A Service-Oriented Infrastructure for Collaborative Learning in Virtual Knowledge Spaces

Thomas Bopp, Robert Hinn, Thorsten Hampel
University of Paderborn, Fürstenallee 11, 33102 Paderborn, Germany
www.open-steam.org

Abstract. Classical knowledge production is an author-centered process. The emergence of cooperative internet technologies such as wikis or blogs shows a shift towards stronger cooperative production and communication of knowledge, closing the gap between readers and authors. The incorporation of such participatory mechanisms into CSCL/CSCW systems raises several new requirements. We present an open service infrastructure that provides an architecture meeting these demands, while integrating itself into a network of already-established services.

1 Introduction

The internet has evolved into an important means of global communication. Email and chat have become common tools of communication, already surpassing their analogue counterparts in some places concerning usage and popularity. Over a long time the web has also become a publishing platform. Hypertext documents have introduced new forms of textual and multimedial contents and structure. Moreover, digital media have enabled users to modify or annotate content within the media itself, overcoming a technological limitation of analogous media. Regarding the internet, this has mainly been an author-centered process, involving feedback or change of the content via interaction through separate media or through up- and download for interaction in separate client software. The emergence of new forms of collaborative publication (e.g., blogs and wikis) offers a different process of media creation and publication with the participation of the reader as an author or co-author. In addition to opening up the publishing process to a collaborative approach, the ability to work and communicate within the digital content medium in the location of its publication helps to avoid further media breaches. Users can modify

or annotate media in place, overcoming the need to create and submit changes separately. This results in a more direct, easier and quicker means of creation, revising and publishing of common media content. The rapid spread and wide acceptance of these new technologies and processes also brings to attention a new phenomenon with relevance to computer supported cooperative learning: We are experiencing a shift from classical production of knowledge towards a stronger communication of knowledge, supported by technology.

A CSCL system based on the theoretical media functions model [1] supporting collaborative semantic structuring and linking of lecture materials and the production and integration of personal materials into a common knowledge structure enables students and teachers to evolve a simple document management base into an external memory. The ability to structure information from an individual point of view, meeting individual demands, allows the creation of an individual learning environment, adapted to the specific needs and desires of each student. Support for synchronous and asynchronous forms of communication and interaction can enhance the cooperative experience of users, but a persistent environment that can be structured by users themselves allows for location- and time-independent work. This enables students to cooperate with others while still being able to choose their own learning speed.

Support for a tight involvement of students into a common knowledge environment and the ability to structure, annotate or modify this environment imposes strong requirements on a CSCL platform. A concept for individual and common nodes of external memory must be developed. Our approach uses the metaphor of virtual rooms, persistent virtual knowledge spaces, to focus on the collection and structuring of, as well as the interaction with, digital media. To allow for enough freedom in the design of individual and common knowledge spaces, a form of self-administration for students is necessary, enabling them to organize their cooperation.

Taking this demand for communication of knowledge and a continuous, individual, non-disruptive cooperative work on media further, the need for a technological infrastructure for continuous work on digital media emerges. To satisfy this need with a strong focus on time- and location-independent collaboration, adapted to the users' needs, we have developed the sTeam server architecture.

The extension and integration of this architecture into open infrastructures for cooperative knowledge management is an ongoing process of our work. In this paper, we will describe our basic concepts, the server architecture and its integration into an open infrastructure for information technology systems.

2 Basic Concepts

We will first describe several basic concepts of an architecture for cooperative knowledge management and knowledge communication. In section 3 we will then present the technical implementation of such an architecture.

Virtual Knowledge Spaces

An architecture for continuous, time- and location-independent cooperation and learning must support the creation and structuring of individual environments as focal points for knowledge representation, structuring and communication. We are propagating the use of virtual knowledge spaces. These "rooms" are meeting places for users, containing user objects, documents and communication channels in one place [2, 3]. Rooms are nodes for synchronous and asynchronous communication and collaboration. Most importantly, however, they can be used for *continuous* cooperation, independent of location and time. This is achieved through persistent object storage and awareness mechanisms, both implicit (e.g., display of changes made to documents) as well as explicit (e.g., message boards). Rooms can contain links to other rooms, so-called "exits", but they may also contain further rooms, so that it is possible to create hierarchical structures of virtual knowledge spaces, as depicted in Figure 1.

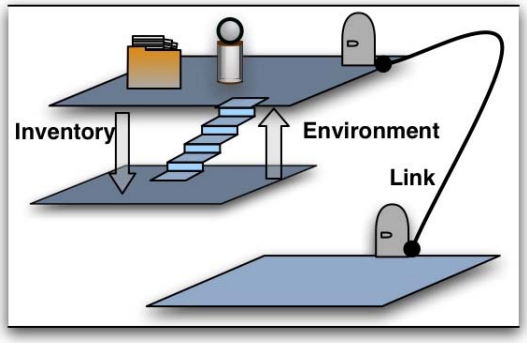

Figure 1 – Structuring virtual knowledge spaces as rooms

Customized Views

To fully exploit the possibilities of virtual knowledge spaces, it is necessary to provide differing, customized views of them. In addition to the common presentation of a room as a list of contents, different views of the same room can be created to match different usage requirements. An example would be a gallery view for a room containing mainly images. Besides the visual representation, views can also provide different interaction methods, e.g. a room where documents can be submitted by a certain group of users but afterwards can be accessed only by a different group. This allows for rooms acting as means for simple workflow mechanisms. Several different views of the same room, the same knowledge space, can exist simultaneously. To recall the example with a room for submitting documents, there could be a "student view", providing only a mechanism to submit

or review the student's own documents, as well as a "teacher view" that shows all submitted documents and provides means for annotating and rating them.

With the vast amount of possibilities for document management provided by the system, user and group management, workflow mechanisms, etc., one of the benefits of customized views is to provide reduced functionality for each specific use. This centers the user interface on the task the user wants to perform, hiding data and interaction elements to provide a clear focus on the task at hand.

The web interface, however, can provide only limited means for collaboration, bound by the peculiarities of the HTTP protocol. Once a web page has loaded, there is no means by which the web server can notify the browser of any events, except when the browser requests or reloads a page. Tighter means of synchronous communication and interaction can be reached by client software not limited by the HTTP protocol. We will later describe an open protocol developed for the sTeam server, by which clients can interact directly with the server, allowing for applications such as chat or shared whiteboards.

User Management

The user management of a system with strong emphasis on individual and collaborative creation and structuring of knowledge must provide flexible means for self-administration. Users must be able to regulate who may or may not structure or modify their material. Self-administration requires that users can delegate access rights on rooms and documents, as well as administrative rights to others.

To achieve a clearer administration of access rights, users can be assigned to user groups, which can serve to define roles of access [4]. Groups have special administrator roles which can add, remove or invite users into the group.

This concept of self-administration requires a clear and flexible system of access rights. Access Control Lists [5], an established concept in the field of operating systems, can be applied to match this requirement. A list of various access rights for different users or groups is attached to each object, controlling usage options. Common access rights models of read, write and execute are not flexible enough for cooperative knowledge management and structuring. They need to be further differentiated into *read*, *write*, *delete*, *execute*, *insert*, *move* and *sanction*.

This enables users to regulate the right to change the content of a document (write), delete an object, insert objects into an environment or move objects out of an environment. This allows, for example, for virtual rooms where documents can be submitted by one group of users, then reviewed by another group and finally be moved to another location for publishing.

The sanction right takes a special role. It is used to allow users to assign rights to or remove rights from other users, thus forming the base for decentralized self-administration on an object-by-object or room-by-room base.

Objects inherit access rights from their environment. This means that when an object is moved from one virtual room to another, its effective access rights change. Users with access to the new location of the object now have access to the object, too. Inheritance of access rights can also be switched off for individual objects.

Figure 2 shows how access control lists and inheritance from environments can be combined. The group on the left has access to the folder because access rights are inherited from its workroom. Explicit permissions on the workroom to the right have been assigned to the user on the right-hand side. The group on the upper right has access to all the depicted objects and rooms. Their permissions from the access control list are inherited to the room on the left because of the room structure and to the room on the right because that room explicitly inherits the ACL.

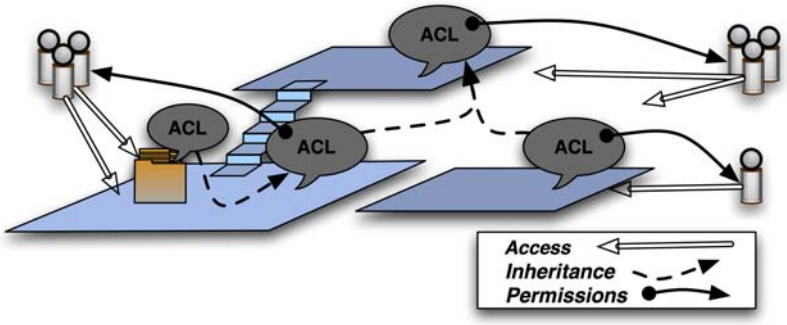

Figure 2 – Object-bound or inherited Access Control Lists

Open Infrastructure

When designing an architecture that is to be flexible, adjustable and at the same time deal with a multitude of digital media and their structuring and linking, it is useful to make it integratable into an infrastructure of specialized systems, ranging from administrative databases over communication channels to content servers. An infrastructure of open protocols for interaction with and between various systems can tackle this requirement.

Another aspect of an architecture that is centered on customized environments for individual users is the demand for support of the users' preferred tools and applications. Being able to use client software that users are already accustomed to lowers the need to adjust to new technology and greatly increases the acceptance by new users.

This approach of opening up to common interfaces also provides a basic level of integration with other established systems. Interaction with the server at this level is very specialized, centered upon the explicit purpose of each supported protocol. To support more generic and flexible modes of access opening up the whole functionality of such an architecture, an open protocol must be defined that provides remote object and method access, or in technical terms, remote method invocation.

3 Architecture

The sTeam server [6] has been designed with a microkernel architecture in mind [7]. This enables a modular approach where the base functionality is provided by modules and services, while the core kernel manages only interaction of its parts. The traditional way of storing all objects in a database has been extended through persistence layers, allowing for different object repositories for storage and retrieval.

One of the main goals of the server development was to provide users with various methods of access to the server, allowing them to connect with different, established tools that they were already accustomed to. This requires the implementation of open, standardized protocols that are supported by various client applications.

Interfaces

The most prominent of the modules dealing with network protocols is the HTTP (Hyper-Text Transfer Protocol) module, which enables the sTeam server to function as a webserver. In combination with an XML/XSL parser module, web pages can be dynamically generated from templates to provide different views on the same data. This is the mechanism used by the server to provide specialized views on knowledge spaces through web browsers.

With wikis becoming increasingly popular on the internet, we have established a wiki component, so that documents written in wiki syntax can be viewed, created and edited in the usual way. Since the main focus of wikis is the quick and easy collaborative editing of text documents, they seem like an ideal contribution to any CSCW system. Because of sTeam's focus on virtual knowledge spaces, we have extended the basic, flat structure of wikis in a way to facilitate the hierarchical structure of knowledge spaces [8].

Another feature that is gaining popularity on the internet is RSS (Really Simple Syndication) feeds. These are usually used as a standardized method for news feeds, though in sTeam every object in the object repository offers its own RSS feed containing a history of changes made to the document. This provides users with an additional means of awareness that can be accessed through established client software (such as email clients or web browsers). In addition, to publish content via HTTP the server also offers secure connections via SSL (Secure Socket Layer) to allow registered users to access an internal web view of the server, again with the option of specialized views.

Apart from HTTP, FTP (File Transfer Protocol) is the most common protocol for the up- and download of files in the internet. The more recent WebDAV (Web-based Distributed Authoring and Versioning) [9] is a common protocol for easier access to remote file repositories. WebDAV enables clients (often the file browser of the operating system itself) to access remote directories in the same manner as local hard disks. In addition to the easy use, WebDAV offers a file revisioning system. Both protocols are supported through kernel modules, so that established client software can be used to access the object repository just like a file server or remote file system.

Other modules provide SMTP (Simple Mail Transfer Protocol) and IMAP (Internet Message Access Protocol) support, so that the server is able to send emails and offer users access to messages received in sTeam. Since these are open, standardized protocols, arbitrary email readers can be used to access the server, meaning that users can use software they already use in everyday work. Every user's and group's name also functions as an email address on the server, so that emails can be sent to single users or to groups, forming simple mailing lists without any additional administrative work [10]. Even knowledge spaces and documents in the object repository function as email recipients. Emails sent to documents are attached as annotations, while emails sent to knowledge spaces place their attachments as new objects into the knowledge space. Likewise, IMAP access to knowledge spaces provides access to the objects stored within.

Synchronous Collaboration

The possibility of using established software for interaction with the server certainly is a great benefit for users, since they are able to use programs they are already accustomed to. However, none of these programs offers support for synchronous collaborative work. To accommodate for this, we have developed Java-based client software, the shared whiteboard.

The shared whiteboard client provides a two-dimensional, spatial view on knowledge spaces. In addition to the location of objects within the virtual environment provided by the server through rooms, containers and links, the shared whiteboard provides a view on rooms where their contents can be arranged two-dimensionally. As on a real whiteboard, text, lines, boxes and other graphical primitives can be used to augment the arrangement. The emphasis of the shared whiteboard clearly lies on synchronous cooperation. In addition to a list of users who are currently in the same room (no matter if through the shared whiteboard client, the web interface, IRC or another client) and a chat channel for that room, the interaction with the whiteboard is immediately visible to other users in real-time. If one user moves objects on the whiteboard, this movement is immediately displayed to users currently connected to the same room via the shared whiteboard. Chat messages written in the whiteboard also appear on the web based java chat or external chat clients and vice versa.

To make the spatial arrangement available to users not working with the shared whiteboard, we have created a new web interface view that displays the spatial arrangement as an SVG (Scalable Vector Graphic), a standardized graphics format that is supported by web browsers or web browser plugins.

Extensibility and Connectivity

The sTeam server offers a variety of external interfaces in the form of different network protocols. In addition to the kernel modules, which are closely integrated into the server core, the sTeam server offers a more high-level means of direct interaction with the server functionality: the COAL protocol (Common Object Access Layer). In comparison to middleware like CORBA, COAL is a lightweight

protocol that provides remote method invocation and object access on the sTeam server. Implementations of COAL are available in Java and C++. There is also a simple PHP API with support for sTeam interaction, so that web-interfaces can also be written in HTML/PHP and hosted on a dedicated web server while using a sTeam server as a CSCW backend.

Implementing server extensions as separate services instead of kernel modules allows for greater stability and flexibility. Services run as client programs on the same or a remote machine, not sharing memory or other resources with the core server. Since service communication is usually asynchronous, they cannot block the main server. A search function is an example for such an external service.

To allow for easier extension of the server through standardized means, we are currently developing a service that provides access to the server's functionality via SOAP, the standard protocol for web services [11]. SOAP encapsulates remote procedure calls over a standard HTTP or HTTPS connection in a platform and programming language independent manner. Web services contain a self-description of their functionality in an XML-based form (WSDL -- Web Service Description Language) that can be obtained and interpreted by clients or other services. There are even some approaches to tie semantic information into the web service description to allow web services to automatically find services that provide a desired functionality and use them.

There are already a number of integrated development environments that can create wrapper classes for certain programming languages from the WSDL data of web services, reducing the complexity of their access to simple high-level remote procedure calls. This, and the increasing prevalence of SOAP-based web services and tools, will certainly simplify the integration of external services into sTeam and of sTeam into established infrastructures. For example, we are currently developing a Java-based service that interacts with a SOAP-based web service of the university library. It enables a transparent integration of literature references and electronic media provided by the library into the object repository of the sTeam server. The documents themselves are not copied into the repository since the electronic archives of a library have different demands and capabilities than the object repository of a CSCW server. However, the documents and references can be accessed transparently through the service, as if they were part of the repository itself. Another prominent example of web services that could be integrated are the SOAP-based services provided, for example, by Google, Amazon or Ebay.

Figure 3 illustrates the interfaces of the sTeam architecture. Web pages for browsers can be provided directly by the sTeam web interface or by the PHP interface on an Apache web server. The WebDAV protocol allows direct access to the object repository. Various client applications for email or chat can connect to the server via standard network protocols. The COAL protocol allows for complex operations on server objects. A more flexible approach is the SOAP protocol that is supported by a Java-based service placed, for example, in a Tomcat web service container.

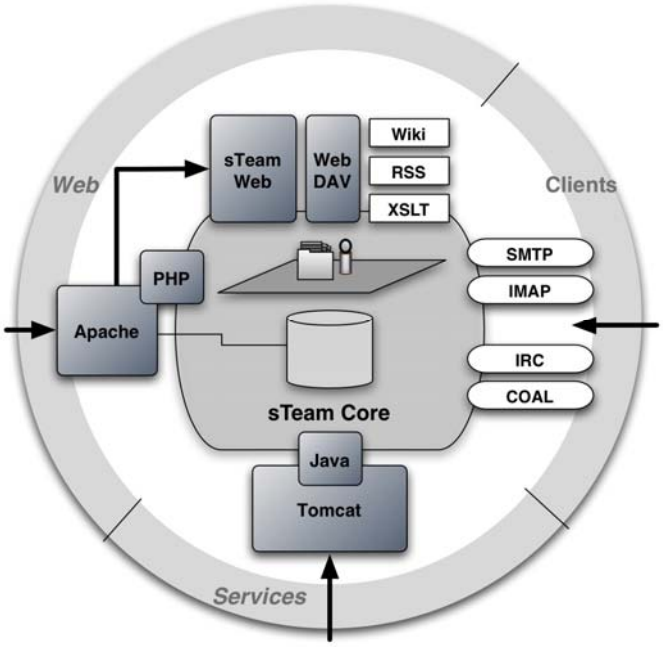

Figure 3 – sTeam service architecture

4 Conclusion and Outlook

In conclusion, the sTeam server has proven to be a solid platform for cooperative learning and working. There have been several evaluations of its practical use in several university lectures and projects [12].

Our current research topics deal with the integration of different information technology systems into a common, flexible infrastructure. We are in the process of fusing the heterogeneous systems at the university, ranging from user directories, administrative platforms, examinations office and email account servers over content management systems, web servers and CSCW/CSCL servers to electronic library archives. Just as we integrated several open, standardized network protocols into sTeam to allow users to keep using their favorite client applications, our vision here is not to replace the various established systems with centralized, monolithic server software, but rather to create an open infrastructure that interlinks the functionality of its specialized components.

Our vision for the next generation of sTeam is an open system that is part of an open, service-oriented infrastructure. User authentication will be valid across servers, and objects can be accessed independently of their physical location. Established servers provide their specialized facilities transparently, augmenting objects with additional functionality.

References

1. Thorsten Hampel and Reinhard Keil-Slawik, sTeam: Structuring Information in a Team – Distributed Knowledge Management in Cooperative Learning Environments, ACM Journal of Educational Resources in Computing, 1(2)2002, 1–27
2. D. Austin Henderson and Stuart Card, Rooms: the use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface, ACM Transactions on Graphics, 5(3)1986, 211–243
3. Mark Roseman and Saul Greenberg, TeamRooms: network places for collaboration, Proceedings of the1996 ACM conference on Computer supported cooperative work, Boston, Massachusetts, United States, ACM Press, 1996, 325–333
4. D. Ferraiolo and R. Kuhn, Role Based Access Control, Proceedings of the 15th National Computer Security Conference, 1992, 554–563
5. B. Lampson, Protection, ACM Operating Systems, 8(1) 1974, 18–24
6. Thorsten Hampel and Reinhard Keil-Slawik, sTeam - Designing an integrative infrastructure for Web-based computer-supported cooperative learning, Proceedings of the 10th World Wide Web Conference, 2001, 76–85
7. Thomas Bopp and Thorsten Hampel, A Microkernel Architecture for Distributed Mobile Environments, Proceedings of the seventh International Conference on Enterprise Information Systems (ICEIS), 2005, 151–156
8. Thomas Bopp and Thorsten Hampel, Integration of New Technologies into a Room-Based CSCW System, Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, AACE, October 2005, Vancouver, Canada, 2822–2827
9. Y. Goland and E. Whitehead and A. Faizi and S. Carter and D. Jensen, RFC 2518 — HTTP Extensions for Distributed Authoring — WEBDAV, February 1999
10. Christian Schmidt and Thorsten Hampel and Thomas Bopp and René Sprotte, We've Got Mail!? A New Quality of Integrating E-Mail Services Into Collaborative E-Learning Environments, Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, AACE, 2004, Washington, DC, USA, 2919–2925
11. Thorsten Hampel and Jörg Halbsgut and Thomas Bopp, Heterogenous Integration of Services into an open and standardized Web Service, Proceedings of the sixth International Conference on Enterprise Information Systems (ICEIS), 2004, 182–189
12. Thorsten Hampel and Reinhard Keil-Slawik, Experience with Teaching and Learning in Cooperative Knowledge Areas, Proceedings of the twelfth International World Wide Web Conference, Budapest, Hungary, May 2003