statements by these authors to form a new line of reasoning. The relevant statements, quotes and paraphrases are fused with the author's own statements to form a new physically coherent textual entity, e.g. an anthology or an article. Besides a physical coupling of this sort, such linking can also be done by grouping, i.e. the physically unconnected objects are brought together at a specific place and – if necessary – structured, i.e. arranged according to one or more criteria. And it is also possible, instead of linking the artifacts themselves, to create references to them and then link them physically via these references (hypertext).

The rationalization potential of the new media lies in the implementation and integration of these primary media functions, the concern being to achieve more effective handling of the physical artifacts in terms of the four basic areas of functionality. Here, the new media offer a wealth of new forms for effectively generating, transmitting, arranging and linking semiotic artifacts, ranging from the integration of different types of media to net-based services and search facilities.

To this extent, learning-supportive infrastructures, like the Paderborn DISCO (Digital Infrastructure for Computer-Supported Cooperative Learning) [12], make use of all materials at any location where learning takes place.

3.2 Secondary Media Functions

Primary functions are essential means for creating and maintaining external memories to the extent that they should allow students and teachers to create and arrange any sort of educational material in any way, without enforcing specific teaching methods or didactic concepts. This allows a high degree of flexibility, but it may not provide adequate support. A textbook or CD-ROM is not just an arbitrary physical arrangement of different media types such as text, graphics or video; it constitutes specifically selected and designed material, taking the form of, say, introductions, examples, exercises or suggestions for further reading. The arrangement and selection of the material follows didactic principles, and in many cases it may be useful to suggest a certain sequence of learning activities (select and arrange materials) or guide learners through the material along a pre-specified path. We call such technological functions that support learners or enforce specific sequences of activities according to underlying didactic principles secondary media functions of the external memory, because these functions embody specific (empirical) knowledge about learning processes and their context. We group these functions into three categories:

- STRUCTURE: Providing operations to create and maintain specific arrangements of tools and documents such as learning units, which may be defined as a compound document containing a specification of the learning goal, an introductory text, some examples, a set of interactive tests and further references.
- INSTRUCT: Developing means for instructional design, especially tutoring and feedback mechanisms for self-guided learning.
- COOPERATE: Creating environments that support specific methods for cooperative learning and problem solving, such as brainstorming, role games or future workshops.

The implementation of secondary functions requires careful analysis of the specific knowledge to be taught and should be based on theoretical principles and empirical investigations to justify the additional resources needed to design the tools and materials. Since secondary functions are tied to a specific didactic

approach, they have to be re-implemented for different learning environments.

3.3 Tertiary Media Functions

In the literature, there are instances of attempts to implement tertiary functions, although such approaches have not yet yielded very promising results. Tertiary functions embody generic knowledge of problem solving and human understanding. The idea is to build smart or intelligent systems that are capable of understanding users or learning from their behavior, enabling the systems to adapt to the learners' individual needs. Tertiary functions are:

- ADAPTATION: Designing mechanisms that analyze the learner's input and change the system's behavior in accordance with this ongoing analysis.
- INTELLIGENT TUTORING: Developing systems that are able to cope with unanticipated or wrong input by users in an intelligent manner. The system must embody models of the learner's knowledge, the problem-solving knowledge of domain experts and knowledge about teachers' intervention strategies.
- NATURAL-LANGUAGE PROCESSING: Enhancing the system's ability to "understand" the input of learners by incorporating natural-language-processing mechanisms.

Tertiary functions represent the most ambitious goals in designing learning environments, and it seems doubtful whether this approach will ever lead to practical solutions. This is why we made it our highest priority to implement the primary media functions, which allow collaborative solutions. Secondary functions are often regarded as playing a key role for a new generation of learning environments, such as computer-based training (CBT) or computer-aided instruction (CAI). However, implementing secondary functions generally culminates in supporting individual learning and so often runs counter to attempts to establish infrastructures that support collaborative learning.

3.4 Product and Process

In a formal system, each operation performed by a machine is determined by the form and arrangement of symbols in the input sequence, the machine storage being independent of what these symbols represent. Every software system, then, embodies a syntactic machine (product). In teaching and learning, however, signs and symbols have to be interpreted in terms of personal experience and knowledge, individual preferences and interests. The respective consequences become apparent through human actions (process). This sort of knowledge is therefore often described as tacit or implicit. Implementing different classes of media functions forces us to make more and more tacit/implicit knowledge explicit. In the case of tertiary media functions – and to some extent secondary media functions – the kind of knowledge involved is typically that possessed by teachers. The learner can proceed individually because the teacher's functions are increasingly replaced by a machine that the learner can use anywhere, any time. Incidentally, the same holds true for traditional print media. The closer we come to the technical implementation of tertiary media functions, the greater the risk we run of trying to build a technical solution for a nontechnical problem – a hugely expensive undertaking.

4. THE PADERBORN DEVELOPMENT

The more we succeeded in solving the problems originally connected with the establishment and use of learning-supportive infrastructures in Paderborn, the more apparent became the direction the next development steps should take. Our orientation to the constructive use of media and the social embedding of learning increasingly revealed certain discontinuities in the use of media. Our distributed document management approach is a marked improvement on classical Web Publishing, but it still has its shortcomings: the volume of the teaching material and the access authorizations are determined by the teachers.

This means that the learner's scope is invariably confined to the content and access structures laid down by the respective teacher. The extent to which learners can arrange and structure the material to suit their own needs is therefore limited. They are not, then, able to establish their own learning environment, in which they could, for example, link documents from different courses or even different universities to match their own learning path and knowledge level.

It is, of course, possible in principle to use all the material freely available in the Net in different contexts. However, to effectively utilize and manage such material, activities like copying, transforming or indexing are needed that can soon involve a great deal of effort. Also, such individually structured knowledge bases are an obstacle to cooperative processing and generation. It is the discontinuities in the use of media and shortcomings like this that we aim to avoid by using the sTeam approach, a sort of learning lab for studying new forms of distributed knowledge organization, implementing them to meet everyday practical requirements and providing the necessary tools for this purpose on an open-source basis. To this extent, sTeam is not a finished product, i.e. a learning tool, but is itself the object of a learning process that needs to be further developed cooperatively.

5. sTeam – OUR PHILOSOPHY

In its basic sense, distributed knowledge organization means the cooperative generation, management and maintenance of artifacts embodying knowledge. Artifacts represented by documents, graphics, notes and comments, and their linking by learners, thus form the basis of any method for supporting learning processes using suitable environments and tools. The temporal component of an interaction or communication between learners can be seen as another essential factor as well as the spatial differentiation of cooperation. Existing environments and approaches can in most cases be assigned either to the class of synchronous, i.e. temporally concurrent, systems (supporting one session), or to the class of asynchronous systems centering on a common location (see [5]). The sTeam approach focuses on integrating the learners' temporal and spatial differentiations. Asynchronous mechanisms for handling multimedia learning components or hypertexts, such as are frequently familiar from document management, are combined with strongly synchronous approaches from the area of session support. Such a synthesis allows new, less familiar, hybrid forms of asynchronous and synchronous cooperation between learners.

Of the cooperation-supporting environments, Multi-User Object-Oriented Environments (MOOs) and Multi-User Domains (MUDs) (see [3]) are basic architectures that have proved well

suited for bringing learners together in common learning and working areas.[1]

Rooms are a key concept in all these approaches (see [10]). They are structured entities that can be dynamically linked together. The room structure enables positions to be defined for objects, representing people, documents, tools and services, that allow a knowledge space to be cooperatively established and, at the same time, responsibilities, rights or competences to be structurally mapped. To this extent, rooms constitute a key metaphor of virtual learning communities. Irrespective of the chosen representation of the room structure (two-dimensional, three-dimensional, as a tree structure or a textual description) the room is viewed as the meeting-point, "playground" and "lifeline" of a virtual community.

The sTeam system combines the idea of a room-based virtual world with the basic functionalities of document management. Rooms function not only as a social meeting-point and center of a virtual learning community but also as a collective external memory, providing the primary media functions as a basis for cooperative learning.

In the different virtual rooms, it is now possible to observe which cooperation partner is currently dealing with which documents and knowledge sources, provided this is explicitly permitted by the students. One use of such mechanisms is to create new forms of cooperative learning. During a seminar, for example, all the participants can be assembled in specific rooms and a "virtual ballot" conducted to evaluate specific documents. Another application is to set up a shared whiteboard that allows not only the students currently present in the room but also those hooked up externally to jointly process objects or to transfer elements of their desktop, graphics, references or entire documents to this synchronous work area, i.e. a work area that is simultaneously visible to all participants at a virtual location, using drag and drop functions. For instance, dragging a reference from a browser directly generates a link object in the room corresponding to the whiteboard. In the following section, we focus on the technical architecture of the sTeam environment; for a description of typical learning scenarios of the sTeam system, refer to [9].

6. THE TECHNICAL ARCHITECTURE

To support the primary media functions within a cooperative learning environment, a specific web-based architecture is needed. It proved possible to continuously further develop such an architecture, through a series of different prototypes, culminating in an open, modern platform. The presented architecture reflects the current status of the Paderborn sTeam development and pursues a rigorous open-source approach.

The sTeam architecture (cf. Figure 2) consists of the sTeam server, an external web server connected with the sTeam server and various clients. A standard Internet browser can also be used as a sTeam client to access most of the asynchronous functionality. Specific functionalities such as synchronous communication (chat) or cooperative applications are provided via Java clients. In early sTeam prototypes, the web server was integrated in the sTeam server proper. In the latest version, an external web server was coupled with the sTeam server. Here, the sTeam server acts as a sort of virtual file system for the web server. This type of architecture allows web accesses (requests) to be handled by the web server. The freely available Roxen web server[2] currently in use is a modern web server supporting all essential features, e.g. different security facilities, Javascript and server-site Java.

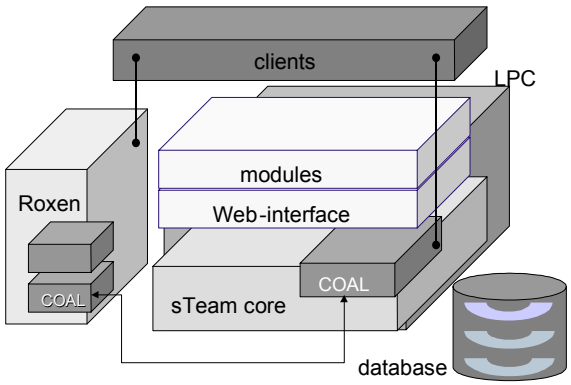

Figure 2: The sTeam architecture

The sTeam server proper comprises the sTeam core server, a number of extension packages and an external SQL database used as an object repository. Extension packages can be easily integrated into the sTeam server. For instance, there is currently an initial extension package (sTeam module) providing a web front-end for access to sTeam functionality via a standard Internet browser.

The architecture of the actual sTeam server was tailored to the needs of supporting cooperation processes. Human interaction in virtual space, awareness mechanisms and in particular the cooperative use of teaching material such as documents or arbitrary objects calls for a specialized management of objects, users and different events.

In terms of its realization, the sTeam server is based on an implementation of the classes and object structure in the interpreter language LPC[3]. The freely available – subject to GNU Public Licence (GPL) – and widely used LPC Interpreter PIKE[4] serves as the runtime environment. Besides offering good performance, PIKE features a broad code basis and a large number of existing libraries that can be implemented in the programming language C. For instance, the PIKE database interface for coupling SQL databases and an HTML parser[5] were used.

The sTeam system is completely object-oriented. Internally, the sTeam environment maps both users and documents – and rooms – as specific variants of objects. These have particular attributes

[1] Computer Science graduates Richard Bartle and Roy Trubshaw wrote the world's first Multi-Dungeon (MUD) game while they were still students at the University of Essex, U.K., in the late 1970s.

[2] cf. http://www.roxen.com

[3] LPC is a C-style interpreted programming language developed by Lars Pensjö for LP-MUD.

[4] cf. http://pike-community.org/

[5] cf. http://www.gingerall.com

or content, e.g. the document types. Interaction between objects takes place via mutual method calls or events. This is an extension of the well-known concepts from the MUDs family, or more specifically their object-oriented relations, the MOOs ("Mud Object Oriented").

6.1 sTeam Class Structure

To illustrate the specific features of the server architecture, the classes and object structure are shown in simplified form, along with their basic object types, e.g. "object", "container", "user", "group" and "document" (cf. Figure 3). In a second step, the event system as well as the sTeam server's access rights model is sketched.

The sTeam class structure consists of the basic components "object", "container", "user", "group" and "document". All classes are derived from the basic class "object". It is therefore imperative that an sTeam object be derived within the environment of "object" to maintain elementary system security.

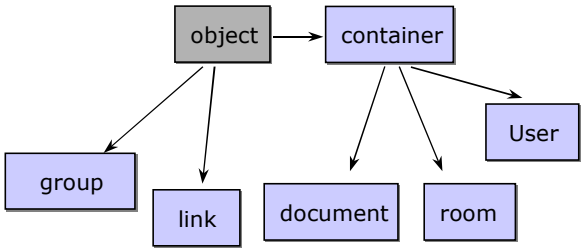

Figure 3: The sTeam class structure

Object: An "object" constitutes the basis of every element of the sTeam environment. Essential features here are object persistence, an unequivocal Object ID, available methods, events that can be linked with objects, and a number of possible attributes (these can be specified or re-registered when instantiating objects). Objects are persistently administered in the sTeam database. The – for the server – unequivocal object ID serves as a primary key to definite identification.

Objects within the sTeam environment have an unambiguous environment, which we shall henceforth simply term "environment". This is intended to create an object structure, every object existing at precisely one point within the sTeam system. (Such a structure forms a tree and has its root in the sTeam-root-room).

Container: One of the simplest variants (derivations) of an object is doubtless the sTeam container. It is used for the logical encapsulation of objects. In accordance with the object-oriented philosophy, a container can hold other objects. To this extent, the internal object representation is implemented analogously to a semantic grouping of materials (documents, graphics) in containers. Similarly, a user's rucksack, in which materials can be carried around and exchanged with other users, is realized by the object representation as a container. Direct results of this feature are the fairly universal characteristics of containers within the sTeam environment, such as the grouping of arbitrary objects. This basic MUD feature enables a room to be implemented as a specific variant of a container. A room groups materials, documents, graphics and tools, i.e. all sTeam objects, as well as users. The only other feature are special methods for communicating within the room. Connections between rooms are implemented as sTeam objects, for whose use special rights are needed. Using an exit or entering another room, then, presupposes the authorization to use – "execute" – a room object.

User: There exists precisely one user object for each user within the sTeam environment. This user object is located within a specific room, depending on the "position" of the user. Communication between the real user, who has logged into the sTeam system via a client, and the user object is implemented by means of a communication object attached to the user object. (Since users can be connected to the sTeam system via several clients simultaneously, they can be in possession of several communication objects at the same time.) As can be seen in the class structure, the user object is inherited from the class container. It extends the class container with different attributes and methods. And, as a feature derived from the container's class, it may also contain different objects. (This means that users' rucksacks – their "inventories" – enable them to take "objects" (to use MUD terminology) with them.

Special user attributes are, for instance, a user's name, access password, etc. These are administered as attributes in the corresponding user object. As with MUDs, the chosen user name should be unequivocal, serving as it does to access, to address the user object within the sTeam environment. Communication between users within a room is realized by means of events. As in the case of an MUD, a user's chat message (in an MUD, the "say" command) triggers an event within a room ("user X says: *text*"), which is processed by the recipients of the event. Various awareness mechanisms – in the simplest case, a list of the current visitors to the room – notify users about who is taking part in a discussion or cooperative session.

Group: A group encapsulates a number of users. The class "group" is directly derived from the class "object" and has a similar status to that of a user. User groups may contain other groups and other users.

This makes it possible to build a hierarchy of groups and users.

Document: All interactions within the sTeam environment are based essentially on generating, manipulating and exchanging sTeam objects. Like the primary media functions, objects serve as a basis for learners' potential artifacts. Unlike a conventional object, a document is characterized by the fact that it has content. This comprises the data of any file stored in the database, e.g. a graphic or word-processing document, or the content of a web document. Different document types are distinguished by a MIME type stored within an attribute – no specific document objects are needed, then, for different types of learning materials or application objects introduced to the sTeam system. The only exception here is the management of hypertext documents. The object type "document" is derived from a container's basic class, thus being able to encapsulate different objects. Such a mechanism enables, say, potential user annotations or comments on documents to be to be stored simply and neatly – they are stored in the environment of the respective object. Technologies such as version management or multilinguality of documents will in future be realizable using mechanisms of this sort.

Strictly separating an object, its attributes and its actual content in this way allows flexibility in the handling of document content.

Current sTeam system prototypes store the document content in the connected relational database. Another conceivable way of storing document content, keeping the effort involved at a reasonable level, would be to use special multimedia databases or even simple file systems.

Separating the document object itself from its content also allows documents to be simply and efficiently managed within sTeam rooms. Only actual accessing of document content by the user (e.g. through an upload or download) causes the respective document content to be loaded from the database.

Special rules apply to the handling of various types of hypertext documents such as HTML or XML[6]. These can contain links to external documents (URLs) in the document (i.e. in the object content). To avoid inconsistencies when inserting such documents in the sTeam environment or when subsequently shifting documents, links are to be handled separately. In the current implementation, a parsing of the HTML documents takes place and links are extracted and replaced by links that are unequivocal in the sTeam environment. (This replacement is carried out, when inserting document content into or reading it out from the database, by overwriting the corresponding functions in the object.)

6.2 Factories – Selective Generation of sTeam Objects

If new instances of objects are generated, this is done using so-called factories available for each object type. Every class has an assigned object with a function for providing instances of the class. The idea of having a definite location for generating new objects has a number of advantages. For one, objects can be updated at runtime – a feature that few systems offer and one which underlines sTeam's claim to be a flexible, user-extendable system. Here, factories administer the instances of objects at runtime, updating them, among other things. This enables attributes or methods to be added to existing classes without having to restart the sTeam server. Living instances of the relevant object type are updated.

A second advantage is that attributes are registered to objects within the factory. Such a mechanism enables clients to register new attributes to existing objects or even new object types and implement specific required features without modifying the sTeam core server. For instance, if a specific client needs an attribute of the local distance of people from a table – perhaps to represent some sort of seating arrangement, the client in question can register this new attribute at the factory of the class room and thenceforth use it.

Finally, the idea of special objects that generate instances of a class allows security to be extended to the object and class structure of the sTeam server. Only authorized users or user groups are able to generate instances of a class and thus permanently modify the system. No security check is made, then, at the level of the user interface, sTeam having a thoroughly uniform security concept, right down to the generation of individual objects. Factories are thus basic prerequisites for building a cooperative application at runtime and for extension of the environment by the users themselves. The idea of self-administration, i.e. granting system users the right to administer, restructure and extend the environment, is based originally on this sort of protected object instantiation.

6.3 The sTeam Event System

As with MUDs, the sTeam server is event-driven. Events play a prominent role in the design of the sTeam system's overall architecture. Nearly all communication between objects within the sTeam system is done through events, e.g. "a user addresses the room". Inside the sTeam core server, such events are processed in the order they occur and forwarded to affected objects.

The sTeam environment is based on a *flexible* event system. Events are triggered by arbitrary objects and can be processed by other objects. A whole series of events have already been defined, and adding new events is an easy matter. Events may have a limited range, e.g. be transmitted to only a limited number of objects, the subscribers to the event (e.g. all users in the room) or globally address all objects in the environment. All the sTeam server's internal structures are event-driven. For instance, the module security, sTeam's key security system, subscribes to all global events and, after a required security check, forwards them to the relevant objects. sTeam's whole authorization system is based on this mechanism.

The ability to generate new tools or document types shows just how powerful sTeam's event system is. Cooperative applications such as a shared whiteboard (a cooperative synchronous drawing area on which the users of a room can make drawings or diagrams or deposit documents) can be easily integrated into the sTeam environment using a powerful event system of this sort. To do so, a new shared whiteboard object is generated in the room (e.g. derived from a container) which – functioning, so to speak, as the common external memory of the room's users – manages the cooperatively administered drawings and documents. At the same time, special methods are provided enabling the data to be accessed inside the shared whiteboard container. If a user now accesses the database via a special client that suitably edits the whiteboard content (e.g. by drawing a new element in the drawing area), these state changes are passed on via events to all subscribers, i.e. those clients (objects) participating in the whiteboard session. Essential characteristics of cooperative applications of this sort are a common persistent data space and the propagation of events to the relevant participants in a cooperative session.

Other, less complex tasks can also be performed using the sTeam event concept. For instance, a simple protocol tool for recording chat conversations in a room can be easily implemented using a container object that subscribes to the relevant communication events of a discussion's participants.

The sTeam event concept is rigorously implemented right up to the client level. Like any server objects, clients can process events, call server functions and thus trigger events. For this purpose, a special COAL-based (COAL = Client Object Access Layer)[7] server API was designed that forwards events via so-called sTeam connectors to the Java client. This enables a completely object-oriented representation of server objects within the sTeam clients. Benefits of object-oriented design that are often

[6] For information on the Extensible Markup Language (XML), cf. http://www.w3.org/XML/

[7] The acronym COAL was also chosen because a sTeam engine cannot function without coal!

attributed to pure Java client-server solutions are thus integrated in the hybrid sTeam architecture.

6.4 sTeam Access Rights Model

Multi-user systems such as operating systems traditionally make access rights to a file dependent on ownership conditions and membership of a specific user group (cf. the UNIX operating system in which access rights to a file can be specified for the user, the user group and other system users by three access-right attributes only.) These rights are evaluated according to a user's membership of specific groups. The classical model for assigning access rights of users and groups to objects is the Access Control List (ACL) according to Lampson. [14]. For every single sTeam object, the users and groups are defined in an ACL along with their respective access rights: *read, write, delete, delegate rights to others (sanctioning)*. Sanctioning is an extension of administration rights as proposed by Satyanarayanan [18]. This right allows the explicit modification of an ACL (in the case of our system, the ACL of a directory) and is thus an important first step toward decentralized administration. The sTeam system extends the right to administer an object, i.e. to modify access rights (sanctioning), by adding the option of delegating responsibility for an object. This delegation option for objects is taken from access-right models in drafts for relational databases. A formal model for the delegation of access rights to database tables was already drawn up by Griffiths and Wade [8]. (The creator of a table can delegate administration rights to other users.)

A meaningful and widely used strategy for administering access rights is to tie such rights primarily to user groups. With this approach, access rights are tied to the relevant user groups and only rarely granted to individual users. (Thus, most ACLs for a particular object list user groups only). The objective is to keep track of who has access to specific objects by granting as few rights as possible.

Such an approach implies that in many cases it will be difficult – if not impossible – to express a specific state of affairs in terms of purely positive rights. For instance, if an individual user is to be denied access to a particular object, the above approach would appear to necessitate removing the user from the group granted the access right in question and explicitly redefining all other access rights relating to that group – certainly a lot of effort to express the simple situation "person X is not allowed to access object Y". One solution here would be to use exclusive access rights – so-called negative rights that explicitly exclude users or user groups from a particular access right. Unlike some database systems, the sTeam system currently uses a model in which negative rights are only admissible for individual users, not for whole groups of users. While such an approach admittedly restricts somewhat the scope and powerfulness of the negative-rights model, it makes it easier to keep track of granted access rights and interpret them unequivocally in cases of conflict.

In daily life, authorizations and rights are dynamic, but at the same time shaped by social laws and norms. We take it, then, as a matter of course that, when being passed from one person to another, materials adapt their status in a number of ways to the respective use context, the actors involved or the place in question. For instance, a set of test questions changes its status from that of a highly confidential object accessible only to the teacher, before the test, to a freely available document that can be publicly discussed and commented on, after the test. Especially in cooperative teaching and learning processes, it would appear essential to dispense with a strict assignment of access rights to documents.

Certainly the simplest example of access rights to documents and materials is the above-described assignment of rights to users or user groups. If access rights are granted to a user group as a whole, all members of the group, and all members of its subgroups, acquire those rights.

Such a mechanism can be used to map access rights resulting from a user's social status, i.e. his/her permanent membership of a user group, but also dynamic processes such as the allocation of roles. For instance, if a participant in a discussion is appointed moderator and admitted to a corresponding group, he/she automatically acquires the access rights to documents and materials, or to the discussion environment, that go with this appointment.

Besides the characteristics of the access-rights model described here, sTeam allows such rights to be derived from the environment of an object or from a fixed object (e.g. the room). Rights can also be granted to groups as a whole, e.g. to enable groups of administrators to be set up. These options are not elaborated on here.

6.5 sTeam Clients

The sTeam clients constitute the real use interface to the sTeam system. At an appropriate point in the development process, implementation focused first on designing a powerful server and enhancing the required server interfaces. This resulted in a corresponding Java API and a number of experimental Java clients. One of the key design decisions with respect to the sTeam client-server architecture was to make the client-server interface as universal as possible, thus allowing server and clients to be largely decoupled in terms of enhancement. One of the essential characteristics, then, is the lean client-server interface. Thus the only commands provided by COAL are for logging a client on to and off from the server and for up- and downloading files, as well as a universal function for calling methods of objects in the server.

For each server object, a corresponding Java object can be replicated in the sTeam clients. The sTeam clients simply work with these replicated cooperative objects (methods and attributes). The access to attributes and methods within the corresponding server objects is mapped onto the server by the COAL interface. This process takes place transparently for the programmer of the client application.

Events are thus subscribed to by the clients and transferred to them from the server. To give a concrete example: to implement a chat in a client, so-called chat listeners can be attached to the corresponding user objects which, on receipt of a chat message, call an appropriate function to display the chat message.

In this way, an sTeam client generates its own personal view of objects in the server. The sTeam structure, consisting of persistent objects and their attributes linked by events, is rigorously implemented right up to the clients.

To this extent, the external Roxen web server connected to the sTeam server works like an sTeam client. Accesses to the web server (requests) are interpreted and converted into corresponding

sTeam object requests. The sTeam server controls the authentification/authorization of Net accesses (requests), manages the presented content from requested websites and simulates a structure of URL addresses. Similarly, the uploading and downloading of documents generates events in the server. The universal nature of the COAL interface means that no special interfaces are needed to attach the web server to the sTeam server.

6.6 Structure of Objects Within the sTeam Environment – ORB: Object Request Broker

All learning materials like documents and graphics – but also the sTeam users themselves – are represented in the server as sTeam objects. Some sort of basic order in this free structure is provided by sTeam rooms and containers which serve to encapsulate documents and users. In this way, materials and persons can be assigned an unequivocal position within a virtual world. The so-called "root" room forms the basis of a hierarchy of rooms and containers. Rooms are generally connected to other rooms by links and are subrooms of a parent room or of the root room. Architecturally, this means that each object has a definite environment from which attributes, for example, can be inherited.

Various situations occur in which the sTeam structure of rooms and documents must be searched or transformed. For instance, if documents are transferred to the server via a file-system-oriented protocol like FTP, a completely different structure must be mapped than, say, in the case of a search request for objects in the server.

The task of viewing the sTeam structure from different angles and perspectives is performed by the sTeam ORBs (Object Request Brokers).

An ORB is used to access the sTeam object structure. There are currently three different ORBs: one simply assigning sTeam objects an unequivocal ID – used, for instance, in object searches; one enabling an sTeam object to be accessed via a URL represented by a name, as typically used on standard web servers (here, arbitrary sTeam objects can be assigned a URL attribute that represents a sort of alias for accessing a particular document via a WWW client); and a special ORB managing the mapping of the sTeam structure by means of a hierarchy of rooms, containers and objects (here, each object has a definite environment).

Only the first ORB, assigning objects an unequivocal ID, can access all objects in the server. The hierarchical room structure and the identification via name URLs only map part of the server structure in each case.

7. RELATED WORK

A comprehensive treatment of the sTeam approach in relation to other research work in the area of Computer-Supported Cooperative Learning is beyond the scope of this paper. Engelbart and English [6] were definitely among the first to address the idea of computer-supported group work with their "augmention system" research on supporting human skills by computer technology. There are a whole series of approaches resembling the sTeam architecture in specific features (see [1], [2], [15], [17]), but none of them offers the sort of open and flexible object/event structure that the sTeam approach does. The GMD, for example, is developing the COAST platform [19] and its purely Java-based successor DyCE for designing Net-based cooperative applications. A characteristic feature of DyCE is its replicated system architecture consisting of cooperative Java objects. Here, though, the developers were not primarily concerned with learners' self-administration and the idea of fusing communication and document-management mechanisms. The idea of developing an environment for cooperative work based on MUD architectural concepts has already been tried out by the MITRE Corporation[8] with its CVE environment. Unfortunately, CVE's room structure is quite rigid and it provides only weak mechanisms for the distributed administration of rooms.

8. CONCLUSIONS

The sTeam's cooperative learning approach aims to provide primary media functions from the learners' perspective. In the sTeam concept, rooms offer users considerable scope for self-administration. Ideally, the participants in a virtual learning community design their own learning environment. This process begins with the creation of a virtual room and involves the processing of objects such as documents, graphics and slides as well as the selection of cooperative tools that are available throughout the learning process.

However, the solutions presented in the sTeam approach are not meant to compete with advanced technological solutions for individual applications. Providing solutions for practical everyday use – an objective calling for the integration of different tools and environments into a sustainable infrastructure – is a research goal in itself. It is not sufficient that technology work in principle; it must work on a practical, everyday basis – a requirement that has certainly been responsible for the spread of World Wide Web and its fundamental success over the past ten years. Similar efforts and quality standards must be applied in the future to the further development of cooperative teaching and learning environments.

This paper attempts to bridge the gap between our underlying theoretical ideas and our practical implementation of an open and flexible client-server structure. It focuses not on reading and browsing in the Net, but on cooperatively building and maintaining distributed knowledge structures. The desire to create a learning environment by learners for learners is what drives the sTeam research. It is in this spirit, then, that we hope to find new impetus and support, especially from proponents of the open-source idea, in our efforts to develop a free and flexible computer-supported cooperative learning platform. The rigorous application of open standards, e.g. providing comprehensive support for XML documents or developing powerful, interactive Java clients, will constitute an important challenge in the near term.

ACKNOWLEDGMENTS

Our special thanks go to Thomas Bopp and Ludger Merkens and all the other programmers and designers of the open-sTeam system for their unceasing efforts to realize sTeam.

This work reported here was funded by the German Federal Ministry for Education and Research (BMBF) and the DFN-Verein as part of our open-sTeam Project.

[8] cf. The open source Collaborative Virtual Workspace website, MITRE Corporation, http://cvw.sourceforge.net/

REFERENCES

[1] Appelt, W. and Mambrey, P. Experiences with the BSCW Shared Workspace System as the Backbone of a Virtual Learning Environment for Students. Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications ED-MEDIA 99, Seattle, June 1999, USA. 1710-1715.

[2] Chabert, A., Grossman, E., Jackson, L.S., Pietrowiz, S.R. and Seguin, C.: Java object-sharing in Habanero, Communications of the ACM, 41, 6 (1998), 69-76.

[3] Curtis, P. Mudding: Social phenomena in text-based virtual realities., 1992. ftp://ftp.lambda.moo.mud.org/pub/MOO/papers

[4] Elias, N. Über die Zeit. Suhrkamp, Frankfurt (Main), 1988.

[5] Ellis, C.A., Gibbs, S.J. and Rein, G. Groupware: some issues and experiences. Communications of the ACM 34, 1 (Jan. 1991), 39-58.

[6] Engelbart, D.C. and English, W.K. A Research Center for Augmenting Human Intellect. AFIPS Conference Proceedings of the 1968 Fall Joint Computer Conference, San Francisco, CA, USA, (December 1968), Vol. 33, (AUGMENT,3954), 395-410.

[7] Gibson, J.J. The Ecological Approach to Visual Perception. Houghton-Mifflin, Boston, 1979.

[8] Griffiths, P.P. and Wade, B.W. An Authorization Mechanism for a Relational Database System. ACM TODS, 1, 3 (September 1976), 242-255.

[9] Hampel, T. Scenarios of a New Dimension of Learning by the Co-operative Structuring of Knowledge. In: Bourdeau, J. and Heller, R. (eds.) Proceedings of ED-MEDIA 2000, World Conference on Educational Multimedia, Hypermedia & Telecommunications, Montreal Canada; (June 26 - July 1, 2000), 369-374.

[10] Henderson, A.J. and Card, S.A. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention, ACM Transactions on Graphics, 5, 3 (July), 1985, ACM Press.

[11] Hesse, F.W. Konzeption und Realisierung virtueller Wissensvermittlung. In: Hamm, I. and Müller-Böling, D. (eds), Hochschulentwicklung durch neue Medien, Gütersloh, 1997.

[12] Keil-Slawik, R. Evaluation als evolutionäre System-gestaltung. Aufbau und Weiterentwicklung der Paderborner DISCO (Digitale Infrastruktur für computerunterstütztes kooperatives Lernen). In: Kindt, M. (ed.): Projektevaluation in der Lehre – Multimedia an Hochschulen zeigt Profil(e). Reihe: Medien in der Wissenschaft, Waxmann, 7, Münster, 1999, 11–36.

[13] Keil-Slawik, R., Klemme M. and Selke H. Information and Communication Technologies in Education and Training (Part A). STOA Programme, European Parliament, Directorate General for Research, PE: 165.710., Luxembourg, 1996.

[14] Lampson, B. Protection. Proceedings of the 5th Princeton Conference on Information Sciences and Systems, Princeton, 1971. Reprinted in ACM Operating Systems Rev. 8, 1 (Jan. 1974), 18-24.

[15] Patterson, J.F, Hill, R.D., Rohall, S.L. and Meeks, S.W.: Rendezvous: an architecture for synchronous multiuser applications. Proceedings of the Conference on Computer-Supported Cooperative work, October 7-10, 1990, Los Angeles, CA, USA, 317-328.

[16] Polya, G. How to solve it. Princeton, NJ: Princeton University Press, 1971.

[17] Roseman, M. and Greenberg, S. TeamRooms: Network Places for Collaboration. Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work, November 16-20, Boston, USA, 325-333.

[18] Satyanarayanan, M. Integrating Security in a Large Distributed System. ACM Transactions on Computer Systems, 7, 3 (August 1989), 247-280.

[19] Schuckmann, C., Kirchner, L., Schümmer, J., and Haake J. M. Designing Object-Oriented synchronous groupware with COAST. In: Ackerman, M.S. (ed.) Proceedings of the ACM 1996 Conference on Computer Supported Cooperative Work (CSCW'96), (November 16-20, 1996) Boston, USA, ACM Press, New York, 30-38.