Journal of Universal Computer Science, vol. 14, no. 1 (2008), 59-87
submitted: 6/11/06, accepted: 17/10/07, appeared: 1/1/08 © J.UCS

Seamless Transition between Connected and Disconnected Collaborative Interaction

Stephan Lukosch
(FernUniversität in Hagen, Department for Mathematics and Computer Science, Germany
stephan.lukosch@fernuni-hagen.de)

Abstract: Nowadays, more and more users make use of web-based collaborative systems. Users participate in communities or search for and provide information in web-based systems. They access shared resources which they need for their professional life or for learning. One of the major prerequisites of such web-based systems is that users have to be connected to the network. But life has become much more mobile over the last years. While traveling, e.g. to the office or the university, users often are disconnected from the network. This makes it difficult to interact with other users or to access shared resources. An application supporting a seamless transition between connected and disconnected phases would allow users to work at any time and place while maintaining the advantages of a web-based collaborative system once they are online again. In this article, we describe the requirements that a web-based collaborative system has to fulfill to enable a nomadic use. We show how we extended the web-based collaborative system CURE to fulfill these requirements and how our approach can be transferred to other web-based collaborative systems.
Key Words: Shared workspaces, collaborative working and learning, nomadic working and learning, interactive situation model
Category: H.1.2, H.5.1, H.5.3, J.7

1 Introduction

In the last years, life has become more and more mobile. A lot of people are traveling not only in their professional life but also in their private life. At the same time, more and more information has become available in the Web. Platforms like, e.g., Wikipedia [Wikipedia, 2007] rely on a large communities that collaboratively construct knowledge. This process is also visible in smaller scenarios. In companies, employees collect company knowledge in web-based systems. At universities, students collaborate during seminars or lab courses in web-based systems.

Knowing that, the FernUniversität developed the web-based collaborative system *CURE* [Haake et al., 2003] which has successfully been in use for about four years now. CURE uses virtual rooms as well as keys to structure collaboration. In these rooms, users can share and create documents. CURE supports various typical use cases in collaborative work, like, e.g., group communication, document sharing, or collaborative writing. Additionally, CURE is used in various collaborative learning scenarios like, e.g., collaborative exam preparation [Lukosch and Schümmer, 2006] or virtual labs [Schümmer et al., 2005].

These use cases include various forms of interaction among the collaborating users. One common model to classify these interaction situations is the *Denver Interactive Situation Model* [Salvador et al., 1996]. This model considers different properties of interactive situations, i.e. dependency of tasks, time of interaction, size of the interacting group, interaction location, and the timing of the interaction. But this model does not consider that nowadays users are more and more mobile. During these times, users are normally disconnected from the Internet. Kleinrock (1996) considers this disconnected state as the usual state instead of an exceptional one. Furthermore, Kleinrock (1996) calls users *nomads* when they often experience the disconnected state and carry their computing and communication devices with them while traveling.

In the sense of Kleinrock (1996), workers who travel from one work place or one customer to another are nomads. At these different places, they often have to access resources in their web-based knowledge repository of their company to accomplish their work. The same is true for students that want to use spare time when traveling for learning or for community member that want to access the community resources not only when they are at home but also when they are mobile. For all these users, it is crucial to access their shared resources not only from home, but also while they are in transit, i.e. *anywhere and anytime* [Kleinrock, 1996].

This vision of anywhere and anytime is currently not supported by web-based collaborative systems. Additionally, it is not part of the *Denver Interactive Situation Model* or other classification models for groupware like the well-known $2x2$ matrix proposed by [Ellis et al., 1991]. In this article, we describe the requirements that we have identified for a nomadic use of web-based collaborative systems. We show how we extended web-based collaborative system CURE to fulfill these requirements and the *Denver Interactive Situation Model* to also reflect the *anywhere and anytime* interactive situations.

We have chosen to extend CURE as the students at the FernUniversität are a very specific clientele. The FernUniversität is a distance teaching university. Apart from regular full-time students, students at the FernUniversität often have full-time positions. Most of the students with full-time employment study as a part-time student. They use every spare time for their studies. Thus, the independence in time and place is one of the major reasons for enrolling in a distance teaching university. Interviews with about 120 students that have used CURE during lectures, seminars, or lab courses in the winter term 2003/2004 revealed a major interest in working nomadically. Since CURE is web-based, spare time during train journeys or flights where no internet access is available cannot be used for reading or editing CURE content. Supporting a nomadic use of CURE would thereby significantly enhance the available learning possibilities.

The remainder of this article is structured as follows: First, we briefly discuss a scenario that describes a common use case of a web-based collaborative system and demonstrates potential advantages of using such a system nomadically (see Section 2). Based on this scenario, we determine the requirements for a nomadic use of web-based collaborative systems. In Section 3, we introduce the web-based collaborative system CURE in more detail while highlighting the deficits for a nomadic use. In Section 4, we describe how we extended CURE to fulfill the requirements for a nomadic use and generalize our approach for web-based collaborative system by using patterns for computer-mediated interaction. We then compare our approach with the state of the art. Finally, we report on our experiences when using the CURE nomadically (see Section 6), before we conclude our paper with a summary and an outlook on future work directions.

2 Requirements analysis

Typical web-based collaborative systems can be characterized by means of patterns[1] for computer-mediated interaction [Schümmer and Lukosch, 2007a]. To enable collaboration, typical web-based collaborative systems offer shared workspaces as Shared File Repositories or Rooms. These shared workspaces rely on Centralized Objects and allow users to cooperate by sharing resources. Prominent examples are wikis [Leuf and Cunningham, 2001], BSCW [Appelt and Mambrey, 1999], or CURE [Haake et al., 2003]. Some systems furthermore support communication and coordination by means of Forums or Embedded Chats. Wikis, e.g., allow to modify the shared resources via the web interface. Other systems only offer a Shared File Repository from which users can download shared resources for modification and later on upload the modified resource. To control who can access and modify the shared resources most systems allow to define Groups.

Such web-based collaborative systems focus on enabling interaction when users are online and connected to the network. They do not consider a nomadic use. It is completely up to the user to prepare for disconnected phases. Users have to manually download content for disconnected phases. Later, when connected to the network again, they have to manually synchronize modified content.

In this section, we describe a typical use case for web-based collaborative systems that offer access-restricted shared workspaces, allow users to modify shared resources directly via the web interface, and offer means for synchronous as well as asynchronous communication. Based on this use case, we determine the requirements for a nomadic use of such web-based collaborative systems. We have chosen a virtual seminar as use case. In such a seminar, students work in small teams up to three members on a given topic. The students collaboratively

[1] Pattern names are set in Small Caps and can be found in [Schümmer and Lukosch, 2007a].

collect references, discuss with their co-students, write the seminar thesis, and review the thesis.

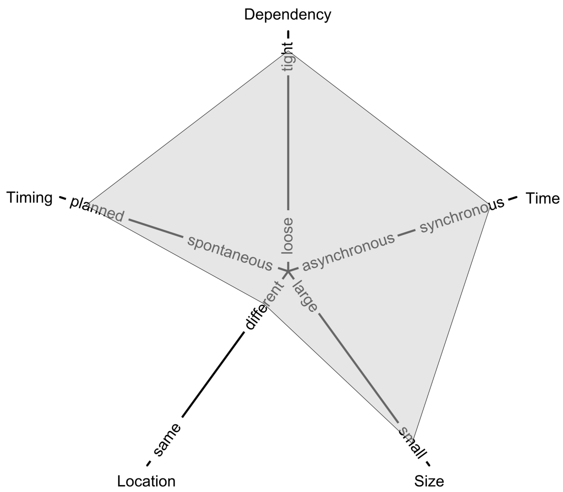

Figure 1: Interactive situations in a seminar

The selected use case makes use of all functional means that are offered by standard web-based collaborative systems. Similar use cases can be found in professional life when small design teams, e.g., collaborate to solve *wicked problems* [Rittel and Webber, 1973] and have to report their results. Figure 1 is based on the *Denver Interactive Situation Model*. It shows the five different properties used to classify interactive situations. The grey area highlights the interactive situations that normally occur in a virtual seminar:

- *Size*: In a seminar, several student teams work on different topics. These topics are often related and the different teams can discuss common issues in a large group. However, specific topic-centered discussion take place in the small team.
- *Location*: In a virtual seminar, the participants are in the most cases geographically distributed, i.e. the interaction takes place at different places.
- *Timing*: The interaction can be planned by, e.g. by communicating via e-mail, but also spontaneously, e.g. by using the chat to contact someone else who is currently online.
- *Dependency*: In a team that works on the same topic the interaction is normally tight. However, there can also be loose interaction between the

different teams when considering the whole seminar.

- *Time*: The interaction has asynchronous phases when team members write sections of the final thesis for which they are responsible, but there are also synchronous phases when team members, e.g., discuss the outline in the chat.

2.1 Virtual Seminar

Heide has just enrolled in a virtual seminar and has been granted access to the main collaborative workspace in the web-based system. This workspace has been created by the supervisors as a central meeting point for the seminar participants. The workspace already contains a number of pages that give a basic overview of the subject of the seminar, the planned workflow, and the timetable. The general subject is split up into a number of sub-topics, each of which should be worked on collaboratively by two students. To reflect this, the workspace contains a page with links to a number of additional workspaces in the web-based system, i.e. one workspace for each sub-topic, and the students have been granted access for their respective workspace.

The supervisors have uploaded some research articles to each workspace that serve as a starting point for further investigations. In addition, a page containing a template for the seminar thesis and a template for collecting literature references is provided. The students are supposed to use their workspace for coordination and communication within their team. They are also encouraged to use the workspace for uploading drafts, intermediary results and the final seminar thesis. The main workspace should be used for communication between all participants of the seminar and as a central repository for pages with information and links of general interest.

Heide would like to read the provided material and then insert some first thoughts right into the thesis draft. Unfortunately, she soon has to leave for a job-related journey. She knows that she will have some hours of spare time on that journey, but she will not be able to access the web-based system during that time. Therefore, she downloads the PDF documents to her notebook to have some reading material. During the journey, Heide has more time than expected, so she starts to work on the thesis by making notes in a simple text editor on her computer. However, since she does not have the provided draft with her, her work remains a coarse collection of thoughts. When returning home, she connects to the network, enters the workspace of her team and notices that her colleague has already inserted some notes into the designated page. Some of these overlap with her work and she has a hard time copying and pasting her sketches to the common page without loosing the existing text.

Heide and her teammate have some long lasting discussions with the teachers about the outline of the thesis. This discussion mainly stems from the teacher noticing a discussion between Heide and her teammate. To avoid such discussions the next time, Heide and her teammate decide to create a private workspace for their topic. They want to use this workspace for private discussions without the teachers.

The next time Heide travels to a customer, she does not need her laptop. However, she still has to take her PDA with her, as she has to coordinate some appointments. Again, she knows that she will have some spare time while traveling. Thus, during the last synchronization with her laptop she adds the latest versions of the pages containing the discussions from the private room to her PDA. During her journey, Heide reads the discussion pages and would like to add some notes to the latest draft of the seminar thesis. However, this is not possible as she only added the discussion pages and not the draft to her PDA.

Finally, the deadline for delivering the final seminar thesis is approaching. Heide and her teammate have still a lot of work to do. Up to now, the whole thesis was represented as one page. Now, they want to work more concurrently and split up the page in several sub-pages, i.e. one page per section. However, for each page they want to create, they have to interact with the server and wait for its response, which can become rather time consuming considering the approaching deadline. Additionally, they have to switch back and forth between the pages and their different workspaces which can become confusing without adequate navigation tools.

2.2 Summary

Web-based collaborative systems focus on providing access using any web browser, with the only requirement being an existing network connection. While this ensures a maximum in flexibility, the user interface is limited due to the possibilities of the web browser. User input has to be sent to the server for processing before the result can be shown within the user interface.

The above scenario highlights some deficits of web-based collaborative systems, when users are nomads. An independent local application on the users computer or PDA can improve the usability by reducing the response time. Furthermore, a local application on a computer can provide a more intuitive user interface compared to the possibilities available in web-based applications. The most important advantage of a local application that can synchronize user data with the server of the web-based collaborative system, and let the user view and edit content locally, is the independence from permanent network access. This allows users to effectively use times of mobility and reduce online costs. Working and learning become possible at any time and place while keeping the advantages of a collaborative environment.

Such a local application enhances the supported interactive situations. We added the property *connectivity* to reflect this in the *Denver Interactive Situation*

Model. Figure 2 shows the extended *Denver Interactive Situation Model*. The grey area highlights the interactive situations in a seminar when students can also work nomadically.

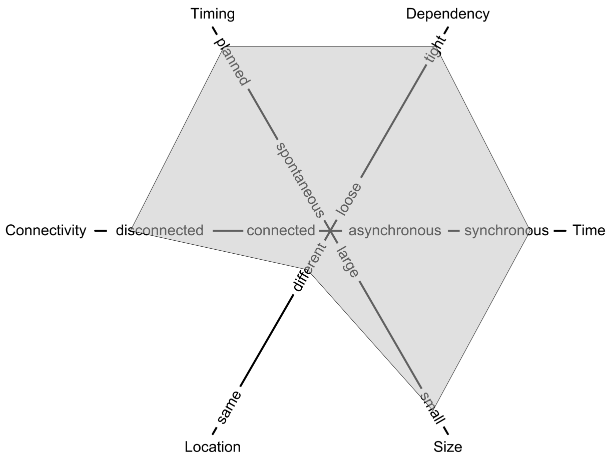

Figure 2: Interactive situations in a seminar when considering connectivity

Summarizing, the following requirements must be fulfilled by an extension of a web-based collaborative system to support these interactive situations:

R1: A communication interface between the server of the web-based collaborative system and a local client on a computer or a PDA has to provide read and write access to the contents of the system. Additionally, users must be able to synchronize the content on their computer with their PDA.

R2: The access rights for group formation as well as access to shared resources must be respected by a local client. This includes authentication of the users.

R3: Users have to be able to individually select the content they want to access locally. The data has to be stored persistently on the users computer or PDA while maintaining the structure existing on the server.

R4: When editing local data the corresponding resources on the server of the web-based system should not be locked, to allow a maximum of concurrency. Therefore, the local application has to provide means for synchronizing local data with the content stored on the server.

R5: Support for the resolution of conflicting changes performed by multiple users on the same page has to be given.

R6: Compared to the web interface, an intuitive user interface has to simplify the creation and editing of content.

3 CURE in a nutshell

In this section, we introduce the web-based collaborative system CURE [Haake et al., 2003]. CURE is used for collaborative work and learning. Typical collaborative learning scenarios are collaborative exercises, tutor-guided groups with collaborative exercises, collaborative exam preparation [Lukosch and Schümmer, 2006], virtual seminars, and virtual labs [Schümmer et al., 2005]. When considering collaborative work typical use cases include group formation, group communication, document sharing, collaborative writing, collaborative task management etc.

From a technical perspective, CURE was built by composing patterns for computer-mediated interaction. For space reasons, we will not go into details of the implementation here, but instead reference to the patterns[2] that were used to create the system and show how they appear in the functional context of user interaction.

Users can structure their interaction in GROUPS that inhabit virtual ROOMS. Room metaphors [Greenberg and Roseman, 2003, Pfister et al., 1998] have been widely used to structure collaboration. Figure 3 shows the abstractions that are offered by CURE. Users enter the cooperative working/learning environment via an entry room that is called *Hall*. Rooms can contain pages, communication channels, e.g. chat, threaded mail, and users. Users, who are in the same room at the same time, can communicate by means of a synchronous communication channel, i.e. by using the chat that is automatically established between all users in the room. They can also access all pages that are contained in the room. Changes of these pages are visible to all members in the room.

The concept of a virtual KEY [Schümmer and Fernandéz, 2006] is used to express access permissions of the key holder on rooms. Each key distinguishes three different classes of rights [Haake et al., 2004]: key-rights defining what the user can do with the key, room-rights defining whether or not a user can enter a room or change the room structure, and interaction-rights specifying what the user can do in the room. Rooms with public keys are accessible by all registered users of the system.

Users can enter a room to access the communication channels of the room and participate in collaborative activities. Users can also create and edit pages in the room. Pages may either be directly edited using a simple wiki-like syntax [Leuf and Cunningham, 2001], or they may contain binary documents or arti-

[2] Again, pattern names are set in SMALL CAPS and can be found in [Schümmer and Lukosch, 2007a] if no other reference is provided.

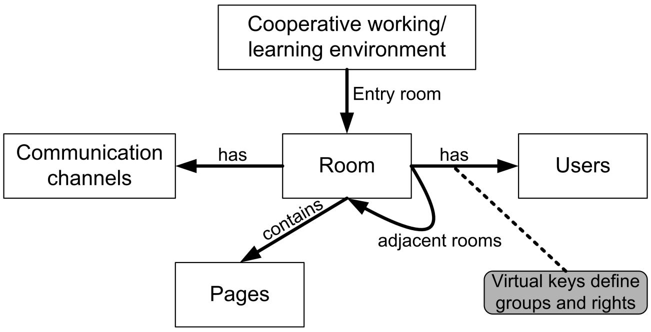

Figure 3: CURE abstractions

facts, e.g. JPEG images, Microsoft Word documents etc. In particular, the syntax supports links to other pages, other rooms, external URLs or mail addresses. The server stores all artifacts to support collaborative access. When users leave the room, the content stays there to allow users to come back later and continue their work on the room's pages. However, this implies that the CURE platform as such requires the user to be connected to the network to access content stored on the CURE server.

Figure 4 shows a typical room in CURE. The numbers in the figure refer to details explained in the following paragraphs. A room contains documents (①, cf. Centralized Objects) that can be edited by those users, who have sufficient edit rights ②. CURE stores all versions of a page as Immutable Versions. Users can browse different versions ③ to understand their colleagues' changes (cf. Timeline). Communication is supported by two room-based communication channels, i.e. a Forum ④ and an Embedded Chat ⑤. Users can use the room-based e-mail to send a mail to the room. Users of the room that have sufficient communication rights will receive this message like being a member of a Mailing List [Schümmer and Lukosch, 2007b].

By providing a plenary room, sharing and communication in a whole class or organization can be supported. By creating new rooms for sub-groups and connecting those to the classes' or organization's room, work and collaboration can be flexibly structured. Starting from the plenary room users can Navigate [Schümmer and Fernandéz, 2006] to the connected sub-rooms ⑥.

For user coordination, CURE supports various types of awareness information:

1. Users can see in the room's properties who else has access to this room ⑦.

2. Users can see in a User List ⑧ who else is currently in the same room.

3. If the Embedded Chat ⑤ is enabled in the room, users can directly start

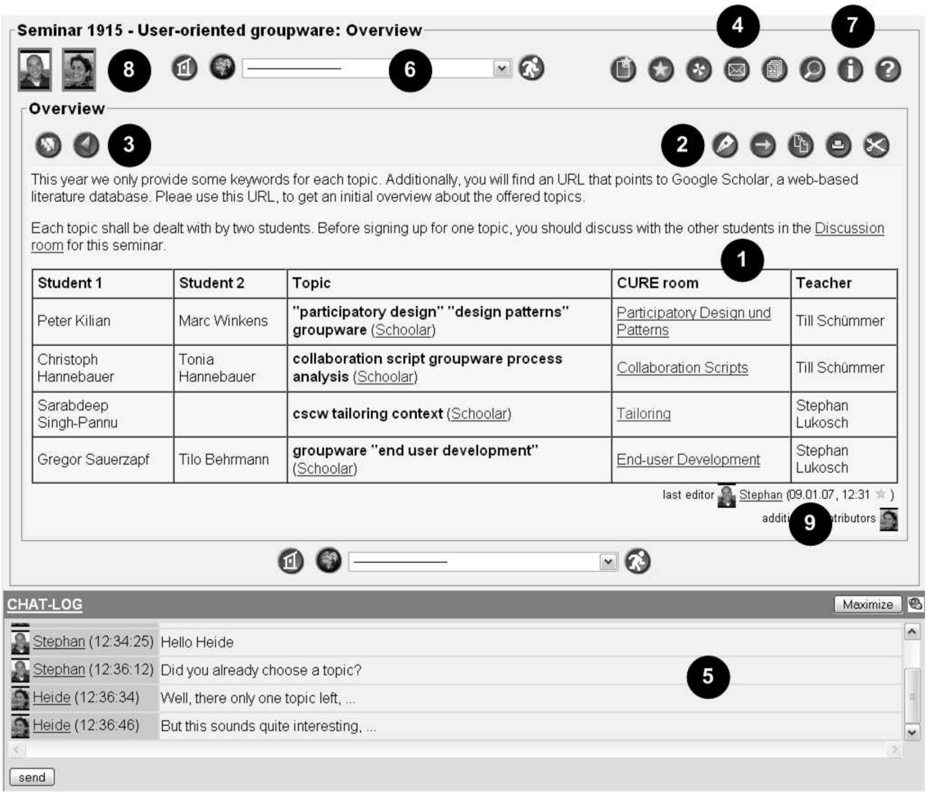

Figure 4: A room in CURE

chatting to each other.

4. Users can trace who has previously edited the current page ⑨ (cf. ACTIVE NEIGHBORS).

5. PERIODIC REPORTS automatically posted to all users of a room include all changes made since the last report was sent.

When considering the above functionality and the extended *Denver Interactive Situation Model* (cf. Figure 2), CURE supports the whole range of all properties apart from the connectivity property. The tasks can be highly dependent or independent. In CURE, users can collaborate synchronously, e.g. by using the chat, as well as asynchronously, e.g. by creating different parts of a shared document as wiki pages. The size of the interacting group can be small or large, as this is defined by the number of users who have access to a virtual room. CURE can be used to support co-located interaction, e.g. during brainstorming, as well as distributed interaction. The timing of the interaction can be planned by using the mail box for coordination as well as spontaneous when triggered by the synchronous awareness functionality.

When considering the connectivity property, CURE does not offer any support. None of the identified requirements R1-R6 is met. Instead, users have to download content manually for use in disconnected phases and later on upload the modified content. In case of conflicts due to parallel modifications, it is up to the users to resolve these. In the following section, we show how CURE can be extended to meet the requirements R1-R6.

4 Approach

In this section, we first describe how we fulfill the identified requirements R1-R6 by extending CURE and implementing *offlineCURE* [Lukosch et al., 2006] as well as *pocketCURE* for seamless transitions between connected and disconnected collaborative interaction. *offlineCURE* as well as *pocketCURE* were implemented as Java applications to be independent from the operating system used by the users. At the end of the section, we summarize our approach and discuss how the approach can be transferred to web-based collaborative systems in general.

4.1 Communication Interface (R1)

To address the requirement R1, we integrated the *CURE communication interface (CCI)* into the CURE server. The *CCI* offers access to CURE data in parallel to the web interface. Figure 5 shows the overall architecture of CURE after being extended with the *CURE communication interface (CCI).* Without *CCI* users can only access CURE content using standard web browsers. The communication between a client and the server is established via HTTP and servlets running on the server respond to the clients requests. CURE content is stored in a database that is accessed by the servlets via the CURE kernel.

Communication between *offlineCURE* and *pocketCURE* is established by using *ActiveSync* [Microsoft Corporation, 2007], a synchronization program developed by Microsoft.

For communication between the *CCI* and its clients, we considered web services as well as a proprietary communication protocol in combination with a direct TCP/IP connection to the server. Web services lead to an open and flexible architecture at the server side. The integration with other applications is simplified due to common standards like HTTP, XML, or SOAP. Additionally, web services reduce possible conflicts with firewalls at the client-side. However, web services also introduce overhead due to the predefined message format which reduces the communication performance. The following text shows a fictive SOAP request for retrieving the page with the name `Welcome` from the room `Hall`:

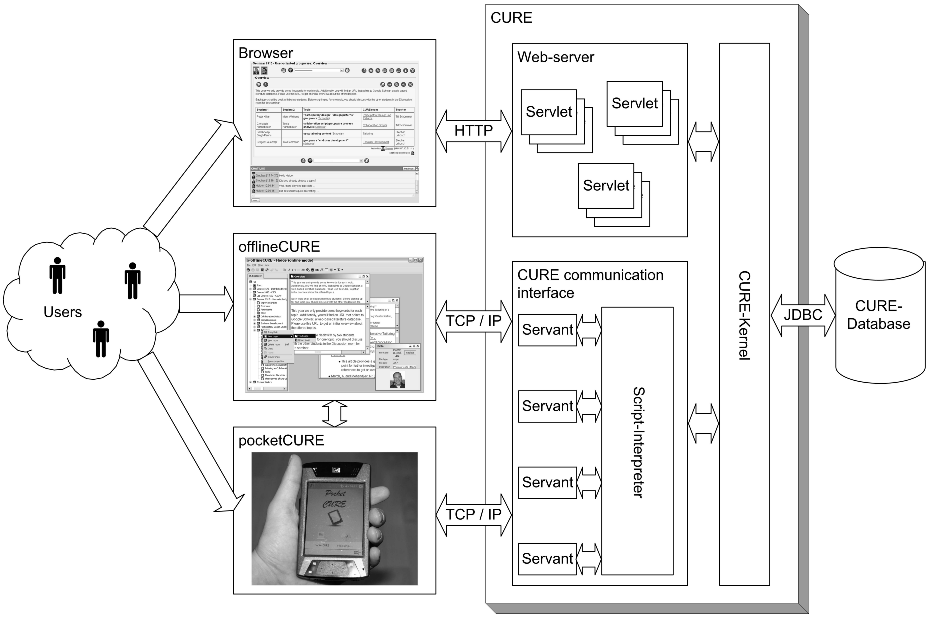

Figure 5: *offlineCURE* architecture

```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body xmlns:m="http://cure.fernuni-hagen.de/services">
    <m:getPage>
      <m:pageName>Welcome</m:pageName>
      <m:roomName>Hall</m:roomName>
    </m:getPage>
  </soap:Body>
</soap:Envelope>
```

This fictive request has a size of 367 bytes, while the necessary data in the SOAP body only requires about 187 bytes. By using a proprietary protocol which makes use of a TCP/IP connection, the amount of bytes for calling the method could be even more reduced, as only the method name and its parameters would have to be transferred, i.e. `getPage`, `pageName Welcome`, and `roomName Hall`. Such a request would only require 38 bytes which reduces the amount of bytes for calling the method `getName` by almost 90 percent. This is a significant reduction. Especially, when considering nomadic interaction where online time is valuable and connection bandwidth can be low. Using web services, the communication furthermore relies on a request and answer paradigm. The client applications have to use specific libraries, like JDOM or SAX, to interact with the *CCI*. These additional libraries reduce the memory space at the client-side which is an important factor when implementing applications for a PDA.

Due to the communication overhead, the additional memory requirements, and the fact that we are not planning to integrate CURE, *offlineCURE*, and *pocketCURE* with other applications, we decided to use a proprietary protocol in combination with a direct TCP/IP connection. The proprietary protocol allows us to reduce the necessary network bandwidth as well as the necessary memory space at the client side. Additionally, the TCP/IP connection allows us to establish a bidirectional communication which has the advantage to offer additional services to the client application, e.g. awareness services. To ensure secure communication, we use a SSL connection. After a client established a connection to the *CCI*, users have to authenticate themselves, before gaining access to the content. Alternatively, users have the chance to authenticate the server with a server-certificate, signed by a certification authority or a trust center.

Communication between the *CCI* and *offlineCURE* as well as *pocketCURE* is handled using a proprietary communication protocol which is based on a intuitive and flexible script language. This language allows to access all necessary methods of the CURE kernel. Clients can send single commands as well as scripts to the *CCI* for processing. The script commands support the download of rooms or pages, the upload of edited pages or rooms, the creation of new pages, rooms or keys, the synchronization of local content with server data, etc. Example script commands are:

- `get page name "Welcome" in room name "Hall";`
  Returns the page with the name *Welcome* in the room *Hall*.

- `get page * in room name "Hall";`
  Returns all pages in the room *Hall*.

- `create room name "Seminar" in room name "Hall";`
  Creates a new sub-room *Seminar* in the room *Hall*.

- `update page name "Welcome" in room name "Hall" -content="...";`
  Updates the page with the name *Welcome* in the room *Hall* with the specified content.

- `create key for room name "My Room" -rigths=124 -freekey;`
  Creates a free key for the room *My Room* with the rights 124.

The servers response can have different output formats, with the standard format being XML. For higher security, the *CCI* filters sensible data, e.g. user passwords, via a filter before sending responses to the clients.

To allow a maximum of concurrency, the server uses one thread for each client that receives and interprets script commands and generates the respective response. To avoid conflicts the *CCI* makes use of the object manager in the CURE kernel. The object manager provides access to the CURE database and ensures consistency by using transactions. *offlineCURE* as well as *pocketCURE* serialize the commands and the *CCI* executes the commands in the received order. Each script command is handled by one transaction. Furthermore, the *CCI* distinguishes between *read*- and *write*-commands for the above-mentioned object manager of the CURE kernel.

In addition to read and write access to CURE content, the *CCI* offers an interface which clients can use to subscribe to notification services. The *CCI* can notify these clients about, e.g., user actions, messages, or news, while the clients are connected to the network. Thereby, the *CCI* allows to provide awareness for *offlineCURE* and *pocketCURE* users that do not want to enter the web-based system, as they are, e.g., only shortly connected to the network again.

4.2 Respecting user rights (R2)

In CURE, users can have different access and interaction rights. For each room users can access, they have a virtual key defining the rights within the respective room [Haake et al., 2004]. These keys are only managed at the CURE server and not at the clients. The *CCI* ensures these user rights by requiring user authentication and by checking the current user rights at the server before executing a script command. Even creating new keys by issuing script commands does not allow to perform malicious operations as the a client can create the necessary key only when the user has the sufficient key-rights anyway.

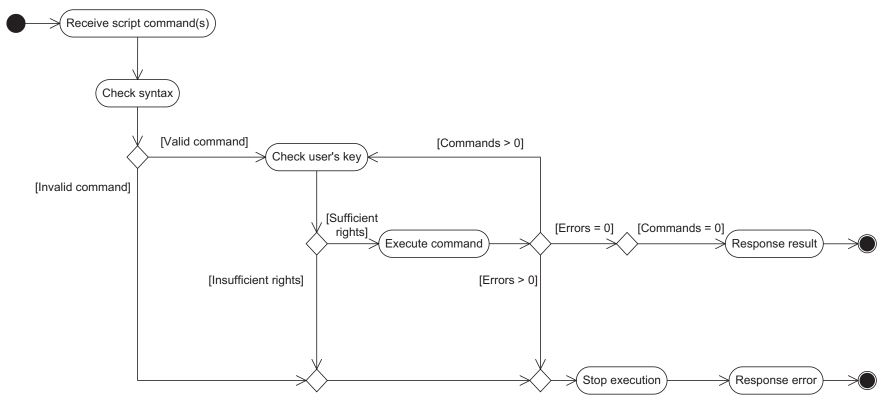

Figure 6: CCI command execution

Figure 6 shows a UML activity diagram describing the execution of a script command by the *CCI*. After receiving one or more commands, the *CCI* checks the syntax of the command to be executed. Since each script command is unambiguously related to a target room, the semantic part of the script interpreter can then check whether the user has sufficient rights to execute the command in the specified room. This guarantees that users can only execute commands via the communication interface that result in actions the user could also perform via the web interface. If a user does not have sufficient rights, the *CCI* responds by sending an error code to the client and stops the transaction. Otherwise, the *CCI* consecutively executes the received commands via the CURE kernel and delivers its results.

4.3 Selection of local data (R3)

Users enter CURE via a main room from where different sub-rooms can be entered. Sub-rooms can again contain further sub-rooms and each room has its own content, i.e. wiki pages or binary pages. Thus, the content of CURE is hierarchically organized and a tree-based visualization is particularly suitable.

For the selection of the CURE content, *offlineCURE* and *pocketCURE* visualize this tree and let users mark the content they want to access nomadically, e.g. a room including all or part of the rooms content, one or multiple pages, or a complete branch of the tree (see Figure 7). To reduce network communication, both client applications only request the information via the *CCI* which is needed to build the currently shown depth of the tree. If a user selects deeper layers, the clients recursively requests the corresponding information.

When the user has selected the desired content for nomadic use, *offlineCURE* and *pocketCURE* download the respective data via the *CCI* and store it locally. Thereby, *offlineCURE* and *pocketCURE* make use of the NOMADIC OBJECTS pattern [Schümmer and Lukosch, 2007a]. Relations between the replicated content objects is maintained, thus providing the same structure of rooms, subrooms and pages as on the server. Subsequently, the user can disconnect from the network and use the replicated data nomadically.

The above approach requires that the PDA has a direct connection to the network. Some PDAs, however, do not provide the possibility to establish a direct connection to the network. Therefore, it is also possible to add CURE content to the PDA by using an export functionality which is integrated in *offlineCURE*. The export functionality allows a user to select the content that is transferred to and later on synchronized with the PDA. For that purpose a similar dialog like in Figure 7 a) is opened and lets the user select the CURE content for the PDA. As mentioned before, the communication between *offlineCURE* and the PDA is handled by using *ActiveSync* [Microsoft Corporation, 2007].

a) *offlineCURE* b) *pocketCURE*

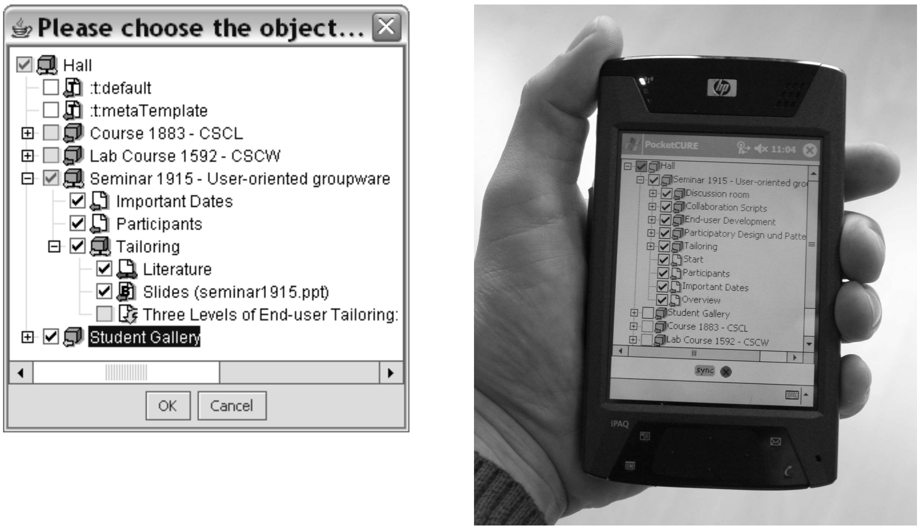

Figure 7: Synchronization dialog in *offlineCURE* and *pocketCURE*

4.4 Synchronization of local data (R4)

Since *offlineCURE* as well as *pocketCURE* are designed to enable the use of CURE content while being completely independent of network connections, both clients cannot directly propagate local modifications to the CURE server and other users. This may result in conflicts between the local version of a document and the corresponding version on the CURE server. One way to avoid this, would be to lock a document on the CURE server whenever a user transfers it to one of the clients. However, this would prevent simultaneous, independent editing and thus contradict the collaborative nature of CURE. The possibility to edit a document would depend, e.g., on the author who has locked it [Vitali and Durand, 1994].

Instead of using such a pessimistic approach, *offlineCURE* and *pocketCURE* use an optimistic one and let users work on separate logical versions of a document, and resolve conflicts when the content is uploaded to (synchronized with) the CURE server. For this purpose, they use an optimistic replication strategy [Saito and Shapiro, 2005], i.e. the original content on the server is not locked and users still have the possibility to modify it using the web interface or edit local copies on their clients. This strategy is also known as Optimistic Concurrency Control [Schümmer and Lukosch, 2007a].

In CURE, version management uses timestamps that are assigned to CURE content when it is created. For Conflict Detection [Schümmer and Lukosch, 2007a] between client and server data, *offlineCURE*

Server	Client	Synchronization action
t_1	n/a	Content only available on the server; download to client
n/a	t_1/+	Content only available on the client; upload to server
t_1	t_1/-	Content unmodified; no synchronization necessary
t_1	t_1/+	Content modified on the client; upload to server
t_2	t_1/-	Content modified on the server; download to client
t_2	t_1/+	Content modified on server and client; conflict resolution

t_1, t_2: Consecutive timestamps on the CURE server
n/a: Document is not available at this site
- / +: Document was not changed / changed by a client

Table 1: Possible combinations of timestamp and modified bit with corresponding synchronization action

and *pocketCURE* use a combination of these server-based timestamps and client-based *modified bits* [Saito and Shapiro, 2005]. While the timestamps included in the local copy of content objects are never changed by the client, modified bits are set whenever a user changes a local document. Table 1 shows possible combinations of the server-based timestamp and the client-based modified bit. When users are connected again and start a synchronization, the clients analyze the local content and communicate with the *CCI* to determine the necessary synchronization actions. The result of this analysis and communication is visualized in a tree. In this tree, users can select which synchronization actions they want to perform (see Figure 7), i.e. upload to the server, download to the client, or conflict resolution (cf. Section 4.5).

4.5 Conflict resolution support (R5)

The optimistic replication strategy used by *offlineCURE* as well as *pocketCURE* can lead to conflicting modifications. Already in CURE, conflicting modifications occur. Depending on the collaboration scenario, these conflicts can be quite obtrusive for the user. In our workgroup, we, e.g., use a CURE page to coordinate oral exams. This page mainly consists of a huge table listing the oral exams and the corresponding examiners. The examiners regularly check the page and confirm or change the exams. Conflicts on this page happen quite often. We expect that the number of such conflicts will increase with the use of *offlineCURE* and *pocketCURE*, especially as there are no awareness mechanisms for disconnected synchronous work that allow to establish a social protocol.

Different approaches can be employed to solve such conflicts. Operational transformation approaches [Sun and Ellis, 1998] are often used in groupware applications to automatically solve such conflicts. These approaches require that the conflicting changes can be described as fine-grained operations which are then transformed to lead to an consistent state. However, the nature of web-based collaborative applications does not allow to describe the changes as fine-grained commands. Another possibility would be to a use a difference algorithm, e.g. [Myers, 1986], and use the result to automatically merge the conflicting versions. Though this approach can lead to a page that contains the modifications from both versions, it cannot guarantee that the merged version is semantically correct.

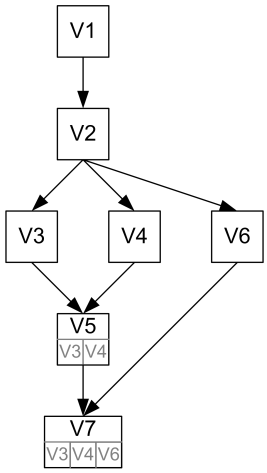

Figure 8: Version tree in CURE

To ensure the semantical correctness, CURE as well as the client applications use a different approach. CURE uses the IMMUTABLE VERSIONS pattern [Schümmer and Lukosch, 2007a] and stores all versions of a document in a version tree. Thereby, it ensures that no version of a document gets lost. Figure 8 shows the evolution of such a version tree in CURE. A newly created document has the version *V1*. When a document is changed for the first time, the version number is changed to *V2*. When this version is changed in parallel, the parallel versions are assigned the version numbers *V3*, *V4*, and *V6*. When the user that created the version *V4* stores the version on the CURE server, the server automatically creates the version *V5* and requests the user to manually resolve the conflict and thereby keep the semantical correctness of the page. *V5* is called *merge-page* and contains the versions *V3* as well as *V4*. The version *V7* shows an even more complicated conflict that emerged from a third user editing *V2* in parallel.

When using the web-based interface of CURE, these merge-pages, like *V5* or *V7*, consist of the conflicting pages and are separated by text that points out the conflicts and requests the user to solve them. CURE as well as *pocketCURE*, use this approach to manage conflicts that might occur when users modify the same version of a document via the web interface. This way, CURE and *pocketCURE* prevent users from overwriting the changes other users have performed and ensures semantical correctness of the pages.

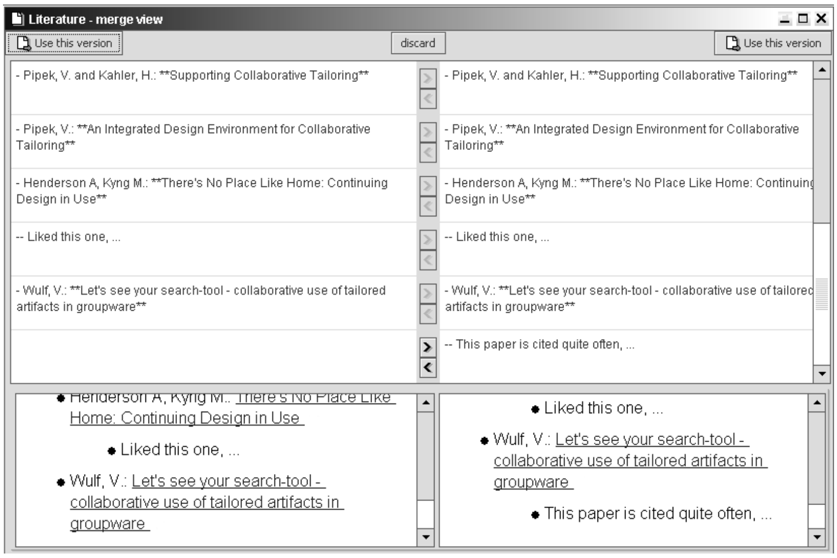

Figure 9: Merge view in *offlineCURE*

Solving these conflicts via the web interface of CURE or on the small display of a PDA, in case of *pocketCURE*, can be quite difficult. To better support users when solving conflicts, *offlineCURE* employs a different visualization for merging pages. First, users can select the versions they wants to merge. The text of the selected versions is then compared blockwise and both versions are aligned in an editable form as shown in Figure 9). Differing blocks are highlighted in the user interface and users can select which block they want to use in the merged version, or edit one version manually. When a user has solved a conflict, the merged document is uploaded to the server resulting in a new version.

4.6 User interface (R6)

Figure 10 shows the user interface of *offlineCURE*. The left area of the user interface shows a tree-based overview of the content that is locally available ①. Compared to the web interface, this simplifies the navigation, since users always have an overview of the local content available. The level of detail can be tailored by expanding or collapsing individual tree branches and the different content types are visualized by using different symbols. Context-sensitive menus allow users to perform actions in relation to the selected object, e.g. create a new sub-room in a room or delete a page ②. Additionally, the *offlineCURE* user interface offers statical menus, keyboard shortcuts, and a toolbar which provides support for incorporating wiki-tags into the page text ③.

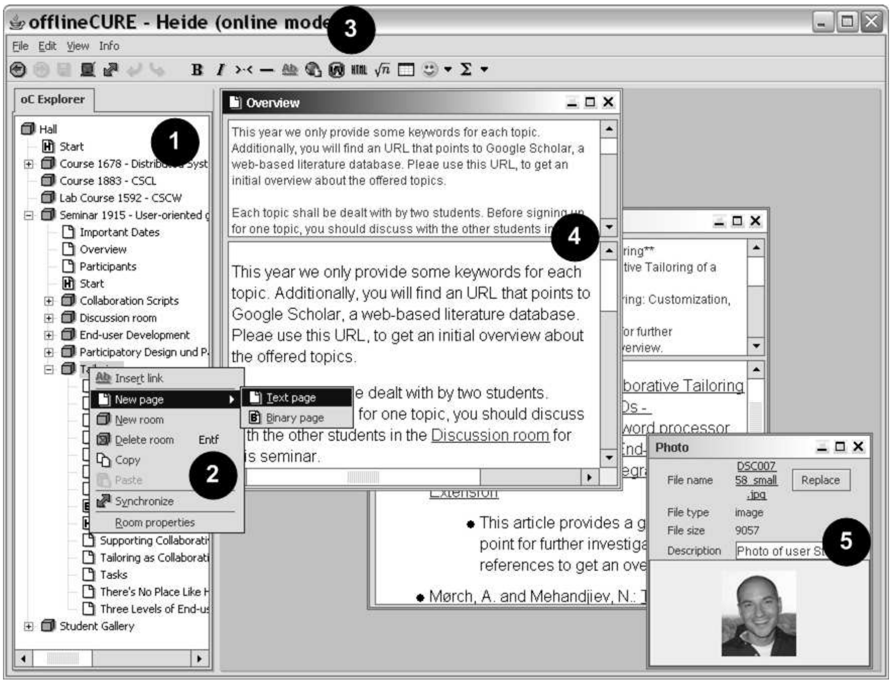

Figure 10: *offlineCURE* user interface

Figure 10 also shows three windows in which the user currently edits CURE pages ④ or views a binary page ⑤. Both edit windows are split horizontally. The upper part is used for editing Wiki pages while the lower shows a real-time preview of the rendered page. The size of the edit and preview areas is variable, i.e. users can decide to display only the edit area while working on a long document, or use the preview area on its own for browsing CURE content offline. The real-time preview shows the page content like it would be rendered in the users browser when working with the CURE server. This also includes usable links to other documents, external URLs etc. Support for creating complex structures is given by indicating whether internal links on a page are valid, i.e. whether the linked content is available locally, and allowing users to conveniently create content via context sensitive menus attached to links. Compared to the web interface, the *offlineCURE* user interface thereby significantly improves the usability and workflow when editing CURE content.

Figure 11 shows the user interface of *pocketCURE*. Due to the limited size of the display, the functionality is split up in several windows. Figure 11 a) shows the tree view of the local CURE content ①. A context-sensitive menu allows to edit, preview, rename, etc. a selected document ②. Additional menus ③, let users perform basic editing operations or synchronize the local content with the CURE server.

a) Tree view b) Edit view

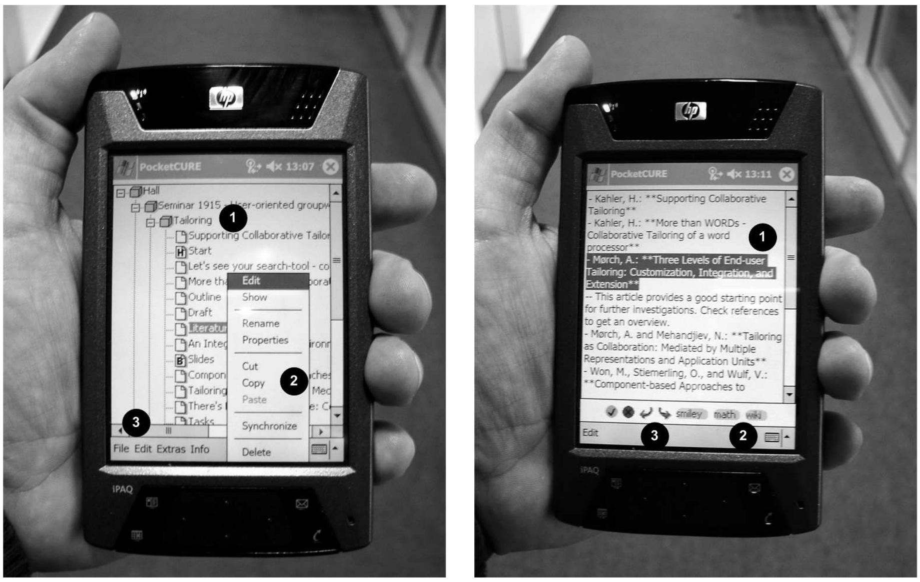

Figure 11: *pocketCURE* user interface

Figure 11 b) shows the edit view of *pocketCURE* in which users can edit the selected page ①. To support the editing process *pocketCURE* offers menus that provide support for incorporating wiki-tags into the page text ②. Additionally, *pocketCURE* supports an undo and a redo of performed editing operations ③.

Thereby, *pocketCURE* enables users to view and edit selected CURE content on their PDA.

4.7 Summary

The previous sections described how CURE has been extended to support a seamless transition between connected and disconnected phases. Now, we transfer the taken approach to web-based collaborative systems in general.

In a first step, the web-based collaborative system has to be extended with a communication interface which allows to access the shared resources. For the communication interface, two design options exist. First, the communication interface can make use of TCP/IP and rely on a proprietary protocol to speed up communication and offer awareness services for the client application. Second, the communication interface can make use of web services if the users of the client applications have difficulties to directly access the internet. As discussed, the web service solution introduces overhead in the communication, does not allow to establish direct communication links between the server and the client applications, and requires additional memory space at the client-side.

If the web-based collaborative system supports mechanisms for access control, it is important that the communication interface respects the defined user rights. It must not be possible that nomadic users gain access to shared resources which they are not allowed to access. For that purpose, the communication interface has to require a user authentication. Based on the authentication, the communication interface has to check whether a user is allowed to perform the requested operations. As the communication interface and not the client application checks the user rights, it is not possible for the user to locally manipulate its user rights while working disconnected.

Normally, users do not need all shared resources while disconnected and their devices, especially PDAs, often have limited space. Therefore, the client applications have to support users in selecting the content for the disconnected phases. In most web-based collaborative systems, the shared resources are organized hierarchically and thus a tree-based visualization is particularly suitable for the representation of the shared workspaces (nodes) and shared resources (leafs).

The shared resources have to be replicated for use in disconnected phases by using the Nomadic Objects pattern [Schümmer and Lukosch, 2007a]. An Optimistic Concurrency Control [Schümmer and Lukosch, 2007a] approach ensures that the shared resources can still be modified via the web interface of the collaborative system. As this approach can lead to diverging shared resources at the client-side as well as at the server-side, the client applications have provide means for Conflict Detection [Schümmer and Lukosch, 2007a] and synchronizing local shared resources with the resources stored on the server. Resources that have to be synchronized can easily be detected by using modified bits at the client-side and version numbers at the server-side.

Apart from diverging shared resources, the optimistic replication approach can also lead to conflicting modifications of the same shared resource. Approaches that automatically resolve conflicts can only ensure the syntactical but not the semantical correctness of a shared resource. Therefore, a versioning approach like the IMMUTABLE VERSIONS pattern [Schümmer and Lukosch, 2007a] that keeps all modifications and lets users manually resolve the conflicts to ensure semantical correctness is most suitable. Client applications should offer means that support users when resolving conflicts.

Finally, the client applications should offer an intuitive user interface that makes use of common single-user application concepts, like,e.g., the tree visualization of the shared resources to simplify the navigation or the real-time preview of the rendered wiki page.

5 Related Work

There has a lot of work been done in order to support disconnected work. Systems like *Bayou* [Terry et al., 1995], *Rover* [Joseph et al., 1997], *Sync* [Munson and Dewan, 1997], or *DOORS* [Preguiça et al., 2000] offer a shared object management which allows users to disconnect and work nomadically. However, none of these systems focusses on extending a web-based system for nomadic work or even to support different devices. In this section, we compare *offlineCURE* and *pocketCURE* with web-based collaborative systems or wiki engines that offer a nomadic use.

BSCW [Appelt and Mambrey, 1999] and *BSCL* [Stahl, 2002] are web-based collaboration platforms. BSCW allows to download documents, edit them locally, and then upload the modified document. *BSCWeasel* [Stevens et al., 2004] allows to access a BSCW server without using the web interface, but still requires a permanent network connection. BSCW, BSCL, and BSCWeasel do not support PDAs or the synchronization of documents, when documents were modified by another user.

The learning platform *KOLUMBUS* [Herrmann and Kienle, 2003] allows users to import and export content, but like BSCW, BSCL, and BSCWeasel it does not support PDAs or resolving conflicts that are due to parallel changes.

Blackboard Backpack [Blackboard Inc., 2007a] is a client-side software application that enables users to download content, e.g. course documents, announcements, calendar items, or tasks, from the *Blackboard* learning platform [Blackboard Inc., 2007b]. However, the content can only be read and annotated locally, i.e. local changes cannot be synchronized. Additionally, the minimum system requirement is a Tablet PC and thus PDAs are not supported.

FirstClass [FirstClass, 2007] is a commercial groupware similar to CURE. But compared to CURE, FirstClass does not offer a versioning support. FirstClass allows users to lock documents when they want to modify it. While a document is locked, other users can only read it. Due to this, FirstClass does not fulfill our requirements concerning conflict resolution and synchronization. FirstClass also offers a PDA client, but this client does not support to edit the content pages. Instead it, e.g., allows to view mails or collected contact information.

eBag [Brodersen et al., 2005] offers a *digital schoolbag* which can contain images, videos, music, or text documents for learning at school. eBag allows pupils to move nomadically between stationary terminals, called *digital oases*, in classrooms, labs, libraries, etc. In these digital oases, pupils can access their documents. Out of these digital oases, the digital schoolbag can be accessed via a web browser. Though the developers call this offline use, the latter obviously requires a network connection. Therefore, eBag does not fulfill our requirements concerning nomadic use.

The collaborative learning platform *sTeam* [Eßmann et al., 2004] allows students to learn collaboratively without being connected to the learning platform. For that purpose, the students have to establish an ad-hoc network among themselves. Thus, sTeam allows to disconnect from the learning platform, but students have to meet at the same place and the same time for collaboration. Individual times of mobility cannot be used for learning.

Personal Wikis, e.g. *WikiWriter* [HyText Consulting, 2007], *EclipseWiki* [EclipseWiki, 2007] or *WikidPad* [WikidPad, 2007], allow to setup a local wiki engine and locally manage, create, and modify the content of this wiki. The content of these personal wikis is stored in a local database. Thus, nomadic working and learning is possible, but as the content cannot be shared, users cannot collaborate.

SimpleWikiEditMode [SimpleWikiEditMode, 2007] differs from the above personal wikis. It uses the text editor *Emacs* to access and modify the content of a remote wiki. However, SimpleWikiEditMode does not support the selection of local content. Additionally, it does not support PDAs and does not provide functions for identifying locally changed content or conflict resolution.

The *Wikipedia Editor* [PLog4U, 2007] is a plugin for the IDE *Eclipse*. With the help of this plugin, users can locally store and modify Wikipedia articles. Compared to our requirements, the plugin does not support PDAs, conflict detection, and conflict resolution. During synchronization, articles that were changed locally are simply replaced with the current version in the Web.

To summarize, none of the above systems fulfills our requirements for a seamless transition between connected and disconnected collaborative interaction. CURE together with *offlineCURE* and *pocketCURE* represent a significant step forward, as they allow users to work and continue their interaction with other users while they are on the move and disconnected from the network. As soon as the users are connected again, they can synchronize their nomadic results with the web-based collaborative system and thereby seamlessly integrate these in their collaboration process.

6 Experiences

We split up the evaluation of *offlineCURE* and *pocketCURE* in three phases. In the first phase, we have conducted functional tests. These functional tests concentrated on validating the intended use cases and checking if our requirements are met. For testing the *CCI*, we developed a console application which allowed us to interact with the *CCI* without using *offlineCURE* or *pocketCURE*. This console application allowed us to define test scripts and check the robustness and performance of the *CCI*. In these scripts, we also tried to execute malicious commands but checking the access rights at the server prevented the execution of these commands. The tree representation worked well for selecting the content for nomadic use. Especially, when setting up the test server with a large database, the recursive approach to build the tree at the client-side significantly reduced the network communication. To test the versioning approach, we have run test scripts which concurrently modified pages at the server. These tests constructed up to ten parallel versions and showed the stability of the versioning approach. However, merging these pages was only possible via the *offlineCURE* user interface, as users had problems to keep the overview about the different page versions in the web-based user interface or the PDA user interface.

In a second phase, we have setup a test environment in our group and let several groups test the system. Feedback from these users with different background, i.e. expert as well as lay users, indicates that *offlineCURE* and *pocketCURE* satisfy all of our requirements. Users have reported that the client applications simplify typical use cases of daily work and learning as the necessary content is now available anywhere and anytime. Furthermore, users have told that they prefer to use *offlineCURE* for creating, structuring, and browsing CURE content, even when connected to the network. This shows that the user interface of *offlineCURE* highly improves the usability when navigating, creating, or editing CURE content. *pocketCURE* is often used when a laptop is not available. In these cases, users mainly use *pocketCURE* to browse CURE content and to lookup information. Users only sometimes modified content when using *pocketCURE*. Additionally, large documents were only rarely created with *pocketCURE*. However, users have told that this was mainly due to the inconvenient input possibilities of the PDA.

CURE is regularly used by our group to conduct seminars as well as lab courses with a blended CSCL approach [Haake et al., 2005]. During these seminars and lab courses, we gather a lot of log data which allows us to analyze the interaction among the students. In the final evaluation phase of *offlineCURE* and *pocketCURE*, we plan to use these applications in seminars and lab courses which follow the same blended CSCL approach. Based on the gathered interaction data with or without nomadic interaction possibilities, we plan to evaluate how the nomadic interaction possibilities affect the interaction among the users. Especially, we want to investigate if users more often lookup information, if users more often create or modify pages, if users concentrate on modifying their own pages or if they also modify other users' pages, and if the number of conflicting modifications increases. In [Schümmer et al., 2005], we already reported that the student groups with a higher communication load produced better results in the lab courses. We expect that the number of created pages and modifications and thus the interaction among the users increases. We want to investigate if the results of the lab courses improve when *offlineCURE* as well as *pocketCURE* increase the interaction.

7 Conclusions

Life has become more and more mobile. In combination with the advent of web-based collaborative systems in private as well as professional life, this increases the demand for a seamless transition between connected and disconnected collaborative interaction. In the sense of Kleinrock (1996), users that require such interaction possibilities are called nomads. When considering the specific clientele of the FernUniversität, one can easily recognize that most of the students learn while they are disconnected from the network and thus are nomads.

In this article, we first have identified general requirements for web-based collaborative systems which must be met to support a seamless transition between connected and disconnected collaborative interaction. Then, we have presented how we extended the web-based collaborative system CURE to fulfill these requirements. The CURE extensions *offlineCURE* and *pocketCURE* use the CURE communication interface *CCI* to selectively download content. They respect the access rights that are represented as virtual keys in CURE. Users can download all content that is accessible for them. They can view and modify the locally available content. For that purpose, *offlineCURE* and *pocketCURE* apply an optimistic replication strategy. As this may lead to conflicting modifications while users are disconnected, *offlineCURE* and *pocketCURE* support users in solving such conflicts.

Our approach can easily be generalized for other web-based collaborative systems. To support nomadic work, a web-based collaborative system has to offer a communication interface that allows to access its content. An optimistic replication strategy which allows users to select content for nomadic use in combination with a version control for the content of the system allows users to modify the content while disconnected. As this might increase the number of conflicting modifications, the tool for nomadic work should offer conflict resolution functionality. Finally, if the web-based system distinguishes different access rights these should only be managed at the server of the web-based system to prevent malicious actions.

Experiences have shown that all of our requirements are satisfied. However, we plan use CURE together with its extensions *offlineCURE* and *pocketCURE* to evaluate how seamless interaction possibilities affect the created content and the results of collaborating students. Additionally, we are currently working on a Web 2.0 approach to improve the usability of the web interface of CURE concerning navigation, tailoring, and awareness [Bourimi et al., 2007]. This includes web-based user support for merging conflicting pages. Finally, we are currently identifying basic collaborative services which have to be offered by a web-based collaborative system to enable computer-mediated interaction. We plan to implement these services as web services and support a context-based orchestration and choreography of these services to improve the interaction among collaborating users. We will include the *CCI* in this generic interface. This will even broaden the usage possibilities of *offlineCURE* and *pocketCURE*.

Acknowledgements

Special thanks are due to Matthias Hellweg and Martin Rasel for their effort in implementing *offlineCURE* and Frank Plieninger for implementing *pocketCURE*.

References

[Appelt and Mambrey, 1999] Appelt, W. and Mambrey, P. (1999). Experiences with the BSCW shared workspace system as the backbone of a virtual learning environment for students. In *Proceedings of ED-MEDIA99*.

[Blackboard Inc., 2007a] Blackboard Inc. (2007a). Blackboard backpack. http://backpack.blackboard.com/.

[Blackboard Inc., 2007b] Blackboard Inc. (2007b). Product homepage. http://www.blackboard.com/.

[Bourimi et al., 2007] Bourimi, M., Lukosch, S., and Kühnel, F. (2007). Leveraging visual tailoring and synchronous awareness in web-based collaborative systems. In Haake, J. M., Ochoa, S. F., and Cechich, A., editors, *Groupware: Design, Implementation, and Use, 13th International Workshop, CRIWG 2007*, LNCS 4715, pages 40–55. Springer-Verlag Berlin Heidelberg.

[Brodersen et al., 2005] Brodersen, C., Christensen, B. G., Grønbæk, K., Dindler, C., and Sundararajah, B. (2005). eBag: a ubiquitous web infrastructure for nomadic learning. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 298–306, Chiba, Japan. ACM Press.

[EclipseWiki, 2007] EclipseWiki (2007). Eclipse Wiki Editor Plugin. http://eclipsewiki.sourceforge. net/.

[Ellis et al., 1991] Ellis, C., Gibbs, S., and Rein, G. (1991). Groupware some issues and experiences. *Communications of the ACM*, 34(1):38–58.

[Eßmann et al., 2004] Eßmann, B., Hampel, T., and Bopp, T. (2004). A network component architecture for collaboration in mobile settings. In *ICEIS 2004, Sixth International Conference on Enterprise Information Systems*, pages 337–343, Porto, Portugal.

[FirstClass, 2007] FirstClass (2007). Product homepage. http://www.firstclass.com.

[Greenberg and Roseman, 2003] Greenberg, S. and Roseman, M. (2003). Using a room metaphor to ease transitions in groupware. In Ackermann, M., Pipek, V., and Wulf, V., editors, *Sharing Expertise: Beyond Knowledge Management*, pages 203–256. MIT Press, Cambridge, MA, USA.

[Haake et al., 2004] Haake, J. M., Haake, A., Schümmer, T., Bourimi, M., and Landgraf, B. (2004). End-user controlled group formation and access rights management in a shared workspace system. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 554–563. ACM Press, New York, NY, USA.

[Haake et al., 2005] Haake, J. M., Haake, A., Schümmer, T., and Lukosch, S. (2005). Collaborative learning at a distance with the project method. *Educational Technology*, 45(5):21–24.

[Haake et al., 2003] Haake, J. M., Schümmer, T., Haake, A., Bourimi, M., and Landgraf, B. (2003). Two-level tailoring support for CSCL. In Favela, J. and Decouchant, D., editors, *Groupware: Design, Implementation, and Use, 9th International Workshop, CRIWG 2003*, LNCS 2806, pages 74–82. Springer-Verlag Berlin Heidelberg.

[Herrmann and Kienle, 2003] Herrmann, T. and Kienle, A. (2003). KOLUMBUS: Context-oriented communication support in a collaborative learning environment. In van Weert, T. J. and Munro, R. K., editors, *Informatics and the Digital Society. Social, Ethical, and Cognitive Issues*, pages 251–260. Kluwer.

[HyText Consulting, 2007] HyText Consulting (2007). Wikiwriter. http://hytext.com/ww/.

[Joseph et al., 1997] Joseph, A. D., Tauber, J. A., and Kaashoek, M. F. (1997). Mobile computing with the rover toolkit. *IEEE Transactions on Computers*, 46(3):337–352.

[Kleinrock, 1996] Kleinrock, L. (1996). Nomadicity: anytime, anywhere in a disconnected world. *Mobile Networks and Applications*, 1(4):351–357.

[Leuf and Cunningham, 2001] Leuf, B. and Cunningham, W. (2001). *The WIKI way*. Addison-Wesley, Boston, MA, USA.

[Lukosch et al., 2006] Lukosch, S., Hellweg, M., and Rasel, M. (2006). CSCL, Anywhere and Anytime. In Dimitriadis, Y. A., Zigurs, I., and Gómez-Sánchez, E., editors, *Groupware: Design, Implementation, and Use, 12th International Workshop, CRIWG 2006*, LNCS 4154, pages 326–340. Springer-Verlag Berlin Heidelberg.

[Lukosch and Schümmer, 2006] Lukosch, S. and Schümmer, T. (2006). Making exam preparation an enjoyable experience. *International Journal of Interactive Technology and Smart Education, Special Issue on 'Computer Game-based Learning'*, 3(4):259–274.

[Microsoft Corporation, 2007] Microsoft Corporation (2007). Windows Mobile - Device Synchronization - ActiveSync and Windows Mobile Device Center. http://www.microsoft.com/windowsmobile/activesync/.

[Munson and Dewan, 1997] Munson, J. P. and Dewan, P. (1997). Sync: A java framework for mobile collaborative applications. *IEEE Computer*, 30(6):59–66.

[Myers, 1986] Myers, E. W. (1986). An o(nd) difference algorithm and its variations. *Algorithmica*, 1(2):251–266.

[Pfister et al., 1998] Pfister, H.-R., Schuckmann, C., Beck-Wilson, J., and Wessner, M. (1998). The metaphor of virtual rooms in the cooperative learning environment clear. In Streitz, N., Konomi, S., and Burkhardt, H., editors, *Cooperative Buildings - Integrating Information, Organization and Architecture. Proceedings of CoBuild'98*, LNCS 1370, pages 107–113. Springer Heidelberg.

[PLog4U, 2007] PLog4U (2007). Eclipse Wikipedia Plugin. http://www.plog4u.org/index.php/Using_Eclipse_Wikipedia_Editor.

[Preguiça et al., 2000] Preguiça, N., Martins, J. L., Domingos, H., and Duarte, S. (2000). Data management support for asynchronous groupware. In *Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work*, Philadelphia, Pennsylvania, USA. ACM.

[Rittel and Webber, 1973] Rittel, H. W. J. and Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, 4:155–169.

[Saito and Shapiro, 2005] Saito, Y. and Shapiro, M. (2005). Optimistic replication. *ACM Computing Surveys*, 37(1):42–81.

[Salvador et al., 1996] Salvador, T., Scholtz, J., and Larson, J. (1996). The Denver model for groupware design. *SIGCHI Bulletin*, 28(1):52–58.

[Schümmer and Fernandéz, 2006] Schümmer, T. and Fernandéz, A. (2006). Patterns for virtual places. In Longshaw, A. and Zdun, U., editors, *Proceedings of the Tenth European Conference on Pattern Languages of Programs (EuroPLoP'05)*, pages 35–74. UVK Universitätsverlag Konstanz GmbH.

[Schümmer and Lukosch, 2007a] Schümmer, T. and Lukosch, S. (2007a). *Patterns for Computer-Mediated Interaction.* John Wiley & Sons, Ltd.

[Schümmer and Lukosch, 2007b] Schümmer, T. and Lukosch, S. (2007b). READ.ME – Talking about computer-mediated communication. In Zdun, U. and Hvatum, L. B., editors, *Proceedings of the 11th European Conference on Pattern Languages and Programs (EuroPLoP'06)*, pages 317–342. UVK Universitätsverlag Konstanz GmbH.

[Schümmer et al., 2005] Schümmer, T., Lukosch, S., and Haake, J. M. (2005). Teaching distributed software development with the project method. In Koschmann, T., Suthers, D. D., and Chan, T.-W., editors, *Computer Supported Collaborative Learning 2005: The Next 10 Years!*, pages 577–586. Lawrence Erlbaum Associates.

[SimpleWikiEditMode, 2007] SimpleWikiEditMode (2007). Product Homepage. http://www.emacswiki.org/cgi-bin/wiki/SimpleWikiEditMode.

[Stahl, 2002] Stahl, G. (2002). Groupware goes to school. In Haake, J. M. and Pino, J. A., editors, *Groupware: Design, Implementation, and Use, 8th International Workshop, CRIWG 2002*, LNCS 2440, pages 7–24, La Serena, Chile. Springer-Verlag Berlin Heidelberg.

[Stevens et al., 2004] Stevens, G., Budweg, S., and Pipek, V. (2004). The "BSCWeasel" and Eclipse-powered cooperative end user development. In *Proceedings of the Workshop "Eclipse as a Vehicle for CSCW Research" at the International Conference on CSCW 2004.*

[Sun and Ellis, 1998] Sun, C. and Ellis, C. (1998). Operational transformation in real-time group editors: Issues, algorithms, and achievements. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work*, pages 139–148, Seattle, Washington, USA.

[Terry et al., 1995] Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J., and Hauser, C. H. (1995). Managing update conflicts in bayou, a weakly connected replicated storage system. In *Proceeding of the Fifteenth ACM Symposium on Operating System Principles*, pages 172–183, Copper Mountain Resort, CO, USA.

[Vitali and Durand, 1994] Vitali, F. and Durand, D. G. (1994). Using versioning to support collaboration on the WWW. *World Wide Web Journal*, 1(1):37–50.

[WikidPad, 2007] WikidPad (2007). Wikidpad - wiki notebook/outliner for windows. http://www.jhorman.org/wikidPad/.

[Wikipedia, 2007] Wikipedia (2007). Main page — wikipedia, the free encyclopedia. http://en.wikipedia.org/. [Online; Stand 24. April 2007].