INSTANT COLLABORATION
Developing a Framework to Support New Patterns of Team-Cooperation

Alexander Schatten, David Pölz and Matthias Epheser
Institute of Software Technology and Interactive Systems , Vienna University of Technology, Favoritenstr. 9-11, Vienna, Austria

Keywords: Instant collaboration, instant messaging, resource exchange, team collaboration.

Abstract: Communication and collaboration is supported by a multitude of different (client) applications and backend systems. At the same time, new patterns of collaboration between knowledge workers arise: people want to communicate in-time, share resources, keep track of their issues and yet do not want to be unnecessarily disturbed in their concentration. We analyse several systems and approaches to support team-collaboration and finally introduce our understanding of "instant collaboration". In our research we support first patterns with a middleware and client-side prototype. The system is built upon proven systems for (instant) messaging, communication and repositories, and suggest a system that allows support for new work-patterns, yet integrates seamlessly into existing IT and workgroup infrastructure and company policies.

1 MOTIVATION AND INTRODUCTION

Knowledge workers want to communicate, they want to share files, and they want to keep track of their work. There are various tools and platforms available today that support many aspects of cooperation. Yet we believe that most systems currently available (commercial or open source) show significant weaknesses in terms of *instant collaboration*, meaning the rather ad-hoc forming of workgroups and supporting their resource-sharing patterns. We will analyse these issues in more depth in the next section.

We started developing a new framework for instant collaboration, named *JMellow* which is part of the larger *Peer Group* research effort. JMellow was created as a research project, because there we see the need for an easy-to-set-up platform that allows instant communication and particularly resource exchange, annotation, discussion and versioning. JMellow is based on available and proven open-protocols and open source systems (like instant messaging solutions and repositories/content management systems).

Yet we notice a resistance against installing new tools or introducing new protocols and proprietary systems: resistance to new tools typically comes either from management (in a corporate environment), from system administration (who has to install, maintain and integrate these new tools) or from the potential users themselves. Already Grudin pointed out, there should not be "a disparity between who does the work and who gets the benefit" (Grudin, 1988). Thus, it should integrate in existing user databases and content management systems (to please system administration), be easy to use and easy to administrate. Moreover it should use (if possible) the client applications people already use and only extend them where required.

Eventually, traceability of work should be possible. Also, security issues and other policies (in the corporate environment)—typically associated with instant messaging (peer to peer) systems (Bhagyavati, 2005)—have to be addressed with our approach. This is hopefully the foundation to encourage management to consider instant collaboration tools as a next step in enhancing the efficiency of their knowledge workers.

JMellow can be seen more as an instant collaboration middleware than a specific client-based system. Of course, we also developed client(s) in this project, but the system can be easily embedded in existing infrastructure. As knowledge workers should have access to the files of their workgroup everywhere and

Schatten A., Pölz D. and Epheser M. (2007).
INSTANT COLLABORATION - Developing a Framework to Support New Patterns of Team-Cooperation.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Society, e-Business and e-Government / e-Learning*, pages 21-28
DOI: 10.5220/0001268300210028

should be able to communicate with their workgroup members, we have chosen to make the web client the first client implementation. Others are following. Of course, JMellow can then be a simple file-sharing and instant messaging application. Workgroup-related performance metrics can be gained from the data that is exchanged over the JMellow platform. Hidden work flows can be found, communication structures be made obvious and project activities be measured.

This paper is structured as follows: section 2 will outline some instant collaboration scenarios in detail, whereas section 3 explains our JMellow infrastructure in more detail. In section 4 we describe related work in this field. Section 5 gives a more in-depth description of the system architecture. Finally in section 6 we give conclusions and outlook for future research work.

2 INSTANT COLLABORATION

A broad variety of applications are available on the market, or as open source software projects that address cooperation and communication. Mandviwalla et al. (Mandviwalla and Olfman, 1994) describe various needs and requirements of groups, and emphasise in detail, that different groups and collaboration scenarios have different requirements and it is impossible to deliver a generic groupware application that works out for everybody. *Open design and customisation* is necessary to support a broad range of applications. Thus, we see openness as a foundation of our project architecture, as will be described later on.

Moreover, in our research work, we focus on what we call *instant collaboration*, which imposes new collaboration patterns that are not well supported by available software (see section 4). Daily work in distributed teams, show that beside communication, the most important feature is a solid way to exchange resources. Typical scenarios are sending files via email or instant messaging clients. These approaches have the disadvantage to be rather intrusive, as the workflow of the colleague is most likely being interrupted by this procedure. Moreover, resources are exchanged on a person-to-person level; meaning that significant bandwidth is wasted, if (as it is often done) many files are “broadcasted” to a larger community of coworkers, while many are probably not interested in this information, or not *at this very moment* interested in the information.

Additional issues come up when documents are changed and sent back to the originator or to other persons. In a short time, documents run out of syncronisation. Other ways to share such resources typically are shared-file systems, or even better CVS or SVN repositories or other web-based file repositories like BSCW (BSCW, 2003). These systems allow a controlled storage of shared information, some also versioning. Hence such system can provide valuable help for group cooperation. However, some other disadvantages persist: to access these repositories, typically specific client software is needed or browser uploads have to be done. But more important, *the repository interaction is not connected to communication and collaboration activities*.

We believe, that *instant collaboration* means communication *and* closely-connected other activities like file-sharing, annotation, access to other information systems, probably project management functionality and the like. Closely-connected means: ideally in the same client software, and with a usability that does encourage also less skilled users. Hence we focus at the first step on integration of communication (instant messaging) with file/resource-sharing, versioning and annotation. Further steps are outlined in section 6. But our concept is rather a communication/collaboration integration project, as we explicitly do not reinvent the wheel. So systems like SVN, JSR 170 repositories, content management systems or instant messaging middleware (Jabber, XMPP) are used as a basis and are integrated to support users in working with those valuable tools in a more productive way.

3 INSTANT COLLABORATION WITH JMELLOW AND INTEGRATION OF AVAILABLE SERVICES

With JMellow, we want to demonstrate the feasibility of our *instant collaboration* concepts (fig. 1 shows a screenshot of the Open Laszlo/Flash client prototype). JMellow is platform-indepentent and will be provided under Apache open source licence (Apache Licence, 2006). Instant collaboration with JMellow is very straightforward. The user sees his or her colleagues ordered in groups in the JMellow client. As JMellow is primarily designed for usage within a closed infrastructure (Intranet), typically the user and group information is provided by the middleware (database). For our prototypical implementation, we use the Java-based Wildfire server (Wildfire Server, 2006). User and group information can be provided either from the Jabber-Server directly, or via an LDAP server. This is particularly handy in a corporate environment.

The closed-group (intranet) infrastructure ap-

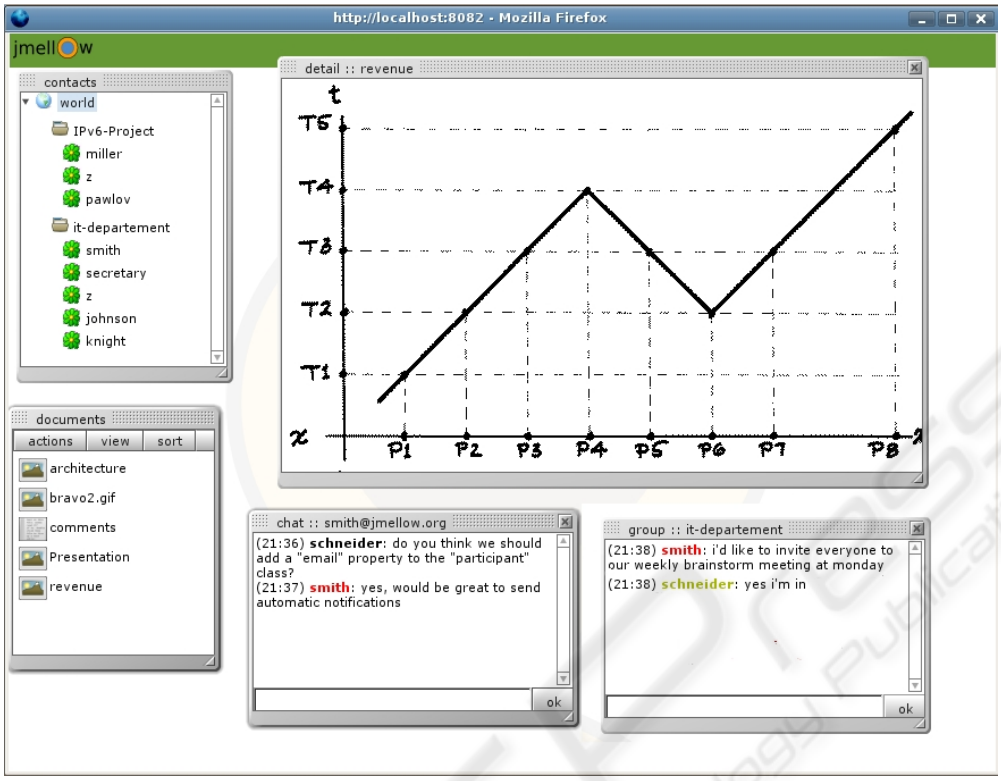

Figure 1: Screenshot of the Flash (Open Laszlo) JMellow client. Contact list and documents are shared. In two chat windows personal and group chats are held. The Flash client even can preview certain types of documents (like the graphics in that case).

proach is also beneficial in addressing issues with IMs on the corporate desktop (Bhagyavati, 2005) (security, policies…). Usage as "open internet collaboration platform" is *not* the target audience.

The consequence is that new colleagues immediately have a filled group/user list including contact information. This list of users can be used to chat via XMPP protocol (Jabber). It is noteworthy here, that the group/user structure also implicitly controls access control and visibility: i.e., if a user wants to chat with a person, she selects this person; if she wants to chat with a group she selects the group and automatically a group, chat is initiated. If a resource is shared with a group, implicitly all group members have access to this resource as long as they are assigned to this group.

Using JMellow for resource management, it is avoided that large chunks of information are distributed via email, and the like. If a user wants to share—say, an Office document—he "drag-and-drops" it on the *person* or *group*. This document is then sent to the JMellow server (which is from a technical point of view a Jabber *client*) and stored in a proven repository. We experimented with a JCR-170 repository (Jackrabbit) (Nuescheler, 2006; Jackrabbit Content Repository, 2006), but currently use the Daisy (Daisy CMS, 2006) content management system.

This is an important strategy, as we do not want to implement new systems where proven solutions are available. The strategy is rather to connect proven systems and enable their usage in instant collaboration setups.

To conclude the advantages in terms of collaboration on resources:

- The *target audience* of the resource is not disturbed by incoming mails or messages (unless notification is desired), yet it is shared and seen presently
- Document is listed in the provided resources of the clients of the target audience
- Document is *not* downloaded automatically, but upon request
- The resources are not only kept on clients, but on reliable (versioned) repository servers
- Document exchange can be monitored and traced

- As the document is stored in a document repository/CMS, it can be used by other applications like the web-publishing infrastructure, the full text indexing and so on

If a user is interested in this provided information, she can download it, annotate it (in a "BLOG-like" style) or put new versions on the server.

Also traceability is guaranteed: first of all via the repository/CMS, and secondly also within the system; e.g., for new users: consider a new user joining a workgroup: as soon as the administrator gives him an account and assigns him to one or more groups, he sees all his colleagues including contact information, plus all resources that are "published", plus all annotations that were made to a specific document, etc.

The *root node* has a specific notion in the group list. This node actually means *world* (whatever *world* means in the very context of the system). The idea here is (depending on the configuration of the system) that every user can easily publish information in a "BLOG"-like style. To give an example: a secretary could drag a picture of the last event to the *world* node, then add some meta-information (like title and description); then JMellow publishes this picture with the according meta-information via the CMS on the corporate News page.

Features for the next versions of our research prototype go some steps beyond: in our notion, *instant collaboration* should also allow to let different systems interact with each other; i.e., it should be possible to associate resources to chats and vice versa. The idea is to provide users a history of their chat sessions with connected documents; even the other way round: it should be possible to annotate documents (like in the current version), but also chat about documents and see this chat associated with the document.

We are also working in integration of the JMellow client in other (communication) applications. The first (obvious) step is to write a plugin for a popular Jabber Client (Spark (Spark IM Client, 2006)). Other steps should include a plugin for the Mozilla Thunderbird mail client using XUL (Thunderbird, 2006). This allows integration (as described before), but this time between emails, chat and resources.

4 RELATED WORK AND APPLICATION CONTEXT

In the previous section we gave an introduction of the term *instant collaboration* and provided our concept as well as a brief description of the JMellow implementation. Currently, several other projects provide comparable functionality, at least for some domains: Cryptoheaven (Cryptoheaven, 2003) is a commercial application, focussing on security, hence, providing secure email, file-sharing and IM integrated in a stand-alone application. It also demonstrated, that there is need for an integrated application. Groove (Groove Virtual Office, 2006) is also a commercial application. Various business plans are considered, reaching from just file-sharing to screen-sharing. In 2005 it was acquired by Microsoft and will apparently *not* be a platform independent solution.

MyWebdesktop (MyWebdesktop, 2006) is also a commercial product in beta phase having a web interface for weblink and file-sharing, which also supports IM and forums. It demonstrates the feasibility of the concept working over the internet.

Jogger (Mecham, 2002) shows blogging via a Jabber client, hence, providing easy weblogging functionality that uses instant messaging protocols.

Geyer et al. (Geyer et al., 2003; Millen et al., 2005) showed an activity-centric desktop using peer-to-peer technology and shared objects and also provided a study of the usage of their system (Muller et al., 2004). However, in our opinion, this work also demonstrated that it is not necessarily a good idea to build everything from scratch. We feel, that there is no need for reinventing the wheel(s) like implementing IM or repository/CMS infrastructure when established open source projects are available. Particularly as there is established open source technology available to provide this functionality.

A very interesting project was introduced by IBM research (Jr. et al., 2002): "YouServe" (sometimes called UServe); The general idea was file-sharing and mirroring by using small private webservers. Still, it appears that this project did not find a place in a commercial IBM product, and currently there is hardly information available about this project any longer.

Besides these more "general purpose" approaches, there are also very few concrete instant collaboration projects in specific domains like designing software with UML (Hansen and Damm, 2002) or collaborative text-editing (SubEthaEdit, 2006).

From our point of view, the promising paradigm of instant collaboration—that has to include non-invasive resource-sharing, versioning, annotation and eventually the integration of different collaboration systems—is not yet fulfilled by any of the approaches we found (neither commercially nor as open source project).

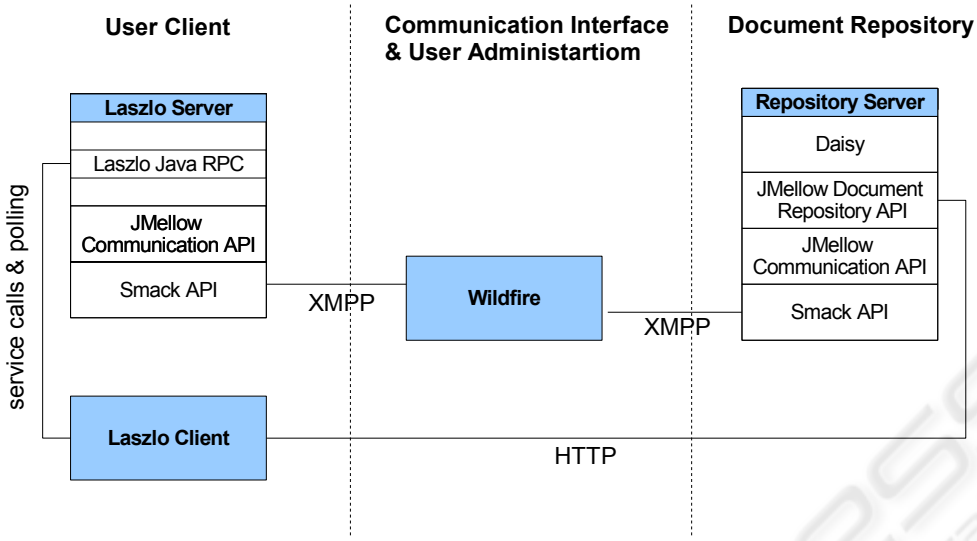

Figure 2: Archtecture of JMellow.

5 JMELLOW ARCHITECTURE

5.1 Components and Design Goals

A JMellow system currently consists of four independent software components:

- Communication Layer (JMellow Communication API)
- User Administration
- Document Repository Server Interface
- User Clients

The communication interface is the link between the other three components. It allows them to send events and retrieve callbacks. Every message/event transport technology like Jabber or JMS could be used. User Administration manages users and groups. It creates the ContactList for a user. The generic interface offers the possibility to implement ports to existing systems like LDAP or Jabber Servers.

The Document Repository Server stores documents (content and meta data). It generates the user's document list and provides methods to fetch document content. The JMellow Data Access API can be a bridge to CMS system or other repository solutions. (e.g. Jackrabbit)

A User Client is a frontend that offers the chat and document browsing functionality. It could be a standalone application, a web application or a plugin to an existing software product.

The JMellow architecture is very flexible. Every component is independent from the others. We try not to reinvent the wheel, by implementing very complex functionalities like message transport or a document repository with versioning by ourselves, but create a framework that makes it possible to plug in every suitable existing software. Developers can choose the right combination of implementations for their circumstances.

5.2 User Administration

A user is a single person identified by a unique ID. Every user can have a human readable alias. Users are organised into groups. Every user can be part of one or more groups (this is essentially following the XMPP specification (XMPP Specifications, 2006)).

Only the administrator can create users and groups. The user's contactlist is created on the server side and sent to the user clients. A user knows only the groups he's part of and all the members of these groups.

5.3 Event Data

Events are sent between the components. Some events are just method calls (e.g. askForContactList), some events contain data (e.g. sendMessageToUser) though. The following types of data could be sent:

ContactList After logging in, the UserClient receives the contact list containing the structured groups and users.

UserMessage Represents a chat message to a user in the contactlist. A one-to-one conversation can be

started, similar to ICQ or MSN.

GroupMessage Represents a message to a group. Everyone in this group receives the message. An IRC-like chatroom is opened for the group.

DocumentList The user receives a list of all documents he has access to. The list only contains the meta data like name, id, date and type.

DocumentContent Represents the content of a document. This could be a text-file, an image-file or other binary data. It is possible to send the data as a binary stream within the the event or to just provide a URL where the content can be fetched by the client. Every implementation can choose what is best concerning the network architecture.

DocumentCommentList A list of all comments attached to the specific document.

5.4 Communication Layer

The communication layer offers an interface for user administration, repository server and user clients to communicate with each other. Various sets of service methods are provided for different components:

User Client ask for contact list, ask for document list, send message to user, add/change document

User Administration send contact list to user client, synchronise user data with document repository

Repository Server send documen tlist, document content and comments to user client

Some actions require a callback mechanism to tell the clients that a particular event has arrived. Some clients may not be able to be called back (e.g. an Ajax web-application). Therefore we implemented two different ways of retrieving data. A client can implement a listener interface for callback or the client may choose a polling mode.

Figure 3 illustrates the login procedure (from client side) and initialisation step.

As mentioned above, the JMellow concept is generic, hence, different concrete implementations and integration of different technologies are feasible, as seen below:

- Jabber Protocol (e.g. Smack API)
- Java Message Service
- Web Service Calls
- Java RMI

5.5 Document Repository Server Interface

A JMellow document belongs to one user, called the document owner. The owner defines the access rules for his documents. A document can be readable/writeable for just the owner, members of one or more groups, or everyone. One or more comments can be added to a document.

The JMellow Document Server Interface is a transparent layer that exposes the following methods:add a document, change a document (content/access rules), get document content, get document list. As a storage component, repositories like JackRabbit or a content managemenet system like Daisy are conceivable (or an own implementation, of course).

5.6 User Clients

User Clients are very customisable, including standalone desktop solutions, plugins to existing projects or web-based applications. Currently we have developed a Flash prototype using Open Laszlo (Open Laszlo, 2006). Other client prototypes are planned (Spark plugin, mail-client plugin).

5.7 Reference Implementation

Figure 2 illustrates the architecture of the reference implementation: Wildfire provides the Jabber chat server as well as the user administration. The repository server synchronises the groups and users with the wildfire database. The repository server is a servlet running in a Jetty servlet container. It retrieves calls for documentlist/contents/comments via Jabber messages within the JMellow communication layer. This layer uses the Smack Java implementation of the XMPT protocol (Smack XMPP API, 2006). The calls are passed to the JMellow document repository interface. This implementation acts as a bridge to the Cocoon-based CMS Daisy. Daisy provides a straight forward Java API to manage document actions.

The User Client component contains the Laszlo presentation server as well as interactive Laszlo Flash clients that run in every browser with a Flash plugin. The server—again a servlet in a Jetty—retrieves and sends messages over the communication layer, similar to the repository server. We use Laszlo's built-in JavaRPC technology to connect the Flash client to the business logic on the server. This technology offers the possibilty to call service methods on the server easily. These methods include e.g. sendMessageTo...-methods as well as the call to the polling

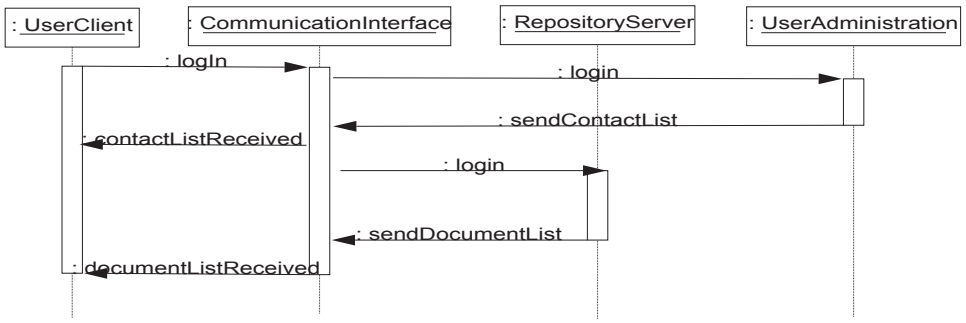

Figure 3: Initialisation Process at User login.

method getQueue(). The DocumentContent event doesn't contain the actual binary data but an HTTP URL, so that we can use Laszlo's implementation of displaying media over the HTTP protocol. This option is configurable for every user client. Some clients may need to get the binary content straight out of the event.

6 CONCLUSION AND FURTHER WORK

Instant Collaboration moves a step ahead from asynchronous resource-sharing and usage of multiple-client applications for communication and collaboration, towards a role-based synchronous sharing of resources using integrated client tools, but accessing proven back-end technologies. In this paper we introduced the concept of *instant collaboration* and presented a first research prototype to support initial instant collaboration patterns. This prototype is partly a flexible middleware to integrate established back-end services with an API to support different client implementations. At the first step, we developed a Flash (Open Laszlo) rich internet client. Further client implementations are planned (Ajax, IM plugin, email integration) as well as enriching the client functionality, e.g., with threaded Chat (Smith et al., 2000) and the option to assign documents/resources to chats.

The next conceptual steps are enriching the middleware side with the integration of more protocols (like WebDAV, LDAP, IMAP), which has been done in the web-application domain in our group, but not for instant collaboration (Kastner, 2005). Additionally, support for ad-hoc workflows, particularly in combination with email and document-flows is planned.

REFERENCES

Apache Licence (2006). Apache open source licence. http://www.apache.org/licenses/LICENSE-2.0.html.

Bhagyavati, B. (2005). Instant messaging usage policies enable ubiquitous communication. In *Wireless Telecommunications Symposium, 2005*, pages 45–48, GA, USA. Columbus State Univ.

BSCW (2003). BSCW basic support for collaborative work. http://www.bscw.de/.

Cryptoheaven (2003). Cryptoheaven: secure collaboration. http://www.cryptoheaven.com/.

Daisy CMS (2006). Daisy content management system. http://cocoondev.org/daisy/index.html.

Geyer, W., Vogel, J., Cheng, L.-T., and Muller, M. (2003). Supporting activity-centric collaboration through peer-to-peer shared objects. In *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*, pages 115–124, New York, NY, USA. ACM Press.

Groove Virtual Office (2006). Microsoft: Groove virtual office. http://groove.net.

Grudin, J. (1988). Why cscw applications fail: problems in the design and evaluation of organization of organizational interfaces. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, pages 85–93. ACM Press.

Hansen, K. M. and Damm, C. H. (2002). Instant collaboration: using context-aware instant messaging for session management in distributed collaboration tools. In *NordiCHI '02: Proceedings of the second Nordic conference on Human-computer interaction*, pages 279–282, New York, NY, USA. ACM Press.

Jackrabbit Content Repository (2006). Jackrabbit jsr-170 content repository. http://jackrabbit.apache.org/.

Jr., R. J. B., Agrawal, R., Gruhl, D., and Somani, A. (2002). Youserv: a web-hosting and content sharing tool for the masses. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 345–354, New York, NY, USA. ACM Press.

Kastner, T. (2005). Document management in distributed project management environments-a new interface approach. Master's thesis, Vienna University of Technology.

Mandviwalla, M. and Olfman, L. (1994). What do groups need? a proposed set of generic groupware requirements. *ACM Trans. Comput.-Hum. Interact.*, 1(3):245–268.

Mecham, J. (2002). Jogger. http://jabber.linux.it/jogger/.

Millen, D. R., Muller, M. J., Geyer, W., Wilcox, E., and Brownholtz, B. (2005). Patterns of media use in an activity-centric collaborative environment. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 879–888, New York, NY, USA. ACM Press.

Muller, M. J., Geyer, W., Brownholtz, B., Wilcox, E., and Millen, D. R. (2004). One-hundred days in an activity-centric collaboration environment based on shared objects. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 375–382, New York, NY, USA. ACM Press.

MyWebdesktop (2006). Mywebdesktop. http://www.mywebdesktop.net/. still in beta.

Nuescheler, D. (2006). Java specification request: JSR-170 content repository. Technical report, Sun Microsystems.

Open Laszlo (2006). Open laszlo; framework to create rich internet applications. http://www.openlaszlo.org/.

Smack XMPP API (2006). Smack xmpp api for java (jive software). http://jivesoftware.org/smack/.

Smith, M., Cadiz, J. J., and Burkhalter, B. (2000). Conversation trees and threaded chats. In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 97–105, New York, NY, USA. ACM Press.

Spark IM Client (2006). Spark jabber instant messaging client (jive software). http://jivesoftware.org/spark/.

SubEthaEdit (2006). Subethaedit: Collaborative editing application. http://codingmonkeys.de/subethaedit/.

Thunderbird (2006). Mozilla thunderbird email client. http://www.mozilla.com/thunderbird/.

Wildfire Server (2006). Wildfire xmpp (jabber) server, jive software. http://jivesoftware.org/wildfire/.

XMPP Specifications (2006). Xmpp specifications. http://www.xmpp.org/.