DEALING WITH CONFLICTING MODIFICATIONS IN A WIKI

Stephan Lukosch and Andrea Leisen
Department of Mathematics and Computer Science, FernUniversität in Hagen
Universitätsstraße 1, 58084 Hagen, Germany

Keywords: Wiki, versioning, page comparison, page merging.

Abstract: Collaborative web-based applications support users when creating and sharing information. Wikis are prominent examples for that kind of applications. Wikis, like e.g. *Wikipedia* (Wikipedia, 2007), attract loads of users that modify its content. Normally, wikis do not employ any mechanisms to avoid parallel modification of the same page. As result, conflicting changes can occur. Most wikis record all versions of a page to allow users to review recent changes. However, just recording all versions does not guarantee that conflicting modifications are reflected in the most recent version of a page. In this paper, we identify the requirements for efficiently dealing with conflicting modifications and present a web-based tool which allows to compare and merge different versions of a wiki page.

1 INTRODUCTION

Web-based applications become more and more important and are used in different kinds of institutions and organizations. An important category of web-based applications is represented by web-based collaborative systems. Examples are *Google Docs* (http://docs.google.com), *Yahoo Groups* (http://groups.yahoo.com), *YouTube* (http://www.youtube.com), *Google Earth* (http://earth.google.com) or *Wikipedia* (http://www.wikipedia.org).

At Google Earth, users, e.g., tag the map of the earth with points of interest or photography that is shared among all users. Through the efforts of the user community, the map becomes a reflection of what the inhabitants of the different places on the map consider as relevant. The same is true for YouTube. By allowing users to tag videos, define categories for videos, comment videos, YouTube becomes a place for exchanging thoughts rather than just consuming.

Yahoo groups support users in creating places for discussion and exchanging ideas and content. The exchange of files is one of the core ideas of Google Docs where users can interact synchronously on a shared document using just their web browser as client infrastructure.

Wikipedia is a web-based encyclopedia that is collaboratively written by many of its readers. For that purpose, Wikipedia uses a wiki that records all changes to the articles and allows to review the history of an article. Since their introduction by Ward Cunningham in March 1994 (Leuf and Cunningham, 2001), wikis achieved sustained success. Wikis are applied in many application domains and many wiki engines have been developed since then (Wiki Choicetree, 2007). Part of the Wiki success is based on their total freedom, ease of access and lack of structure (Rick et al., 2002). Wikis serve as a means for quickly expressing ideas and share information. The easy way to link pages (e.g., by simple framing with characters or by using CamelCase page names) allows the user to create a hypertext *on-the-fly*.

All of the above examples share an important characteristic: They activate users to create and share information instead of only consuming information created by professional site owners. In most of these examples, users are able to create information concurrently. This may lead to conflicting modifications of the same information. To overcome these issues, most wiki engines keep a history of all page versions and allow users to review recent changes by providing a summary. However, when several users edit the same version of a page at the same time, a conflict is reported and it is up to the user to manually include their modifications in the most recent version. Without tool support, this can be tedious and users often omit the merge process. As result not all modifica-

Lukosch S. and Leisen A. (2008).
DEALING WITH CONFLICTING MODIFICATIONS IN A WIKI.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 5-15
DOI: 10.5220/0001515600050015
Copyright © SciTePress

tions find their way in the most current version of a wiki page.

In this paper, we describe how we addressed the above issues by extending the standard versioning concept and developing a web-based tool for comparing and merging different wiki pages. For testing and evaluating our approach, we extended the CURE wiki engine (Haake et al., 2004b). However, our concepts can easily be transferred to other wiki engines. In the following, we first analyze the requirements for a extended versioning concept and a web-based tool that allows to compare and merge different pages. Then, we describe the essential concepts and features of the CURE system, before we present our approach. Finally, we compare our results with the current state of the art, report on first experiences, and conclude our paper with an outlook on future work directions.

2 REQUIREMENTS ANALYSIS

In this section, we will determine the requirements for versioning wiki pages and a tool that allows to compare and merge different versions. For that purpose, we will describe a typical use case in which students have to collaboratively write a paper about Rembrandt using a wiki engine that keeps track of different page versions.

Anja has to write a paper about Rembrandt. The teacher Mr. Miller has already created a wiki page which all students can access. He also has prepared the structure of the paper as scaffolding in a first version *V1*. Anja starts writing the text in the afternoon and saves her version *V2*. While reading the text she detects a mistake and starts editing the page. When she has finished her changes, she forgets to save the page and leaves the edit view opened.

Next day, Anja is not in school. Mr. Miller asks Beate and Carla to help Anja with the paper. Beate and Carla divide the work. While Beate checks and corrects the part about Rembrandt's life, Carla extends the part about Rembrandt's work. During writing, Beate notices that she also can add content to the other parts of the paper and extends them. When she finishes, she saves the page as version *V3*.

Carla also knows much about Rembrandt and extends first parts of the paper starting from version *V2*. When she saves her work, the wiki reports a conflict as Beate has already modified the version *V2* and created a new version *V3*. Because she has a date, she ignores the message and leaves her computer. Now, Anja as well as Carla have modified version *V2* and none of them has integrated the changes in the most recent version version *V3*. When their computer or just their web browser would crash now, their changes would get lost. Especially when schedules are tight, loosing intermediate results can make it difficult to meet deadlines. Therefore, the linear versioning concept of wiki engines has to be extended and the following requirement has to be met:

R1: Store all page versions in a version tree and make users aware of parallel versions.

When keeping track of all versions, no intermediate results will get lost and the history of a page is completely available. However, when working with text documents merging is time-consuming and difficult. There exist different approaches. Users could copy and paste the differences between different versions. Another approach could be to use an external merge tool. In our opinion, users should not have to switch their work context. Instead users should be able to resolve conflicts in the same application in which the conflicts were created. This leads to the following requirement:

R2: Offer a web-based tool and user interface for comparing and merging different versions of a wiki page.

If there are a lot of differences between two page versions, maintaining the readability of the compared versions becomes difficult. But for merging different versions, readability and thus a semantic understanding is crucial. The readability can be improved when users can decide to suppress differences which are rather unimportant for a semantic understanding, e.g. added empty lines or added white spaces. Therefore, the following requirement has to be fulfilled.

R3: Allow users to define options for the comparison.

When merging different text versions, one often notices typos or comparable small mistakes which require correction. To make such small corrections, it is necessary that the merged version can be edited on-the-fly. Otherwise, the merged version would have to be stored first and changed afterwards which would slow down the merge process. This leads to the following requirement:

R4: Allow users to edit the text of a page while merging two pages.

Continuing in our scenario, Anja continues working on the paper about Rembrandt. Since she was not in school, she does not know that Mr. Miller has involved Beate and Carla in the writing process and Anja saves her changes to version *V2*. Considering

R1, Fig. 4 displays the resulting version tree. Starting from version *V1*, Anja has created version *V2*. *V2* has been changed by Anja, Beate, and Carla. Beate has created version *V3*. When Carla stored her changes, she created version *V4*. Finally, Anja now stored her changes and the wiki created version *V6*. Now, three different versions of the same page have to be merged to keep all changes. Obviously, this is much more complicated than merging two different versions. As multiple conflicts can be serialized in conflicts of two different versions, e.g. *V3* and *V4* and the result of this merge with *V6*, the following requirement has to be met:

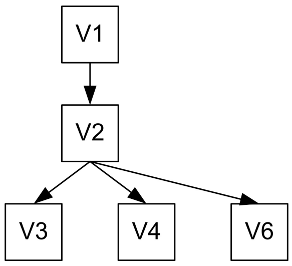


Figure 1: Version tree.

R5: Support users to solve multiple conflicts by serializing multiple conflicts.

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[1] that were used to create the system and show how they appear in the functional context of user interaction.

[1]Pattern names are set in SMALL CAPS and can be found in (Schümmer and Lukosch, 2007a) if no other reference is provided.

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 2 shows the abstractions that are offered by CURE. Users enter the web-based collaborative environment via an entry room that is called *Hall*. Rooms can contain further subrooms, content in the form of so called pages, communication channels (e.g. an EMBEDDED CHAT or a FORUM) and users.

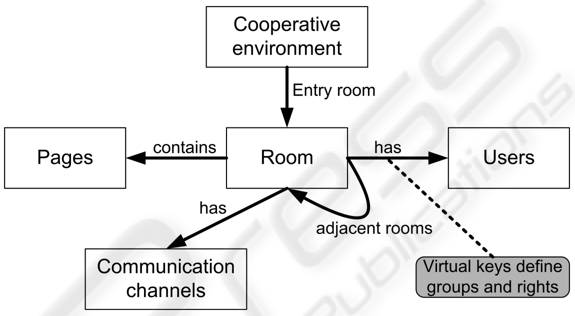
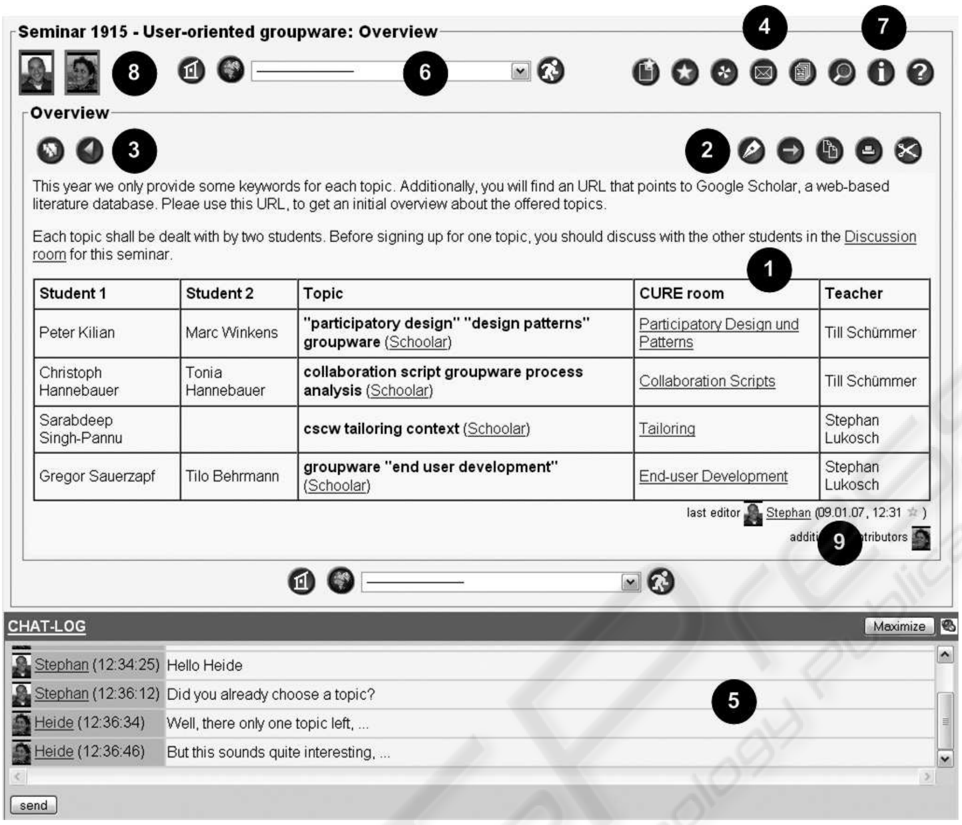


Figure 2: CURE abstractions.

When users enter a room, they can participate in collaborative activities and access the room's communication channels. They can also view the pages that are contained in the room. Users possessing suitable access rights, which are represented as virtual KEYS (Schümmer and Fernandéz, 2006), can freely edit the content of pages (Haake et al., 2004a), with the changes being visible to all members in the room after uploading. Earlier versions of a page remain accessible to allow tracing of recent changes. Pages may either be directly edited using a simple Wiki-like syntax (Leuf and Cunningham, 2001), or they may contain binary documents, 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. Thus, when users leave the room, the content stays available, allowing them to come back later and continue their work on the room's pages.

Figure 3 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-



Seminar 1915 - User-oriented groupware: Overview

Overview

This year we only provide some keywords for each topic. Additionally, you will find an URL that points to Google Scholar, a web-based literature database. Pleae use this URL, to get an initial overview about the offered topics.

Each topic shall be dealt with by two students. Before signing up for one topic, you should discuss with the other students in the Discussion room for this seminar.

Student 1	Student 2	Topic	CURE room	Teacher
Peter Kilian	Marc Winkens	**"participatory design" "design patterns" groupware** (Schoolar)	Participatory Design und Patterns	Till Schümmer
Christoph Hannebauer	Tonia Hannebauer	**collaboration script groupware process analysis** (Schoolar)	Collaboration Scripts	Till Schümmer
Sarabdeep Singh-Pannu		**cscw tailoring context** (Schoolar)	Tailoring	Stephan Lukosch
Gregor Sauerzapf	Tilo Behrmann	**groupware "end user development"** (Schoolar)	End-user Development	Stephan Lukosch

last editor Stephan (09.01.07, 12:31 ☆)

additi… …tributors

CHAT-LOG

Stephan (12:34:25) Hello Heide
Stephan (12:36:12) Did you already choose a topic?
Heide (12:36:34) Well, there only one topic left, ...
Heide (12:36:46) But this sounds quite interesting, ...

Figure 3: A room in CURE.

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 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.

4 APPROACH

The following sections will give a detailed description of our approach to address the identified requirements (R1 – R5). To show the feasibility of our approach, we extended the web-based collaborative system CURE that so far does not fulfill any of the identified requirements.

4.1 Version Tree (R1)

To keep a history of all changes, even when users changed the same page version in parallel, we extended the versioning concept in CURE by implementing the pattern IMMUTABLE VERSIONS (Schümmer and Lukosch, 2007a). In this pattern, versions are immutable as they cannot be changed afterwards. Each modification of a page is stored as new version in a version tree even when two users modified the same page version concurrently. This approach is also used to handle conflicting modifications that result from nomadic interaction with the CURE system (Lukosch et al., 2006; Lukosch, 2008).

Apart from creating a new parallel page version, CURE also creates a so-called *merge-page* which references the parallel versions. After extending the version tree, CURE asks the users to resolve the conflict on the merge-page. CURE thereby prevents users from overwriting changes other users have performed. Fig. 4 shows how the version tree of our scenario is extended with the merge-pages which make users aware of conflicting changes.

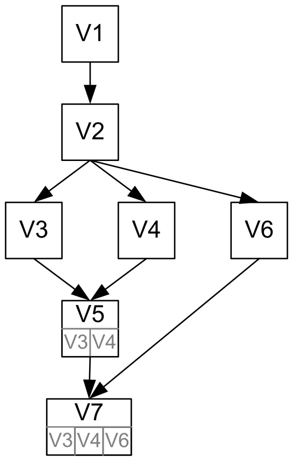


Figure 4: Version tree in CURE.

4.2 Web-based Comparison and Merge Tool (R2)

4.2.1 Selecting Different Page Versions

For comparing and merging, users first have to select two different page versions. We integrated a button, which allows users to directly compare the currently viewed version of a page with its direct predecessor. Apart from that direct access, users can also freely select two different page versions from the version tree of a page in CURE. Fig. 5 shows how CURE displays the version tree. Users can select two pages for comparison ① or merging ② by selecting the corresponding radio buttons ③.

Since the user must select exactly two versions, two groups of radio buttons were integrated in the page showing version tree (see Fig. 5). In each group, only one version can be selected. All existing versions of a page are available in each group. The selection takes place on the client side in the browser. The number of available radio buttons in each group

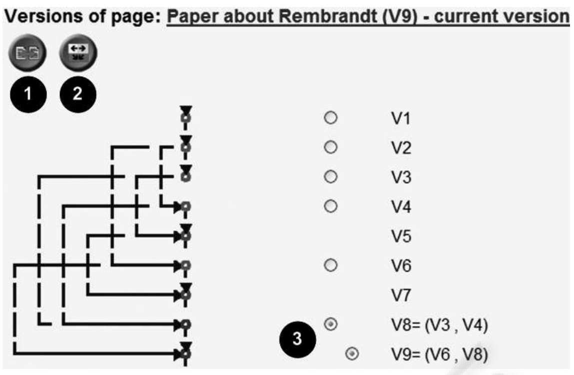

Versions of page: Paper about Rembrandt (V9) - current version
1
2
3
V1
V2
V3
V4
V5
V6
V7
V8= (V3 , V4)
V9= (V6 , V8)

Figure 5: Selecting pages for comparison or merging in CURE.

are adapted to the reasonable ones, once users make their choice. This ensures that users can only select two different versions for comparison or merging.

Additionally, users can alter their selection and can compare or merge different versions, while using the comparison and merging tool. This is possible by using drop-down menus on top the comparison or merging tool (see *Version selection* area in Fig. 6, 7, and 8).

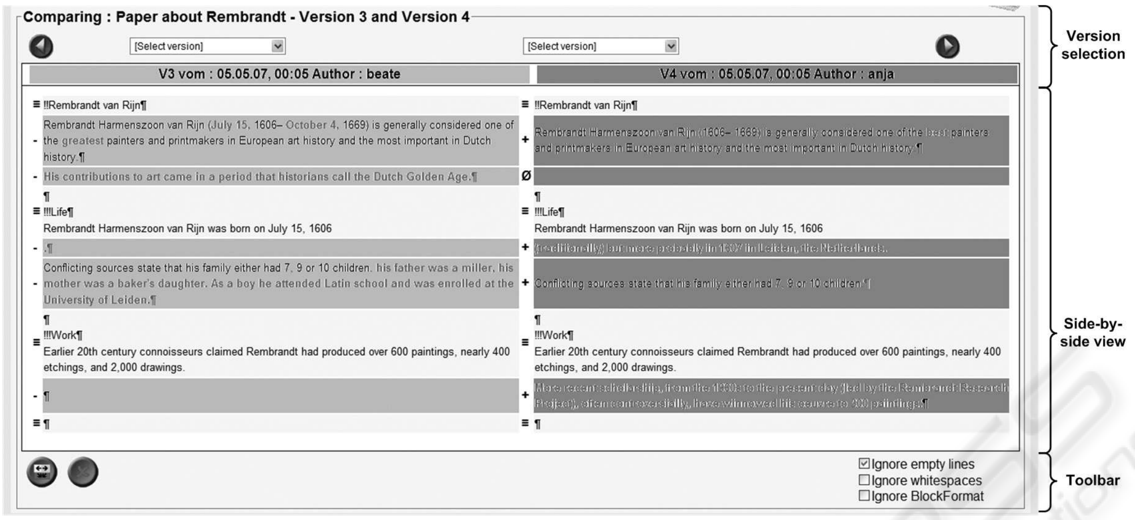
4.2.2 Comparing Different Page Versions

For the comparison of two pages, we use a standard comparison algorithm (Myers, 1986). This algorithm first identifies the different lines and then the differences in the different lines. The output of the algorithm is an *edit script*, i.e. a list of `delete` and `insert` operations, which is used to mark the differences. The differences are presented on the comparison page which is divided into the three areas *version selection*, *side-by-side view*, and *toolbar* (see Fig. 6).

The *version selection* area provides two possibilities to choose the versions for comparison. Either the user can skip to the previous/next version of the page by using buttons or by selecting a dedicated version out of a list. This list offers only reasonable versions, i.e. those which are older or newer than the displayed versions.

The *side-by-side view* area shows both versions in two columns beside each other (Yang, 1991). Similar lines are shown at the same height in order to compare them easily. The differences between two page versions are marked by using different font and color attributes. In addition to the different layout, differences are indicated by suitable symbols (+, -, ≡) in an additional column per version. This approach is also used in Wikipedia (Wikipedia, 2007).

The *toolbar* area offers two buttons for either invoking the merge page or canceling the comparison



Comparing : Paper about Rembrandt - Version 3 and Version 4

	V3 vom : 05.05.07, 00:05 Author : beate		V4 vom : 05.05.07, 00:05 Author : anja
≡	!!Rembrandt van Rijn¶	≡	!!Rembrandt van Rijn¶
-	Rembrandt Harmenszoon van Rijn (July 15, 1606– October 4, 1669) is generally considered one of the greatest painters and printmakers in European art history and the most important in Dutch history.¶	+	Rembrandt Harmenszoon van Rijn (1606– 1669) is generally considered one of the best painters and printmakers in European art history and the most important in Dutch history.¶
-	His contributions to art came in a period that historians call the Dutch Golden Age.¶	Ø	
≡	¶ !!!Life¶ Rembrandt Harmenszoon van Rijn was born on July 15, 1606	≡	¶ !!!Life¶ Rembrandt Harmenszoon van Rijn was born on July 15, 1606
-	.¶	+	(traditionally) but more probably in 1607 in Leiden, the Netherlands.
-	Conflicting sources state that his family either had 7, 9 or 10 children. his father was a miller, his mother was a baker's daughter. As a boy he attended Latin school and was enrolled at the University of Leiden.¶	+	Conflicting sources state that his family either had 7, 9 or 10 children.¶
≡	¶ !!!Work¶ Earlier 20th century connoisseurs claimed Rembrandt had produced over 600 paintings, nearly 400 etchings, and 2,000 drawings.	≡	¶ !!!Work¶ Earlier 20th century connoisseurs claimed Rembrandt had produced over 600 paintings, nearly 400 etchings, and 2,000 drawings.
-	¶	+	More recent scholarship, from the 1960s to the present day (led by the Rembrandt Research Project), often controversially, have winnowed his oeuvre to 300 paintings.¶
≡	¶	≡	¶

[✓] Ignore empty lines
☐ Ignore whitespaces
☐ Ignore BlockFormat

Figure 6: Comparing two pages in CURE.

which means to invoke the version selection page. Additionally, users can choose between three different options which influence the comparison (see Section 4.3).

4.2.3 Merging Different Page Versions

Apart from the comparison tool, we integrated two different possibilities to merge different page versions in CURE. The first possibility uses the *side-by-side view* (see Fig. 7). The side-by-side view is preferred by, e.g., programmers when comparing source code (DiffDoc, 2007). The second possibility uses only the *merge view* and adds a *result view* (see Fig. 8). Thereby, it offers a more integrated view of the different versions. Experiences have shown that the merge view is preferred for the comparison of literary texts. Since the type of presentation depends on the text type which is compared as well as on the user's personal preferences, we implemented both possibilities.

Both user interface variants support the user while merging. The vertical scrollbars of the side-by-side and merge view in Fig. 7 as well as the merge and result view in Fig. 8 are synchronized. Thereby, users can always see the corresponding parts of the compared and merged versions. In the following, we will describe the merge view, result view, and the toolbar in more detail.

The *merge view* presents both versions in a single integrated view. In contrast to the side-by-side view, identical text-parts are presented only once.

For each difference, users have to decide in the merge view which text should be transferred into the merged version. An automated decision is not possible, because the two versions are not necessarily based on the same version. The differences which have to be solved manually are called conflicts. There are two different types of conflicts:

- *Unary conflict:* Text-parts that appear only in one of the two versions, i.e. difference in one text e.g. inserted or deleted words or lines.
- *Binary conflict:* Text-parts that exist in both versions at the same position, i.e. concurrent difference at the same position in the text.

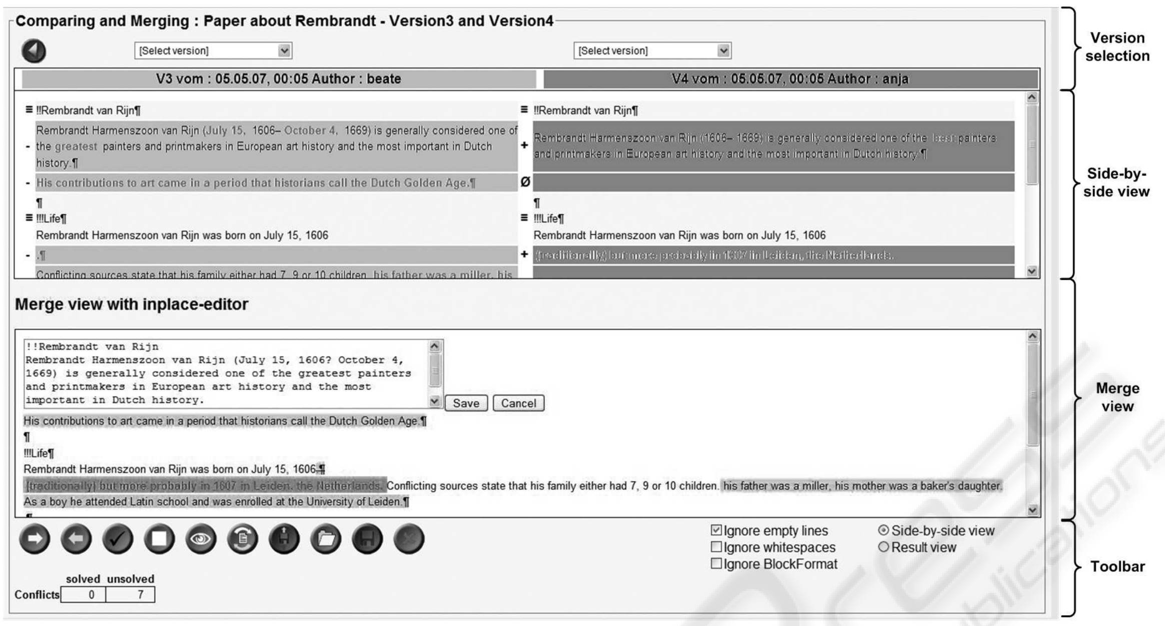
In case of a *unary conflict*, users have to decide whether this text-part should be transferred into the merged version or not. *Binary conflicts* are positioned directly one after the other. Users have to decide, which of the two possible text-parts should be transferred. It is not possible to transfer both text-parts.

Users can resolve such conflicts by directly clicking on the conflict. With each click on a conflict, the conflict changes its state. A conflict can be in three different states:

1. *Initial:* Status at the beginning of merge process.
2. *Active:* Text is part of the merged version.
3. *Inactive:* Text is not part of the merged version.

When the conflict state changes, the layout of the conflict is also changed. Table 1 shows the layout of the different conflict states. When all conflicts are solved, i.e. all conflicts have the status active or inactive, the merge process is finished and users can save the merged page as new version.

The *result view* (see Fig. 8) shows users a preview of the merged page. In the beginning of the merge process, only the identical text-parts are shown. With each solved conflict in the merge view the conflict free


Comparing and Merging : Paper about Rembrandt - Version3 and Version4

[Select version] [Select version]

V3 vom : 05.05.07, 00:05 Author : beate | V4 vom : 05.05.07, 00:05 Author : anja

≡ !!Rembrandt van Rijn¶ | ≡ !!Rembrandt van Rijn¶

- Rembrandt Harmenszoon van Rijn (July 15, 1606– October 4, 1669) is generally considered one of the greatest painters and printmakers in European art history and the most important in Dutch history.¶ | + Rembrandt Harmenszoon van Rijn (1606– 1669) is generally considered one of the best painters and printmakers in European art history and the most important in Dutch history.¶

- His contributions to art came in a period that historians call the Dutch Golden Age.¶ | Ø

¶ | ¶

≡ !!!Life¶ | ≡ !!!Life¶

Rembrandt Harmenszoon van Rijn was born on July 15, 1606 | Rembrandt Harmenszoon van Rijn was born on July 15, 1606

- .¶ | + (traditionally) but more probably in 1607 in Leiden, the Netherlands.

Conflicting sources state that his family either had 7, 9 or 10 children, his father was a miller, his

Merge view with inplace-editor

```
!!Rembrandt van Rijn
Rembrandt Harmenszoon van Rijn (July 15, 1606? October 4,
1669) is generally considered one of the greatest painters
and printmakers in European art history and the most
important in Dutch history.
```

Save Cancel

His contributions to art came in a period that historians call the Dutch Golden Age.¶
¶
!!!Life¶
Rembrandt Harmenszoon van Rijn was born on July 15, 1606.¶
(traditionally) but more probably in 1607 in Leiden, the Netherlands. Conflicting sources state that his family either had 7, 9 or 10 children. his father was a miller, his mother was a baker's daughter.
As a boy he attended Latin school and was enrolled at the University of Leiden.¶

[✓ Ignore empty lines] [✓ Side-by-side view]
Ignore whitespaces
Result view
Ignore BlockFormat

	solved	unsolved
Conflicts	0	7

Figure 7: Merging two pages with the side-by-side view in CURE.

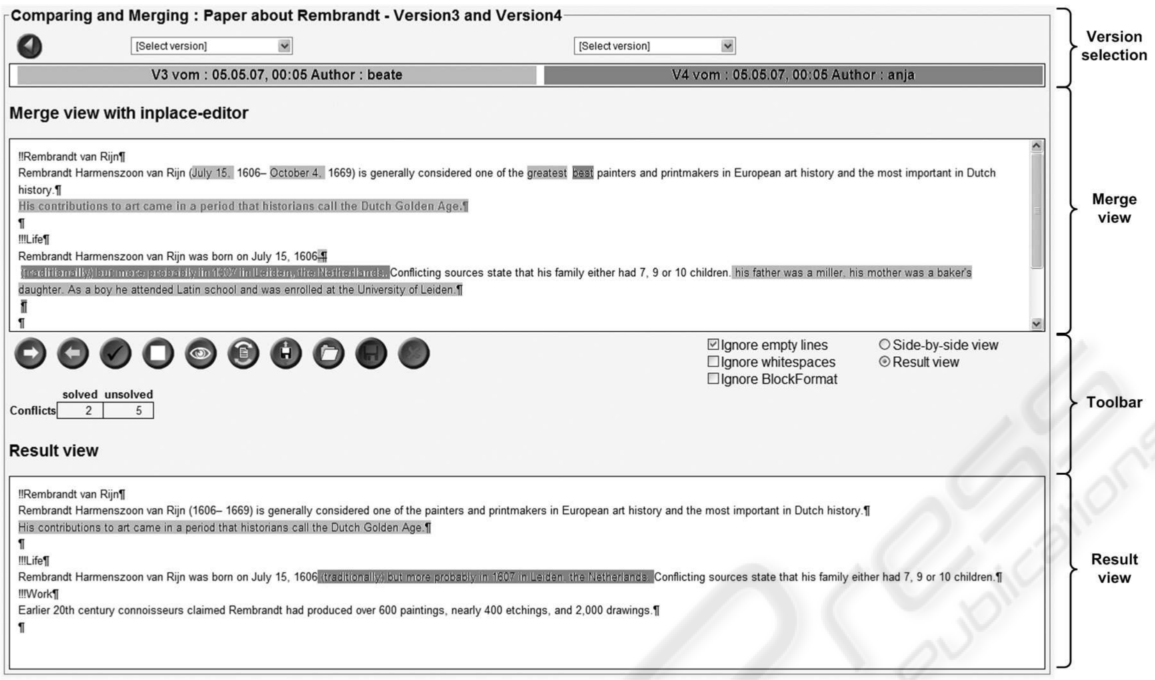
Table 1: Layout of the different conflict states.

Conflict state	Unary conflict *V1*	Unary conflict *V2*	Binary conflict
Initial	black on light yellow	black on light green	Text *V1*. Text *V2*.
Active	**red and fat** on light yellow	**red and fat** on light green	**Text *V1*.** ~~Text *V2*.~~
Inactive	~~black and crossed out on light yellow~~	~~black and crossed out on light green~~	~~Text *V1*.~~ **Text *V2*.**

text-part is added to the result view as well. The text-parts are marked with the same background-color as in the merge view such that users can recognize the originating version.

For awareness purposes, the *toolbar* area offers information about the number of solved and unsolved conflicts. Again, users can define the options for comparing to page versions (see Section 4.3). Additionally, the toolbar offers a variety of functions to support users when merging two pages (see Fig. 7 from left to right):

- *Transfer remaining differences from the right version*: Often, users want to accept most of the conflicts from one page version. This function allows to accept all not yet decided conflicts from one version.
- *Transfer remaining differences from the left version*: The same function as the previous one, just vice versa.
- *Accept all active and decided conflicts*: All *active* conflicts are accepted and displayed as identical text-parts.
- *Cancel all modifications*: All modifications are reset and the the user can start the merging process again.
- *Preview the merged page*: This function allows the user to preview the merge page.
- *Back to the merge view*: All unsolved and all crossed out text-parts are displayed again.
- *Save the intermediate merge result*: Users can save the intermediate result of their merge process. This function does not result in a new page version.
- *Load the last intermediate merge result*: Users can load the last intermediate result.
- *Save merged page*: When all conflicts are resolved, the save button is enabled and users can save the merged page as new version.
- *Return to the version selection*: By using this function, users can return to the page that shows the version tree of a page (see Fig. 5).



Comparing and Merging : Paper about Rembrandt - Version3 and Version4

[Select version] [Select version]

V3 vom : 05.05.07, 00:05 Author : beate | V4 vom : 05.05.07, 00:05 Author : anja

Merge view with inplace-editor

!!Rembrandt van Rijn¶
Rembrandt Harmenszoon van Rijn (July 15, 1606– October 4, 1669) is generally considered one of the greatest best painters and printmakers in European art history and the most important in Dutch history.¶
His contributions to art came in a period that historians call the Dutch Golden Age.¶
¶
!!!Life¶
Rembrandt Harmenszoon van Rijn was born on July 15, 1606.¶
(traditionally) but more probably in 1607 in Leiden, the Netherlands. Conflicting sources state that his family either had 7, 9 or 10 children. his father was a miller. his mother was a baker's daughter. As a boy he attended Latin school and was enrolled at the University of Leiden.¶
¶
¶

[✓ Ignore empty lines] Ignore whitespaces Ignore BlockFormat
Side-by-side view [✓ Result view]

	solved	unsolved
Conflicts	2	5

Result view

!!Rembrandt van Rijn¶
Rembrandt Harmenszoon van Rijn (1606– 1669) is generally considered one of the painters and printmakers in European art history and the most important in Dutch history.¶
His contributions to art came in a period that historians call the Dutch Golden Age.¶
¶
!!!Life¶
Rembrandt Harmenszoon van Rijn was born on July 15, 1606 (traditionally) but more probably in 1607 in Leiden, the Netherlands. Conflicting sources state that his family either had 7, 9 or 10 children.¶
!!!Work¶
Earlier 20th century connoisseurs claimed Rembrandt had produced over 600 paintings, nearly 400 etchings, and 2,000 drawings.¶
¶

Figure 8: Merging two pages with the result view in CURE.

4.3 Comparison and Merge Options (R3)

Users can choose between three different options which influence the comparison or the merge process (see, e.g., lower right corner of Fig. 6):

- Ignore empty lines.
- Ignore whitespaces.
- Ignore block-format.

Empty lines between text-sections and whitespaces are not relevant for the content of a text. With the options *ignore empty lines* and *ignore whitespaces* users can change the treatment of empty lines and whitespaces as differences. When only textual differences are important, these options help to improve the clearness of a comparison.

Block-formats are formatting instructions in the wiki syntax which mark beginning and end of a character string, e.g. in CURE `~~text~~` is rendered as *italic*. When users choose to *ignore block-formats*, only the formatting instructions are presented as differences and not the formatted text.

4.4 Editing Merged Text (R4)

When merging two pages, users often notice text-parts or words which they would like to correct immediately, e.g. typos, punctuation marks etc. For that purpose, we integrated an editor which allows users to choose the text they want to edit by simply clicking on it. Once the text is chosen, a small edit window appears and allows the users to make spontaneous corrections (see merge view in Fig. 7).

To encourage users to integrate the modifications of other users, only text parts without conflicts can be edited. Thus, users first have to resolve the conflicts before editing can take place. Once all conflicts are resolved, it is possible to edit the entire text. But this is not reasonable. In case of larger modifications, users should save the merged version first and edit the resulting new version afterwards.

4.5 Solving Multiple Conflicts (R5)

When multiple users edit a version of a page simultaneously, conflicting pages version result. By fulfilling R1, no page version gets lost. However, this also results in more than two parallel versions. In case of our example (cf. Section 2), Anja, Beate, and Carla edit the same page which results in three parallel versions. When Carla saves the modified page, CURE now detects the conflict and shows the comparing and merging tool as shown in Fig. 7. However, when Anja saves her page version, three different versions have to be merged.

Adding an additional column in the side-by-side view (Fig. 7) or integrating another text in the merge

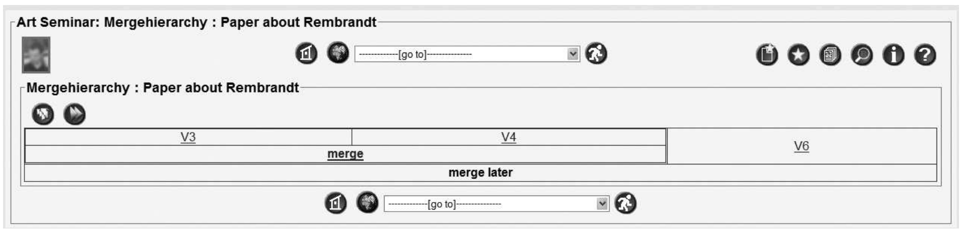

Art Seminar: Mergehierarchy : Paper about Rembrandt
[go to]
Mergehierarchy : Paper about Rembrandt
V3
V4
V6
merge
merge later
[go to]

Figure 9: Merge hierarchy in CURE.

view (Fig. 8) is not feasible, as it would be difficult for users to keep an overview. This becomes even more obvious, when considering 4, 5, 6 or even more parallel versions.

In order to simplify the merge process of three or even more pages, we split up the merge process in serially merging two versions. To support the user in serializing the multiple conflicts, CURE shows the merging hierarchy in form of a table (see Fig. 9). The user can start the merging process by clicking on the *"merge"* button and merge the two corresponding page versions. If further versions need to be merged, an updated merging hierarchy is shown again after the user has merged the selected pages. Thereby, the conflicting versions are serially merged and the modifications of all users are taken into account.

5 RELATED WORK

There are several available comparing- and merging tools that run on different platforms and are suitable for different file types like text files, binary files and HTML or XML files. A large part of these tools is based on the *GNU diffutils* (Diffutils - GNU Project, 2007) and offers a graphical user interface. Some of these tools also support editing and merging of files. In this section, we will analyze some of these tools concerning their functionality. The following list contains some of those tools in alphabetic order: *CharDiff* (CharDiff, 2007), *DiffDoc* (DiffDoc, 2007), *DiffDog* (DiffDog, 2007), *CSDiff* (CSDiff, 2007), *ExamDiff Pro* (ExamDiff Pro, 2007), *Guiffy* (Guiffy, 2007), *KDiff3* (KDiff, 2007), *Meld* (Meld, 2007), *RCS* (Tichy, 1985), *TkDiff* (TkDiff, 2007), *Unix diff* (Hunt and McIlroy, 1976), *Wikipedia* (Wikipedia, 2007), and *WinMerge* (WinMerge, 2007).

Most of these tools work line-based. They try to find common lines and lines that have been inserted or deleted by searching the longest common subsequence between the texts. From the sighted tools *Wikipedia*, *CharDiff*, *DiffDoc*, *WinMerge* and *Guiffy* are analyzed in depth, since they have exemplary character for the different types of comparing and merging tools.

Wikipedia (Wikipedia, 2007) is a wiki that allows visitors to add, remove, edit and change content, typically without the need for registration. If a wiki page is changed, the whole page is stored as new version. In case of a conflict, the new version is not stored and users have to integrate their changes in the most recent version. Otherwise, their changes are lost (violating R1). Within Wikipedia users can compare two versions of a page. The changes between two revisions of a page are displayed side-by-side.

However, Wikipedia does not support merging of two different versions (violating R2).

CharDiff 1.2a (CharDiff, 2007) is a shareware program that detects and presents differences with character precision and overlays files for an integrated display. CharDiff uses standard web browsers for the presentation of the differences. After selecting two files for comparison, the result of the comparison immediately is shown in a single integrated view in the web browser. If the mouse is positioned on the difference, the difference *waggles* in the integrated view. This effect draws the user's attention to the selected difference. A support to merge two files does not exist (violating R2)..

Diff Doc 3.50 (DiffDoc, 2007) is a commercial tool for comparing files, e.g. PDF, Word, Excel, RTF, text or HTML files. The two compared files are represented in a *side-by-side* view in the upper part of the user interface in which users can also edit the viewed files. The two compared files can be loaded by opening existing files, by copy and paste of text parts, or by creating new text. In the lower area of the user interface, the differences are either presented in a side-by-side-view or in an integrated view. For the comparison, three options are available: ignore upper and lower case, ignore white spaces, and ignore empty lines. Again, merging is not supported (violating R2).

WinMerge 2.6 (WinMerge, 2007) is an open source tool for comparing and merging of text files. The two compared texts are shown in a side-by-side view, where identical lines are ordered on the same height and differences within the lines are marked in another color. However, WinMerge does not support to resolve multiple conflicts (violating R5).

Guiffy 7.4 (Guiffy, 2007) is a Java shareware program for comparing and merging of files. Thus, it can be used with different operating systems. Guiffy supports 3-way merging with its *SureMerge* function and offers an intuitive user interface. But Guiffy is not a web-based tool and does not offer support for solving multiple conflicts (R5).

Table 2: Comparison of tools.

Requirement	Wikipedia	CharDiff	Diff Doc 3.50	WinMerge 2.6	Guiffy 7.4
R1	-	-	-	-	-
R2	-	-	-	○	○
R3	○	+	+	+	+
R4	-	-	○	+	+
R5	-	-	-	-	-

+:	Requirement is fulfilled
○	Requirement is partially fulfilled
- :	Requirement is not fulfilled

Table 2 shows the results of our analysis in accordance with our requirements (R1 – R5). To summarize, none of the above tools fulfills all of the requirements. Especially, none supports version trees or offers web-based merging support.

6 FIRST EXPERIENCES

We split up the evaluation of our support for comparing and merging different wiki page versions in three phases. In the first phase, we have conducted functional tests. These functional tests concentrated on validating the comparing and merging possibilities and checking if our requirements are met. These tests showed that all identified requirements R1 – R5 are fulfilled.

In a second phase, we 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 from different departments at our university, indicates that our tools for comparing and merging different wiki pages improve the usability of the system and are highly intuitive for dealing with conflicting modifications in wikis. Furthermore, users have reported that the tools significantly simplify to review recent changes and to resolve conflicts.

CURE is regularly used by our group to conduct seminars as well as lab courses with a blended CSCL approach (Haake et al., 2005). In the final evaluation phase, we plan to use CURE together with our tools for comparing and merging in seminars and lab courses which follow the same blended CSCL approach. Based on the gathered interaction data, we plan to evaluate how the tools affect the interaction among the users.

7 CONCLUSIONS

Nowadays, web-based applications become more and more important. Especially, collaborative web-based applications activate users to create and share information instead of only consuming it. However, when collaboratively accessing and modifying shared information, parallel modifications can take place. Most wikis record all versions of a page to allow users to review recent changes, but also to ensure that no modifications get lost.

In this paper, we identified five requirements for supporting users when working with different versions in a wiki. We showed how to fulfill these requirements by extending the linear version concept of wikis and integrating a web-based tool for comparing and merging wiki pages in CURE. The overall concepts, like the version tree, the side-by-side view, the result view, the different layouts for highlighting conflicts and differences, and the process for solving multiple conflicts, can easily be transferred to other wiki engines as well.

Up to now, there is no wiki that supports parallel versions and offers a tool for comparing and merging different wiki pages. Thus, our approach which has been integrated in CURE presents a significant step forward. In future, we want to further evaluate the effects of the comparing and merging tools on the usage of CURE. Additionally, we want investigate in how far the merging process can be even more supported by offering more user-defined options for the comparison, by considering the structure of the text, or by enabling automatic merging.

REFERENCES

CharDiff (2007). http://calendar.google.com.

CSDiff (2007). http://www.componentsoftware.com/products/CSDiff/index.htm.

DiffDoc (2007). http://www.softinterface.com/Compare-File-Programs/Compare-File-Programs.HTM.

DiffDog (2007). http://www.altova.com/products/diffdog/diff_merge_tool.html.

Diffutils - GNU Project (2007). http://www.gnu.org/software/diffutils/.

ExamDiff Pro (2007). http://www.prestosoft.com/ps.asp?page=edp_examdiffpro.

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.

Guiffy (2007). http://www.guiffy.com/.

Haake, J. M., Haake, A., Schümmer, T., Bourimi, M., and Landgraf, B. (2004a). 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, 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, 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.

Haake, J. M., Schümmer, T., Haake, A., Bourimi, M., and Landgraf, B. (2004b). Supporting flexible collaborative distance learning in the CURE platform. In *Proceedings of the Hawaii International Conference On System Sciences (HICSS-37)*. IEEE Press.

Hunt, J. W. and McIlroy, M. D. (1976). An algorithm for differential file comparison. Computing Science Technical Report 41, Bell Laboratories.

KDiff (2007).http://kdiff3.sourceforge.net/.

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

Lukosch, S. (2008). Seamless transition between connected and disconnected collaborative interaction. *Journal of Universal Computer Science (JUCS), Special Issue on 'Groupware: Issues and Applications'*.

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, 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.

Meld (2007). http://meld.sourceforge.net/.

Myers, E. W. (1986). An O(ND) difference algorithm and its variations. *Algorithmica*, 1(2):251–266.

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.

Rick, J., Guzdial, M., Carroll, K., Holloway-Attaway, L., and Walker, B. (2002). Collaborative learning at low cost: CoWeb use in english composition. In *Proceedings of CSCL 2002*, Boulder, Colorado, USA.

Schümmer, T. and Fernandéz, A. (2006). Patterns for virtual places. pages 35–74. UVK Universitätsverlag Konstanz GmbH.

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

Schümmer, T. and Lukosch, S. (2007b). READ.ME – Talking about computer-mediated communication. pages 317–342. UVK Universitätsverlag Konstanz GmbH.

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.

Tichy, W. F. (1985). RCS – a system for version control. *Software Practice and Experience*, 15(7):637–654.

TkDiff (2007). http://sourceforge.net/projects/tkdiff/.

Wiki Choicetree (2007). http://c2.com/cgi/wiki?WikiChoicetree.

Wikipedia (2007). Main page – wikipedia, the free encyclopedia. (Online; Stand 24. April 2007).

WinMerge (2007). http://winmerge.org/2.6/manual/.

Yang, W. (1991). Identifying syntactic differences between two programs. *Software-Practice and Experience*, 21(7):739–755.