A Concept Design for Interacting with Change Representations in Web-based Collaborative Writing Systems

Chien-Ting Weng

University of Tampere
Department of Computer Sciences
Interactive Technology
M.Sc. thesis
Supervisor: Roope Raisamo
May 2009

University of Tampere
Department of Computer Sciences
Interactive Technology
A Concept Design for Interacting with Change Representations in Web-based Collaborative Writing Systems
M.Sc. thesis, 47 pages, 4 index pages
May 2009

Collaborative writing is an interest area in the study of computer supported cooperative work (CSCW) and groupware raised in mid 1980s. Among variant aspects of CSCW, collaborative writing emphasizes on a group editing environment for synchronous and asynchronous collaborative document development. For tools supporting collaborative writing, studies and pioneered applications have suggested required functions: roles, communication support, permission control, track changes, change representations, version control, comment, and revision history. Among them, few efforts have been done in the representations of changes. This thesis intends to design a way to better represent changes of documents, and for subjects in collaborative writing to better interact with changes of documents. The result is represented as GUI mockups, which visualizes differences between revisions.

Keywords: Change representations, collaborative writing, groupware, cscw

Contents

1. Introduction

The term "computer supported cooperative work" (CSCW) was coined by Paul Cashman and Irene Grief in 1984; with aims to understand how people work and how technology could support them [Grudin, 1994]. This trend was due to the success of individual office applications such as spreadsheet and word processor, and development of networks. The success of individual office applications for single users proved that technology can help people in work, while networked PCs and workstations suggested a potential user base that enabled researchers and developer could further imagine tools supporting not just single users, but groups. Therefore, the CSCW research inevitably involves some form of collaboration.

Technically, the goal is to create systems that can support the work of groups and organizations in more sophisticated and interactive ways. However, with the lack of precise requirements, it requires knowledge from social psychologists, organizational theorists, educators and many other fields to gain an understanding on group activity before diving into practical design. This characteristic makes CSCW a research field crossing multiple disciplines. However, if we look into the history of CSCW research and development, the fields of computer-human interaction and information systems played the major roles.

In addition to the ambiguity of research fields, there have been different opinions of the term "computer support cooperative work". Other preferred terms include "computer supported collaboration (CSC)", "Workgroup computing" and "groupware". Nowadays, CSCW and groupware are the most widely adopted terms. Grudin relied on the term CSCW to describe the research and groupware for the technology. There are more labels other than groupware, such as: collaborative computing, workgroup computing, multiuser applications, and CSCW applications [Grudin, 1994]. In this thesis, CSCW is used to describe the research and groupware is used to describe the CSCW applications.

CSCW applications considered under the groupware umbrella vary a lot, but the key examples include the following: desktop conferencing, video conferencing, coauthoring features and applications, email and bulletin boards, meeting support systems, voice applications, workflow systems, and group calendars.

Despite of the diverseness, Grudin proposed a groupware typology, which is a variant of space and time categorization from DeSanctis and Gallupe [DeSanctis and Gallupe, 1987]. In the typology, there are three factors in each dimension, thus forming nine CSCW research domains. The table is shown as Table 1. Judging from the map, collaborative writing is identified as a kind of groupware with a different and unpredictable time and a different but predictable place.

Before the term CSCW was coined, there had been attempts at developing computer tools to assist collaborative writing in the 1970s [Newman and Newman, 1993]. Now it is a research field included within the CSCW umbrella. Studies on collaborative writing activity began in the late 1980s; the purpose was to study how

T I M E

P L A C E	**Same**	**Different but predictable**	**Different and unpredictable**
Same	Meeting facilitation	Work shifts	Team rooms
Different but predictable	Teleconferencing Videoconferencing Desktop conferencing	E-mail	Collaborative writing
Different and unpredictable	Interactive multicasting seminar	Computer boards	Work flow

Table 1. 3x3 map of groupware options [Grudin, 1994]

collaborative writing activity is conducted within a group and an organization.

Collaborative writing involves two or more people working together to produce a document [Miles *et al.*, 1993]. By "different but predictable place", it means the collaborative writing activity is carried out in several places that are known to the participants. The examples are: email exchanges, specific IRC channels, and specific web URLs. By "different and unpredictable time", it means that the activity can be carried out at different times that are unpredictable. An open-ended collaborative project like Wikipedia is an example of "different and unpredictable time" collaborative writing.

Grudin's categorization of collaborative writing activity can be further divided into synchronous and asynchronous. If the writing activity happens at the same time, which means more than two people are working on the same document at the same time, it is synchronous. For example, people get together face to face in a fixed room or place to work on one document, or using shared editor to edit the same document at the same time. On the other hand, if more than two people work on the same document at different times, then it is asynchronous writing. For example, one writes part of the content and sends the file through email to others afterwards.

Noël and Robert [2004] analyzed 12 previous studies from 1989 to 2002 on collaborative activities, giving us an overview about different research interests of collaborative writing. Researches on collaborative writing do not focus only on writing, but also activities and tools related to completing a collaborative writing project. Therefore, collaborative writing research itself can be further classified. For this part, Posner and Baecker created a taxonomy of collaborative writing based on their research on finding similarities among collaborative writing processes [Posner and Baecker, 1992]. There are four categories in the taxonomy: roles, activities, document control methods, and writing strategies.

Roles in collaborative writing systems are meant to support the definition of social roles in a collaborative writing project, because a collaborative writing group is usually composed of different people fulfilling several different social roles. Defining roles reduces the coordination problem by specifying proper access privileges to each role. Fox example, Quilt [Fish *et al.*, 1988; Leland *et al.*, 1988] provides three default roles -- co-author, commenter, reader, and user-defined roles. A co-author has full rights to a document: read, write, modify other co-author's text, and give comments. A commenter cannot modify the content directly, but can give comments. A reader can only read the document, but cannot do anything else. Other common roles in collaborative writing projects are editors, proofreaders, reviewer or visual designer. The functions of roles vary in groups: editors in a scientific paper-writing group, student report group, journalism may be given different duties.

Activities include not only writing but also other activities for participants in a collaborative writing project. Ede and Lunsford divided the collaborative writing activities into several related activities, including brainstorming, note taking, organizational planning, writing, revising, and editing [Ede and Lunsford, 1990]. The roles that the participants play and the activities that they perform in a collaborative writing project are closely related, however, one individual in a single role can perform several activities.

Writing strategies and document control methods are closely related. Different document control methods are used to support different writing strategies. Common writing strategies are the following: single writer, separate writers and joint writing strategy. In single writer strategy, there is only one member who is in charge of writing the document with help from other members. Such strategy usually comes with centralized document control method, the writer maintains the document, and other members have the privilege to read or comment on it. In separate writer strategy, a document is divided into parts and different participants are responsible of writing various parts. The document control methods used for separate writer strategy vary. Shared control method allows every co-author to have equal rights to the document at the same time, but the co-author does not modify the parts that belong to the other co-author. Or every co-author only has full rights to their own parts, but specific co-authors have full access to everyone's work to do the final integration. In joint writing strategy, several participants compose the document together, there is no clear separation on who writes which parts. Shared control method is usually applied to this writing strategy.

From taxonomies and empirical studies, we can derive the requirements for developing collaborative writing systems (Table 2). Even now, it is still hard for a collaborative writing tool to fulfill all requirements proposed by Posner and Baecker; different tools fulfill partial requirements.

While collaborative writing can be synchronous and asynchronous, Posner and Baecker found that writing usually proceeds asynchronously. Therefore, collaborative systems provide more advantages at the reviewing phase than the composing phase. Requirements derived from later studies support the same conclusion [Kim and Eklundh, 1998; Noël and Robert, 2004]. Noël and Robert summarized the basic functions from their empirical study as: track changes, version control, add comments and identify the contributor. Kim and Eklundh intended to find out the common collaborative writing practices while particularly focusing on reviewing documents. They proposed five aspects of the design of

Taxonomy	**Design Requirements**
General	1. Preserve Collaborator identities. 2. Support communication among collaborators — document annotations, synchronous interactions, and asynchronous messages.
Roles	3. Make collaborator roles explicit
Activities	4. Support the six primary writing activities: brainstorming, researching, planning, writing, editing, reviewing. 5. Support transitions between activities. 6. Provide access to relevant information. 7. Make plans explicit — process and outline plans. 8. Provide version control mechanisms — change indicators.
Document Control Methods	9. Support concurrent and sequential document access. 10. Support several document access methods: write, comment, read. 11. Support separate document segments.
Writing Strategies	12. Support one and several writers. 13. Support synchronous and asynchronous writing.

Table 2: Design Requirements Proposed by Posner and Baecker [1992]

collaborative writing tools: centralized document control, commenting function, maintenance of revision history, change representation and need for good network-centric user interfaces [Kim and Eklundh, 2000].

There are similarities in the two findings: version control can achieve centralized document control, maintenance of revision history can help to identify the contributor, and change representation is related to track changes.

Modern version control systems provide functions that can fulfill the requirements proposed by the researchers mentioned. However, because version control systems were originally developed for software development, they do not satisfy the needs of collaborative writing well. The issue “interacting with change representations” I am addressing in this thesis, is a part of version control systems related to visualizing differences between two revisions. Although modern desktop word processors improve the way for users to interact with change representations, those improvements are not applied to web-based collaborative writing systems yet.

In Chapter 2, I will introduce the role of version control in collaborative writing, the role of change representations in version control systems, and the functions of change representations in collaborative writing. In Chapter 3, I will go through the existing approaches to change representations, and analyze their pros and cons. In Chapter 4, I will propose my approach to interact with change representations in web-based collaborative writing tools. In Chapter 5, I will discuss the possible further development of this approach.   In Chapter 6, I will give a summary of this thesis.

2. Version Control Mechanisms in Collaborative Writing

Version control systems originate from tools designed for software development management [Hawley, 2003]. During the development of software, no matter the number of developers, the structures and code are modified frequently especially in the early phase. Version control tools help the management and consistency of code, which are important for the development of software projects.

The first notable program to offer version control was the Source Code Control System (SCCS) written by Marc Rochkind at AT&T Bell Labs in the 1970s. Then the Revision Control System (RCS) designed by Walter F. Tichy and developed at the Department of Computer Science at Purdue University came out in 1982. Both systems feature versioning and the ability for multiple developers on a single system to work together. In 1992, Brian Berliner and Jeff Polk developed Concurrent Version Control (CVS), which is the first notable program to offer network-capable version control. With the network capability, developers can access the CVS system via Internet, so they can work on the same project at the same time or different time from different places. Led by Karl Fogel, the CVS development team developed Subversion (SVN), the replacement of CVS in 2002. RCS, CVS and SVN are used in the software development world nowadays.

Take CVS for example, CVS features include: repository, a central place in where the documents are stored; revisions, versioning mechanism; branching and merging, diverging / rejoining development of a project; history browsing or logs, viewing history of files, what files have been changed, when, how, and by whom.

Looking at version control from the perspective of collaborative writing, the reasons for having version control in collaborative writing can be derived from Noël and Robert's empirical study: the users wanted to be able to, for example, view the changes made to the document by the different writers, make sure everyone is working on the same version of the document, add comment to the content of the document, and identify the contributors [Noël and Robert, 2004].

Those requirements can be addressed by repository, revisions and history-browsing features provided by version control systems. However, the ability to view changes made to the documents -- also known as change representations -- in version control systems has another term called diff, which is also a program used by the version control system to generate changes -- diff.

2.1 The Role of Change Representations in Collaborative Writing

In a collaborative writing project, depending on each participant's role in the project, participants can modify the content created by other participants. Therefore, the ability to follow which changes are made and why they are made to a revised document in a collaborative writing system has been pointed out in different studies about collaborative writing [Cross, 1990; Neuwirth *et al.,* 1992; Posner and Baecker, 1992; Kim and Eklundh, 2001; Noël and Robert, 2004]. Noël and Robert [2004] found that the participants tended to discuss when they intended to modify the content written by other members. In addition, the participants said their favorite function in collaborative writing tools was the one that lets them follow the changes made to a document. In Cross' study of eight writers working on an annual report, it was observed that each writer "omitted, added, highlighted or modified" the text to agree with his or her preconceptions, with unexplained changes that caused "considerable frustration" for other writers.

2.2 Scenarios of Using Change Representations

The following scenarios help to understand how change representations are used in collaborative writing processes. One of them is in a synchronous context, the second is in an asynchronous context, and the third is in a review context.

Consider the hypothetical case of three students, A, B and C, working together on a final report for a course. They all have access to computers and Internet, and everyone has the same privileges to the working document.

In a synchronous writing context, A starts writing the document while C is also working on it. When C finishes his writing, he saves the document back to repository while A is still writing. When A finishes her writing and saves the document back to repository, she is notified by the system that there has been a version saved beforehand, so she has to merge this saved version and her version before she can save her document. The system produces a change report for A to compare the differences between her version of document and C's version of document. A can see the differences between the two versions, decide whether to accept or reject changes made by C, and add comments to modifications she makes. After A completes merging her version with C's version, she can save the merged document back to repository. Next time, when A, B or C opens the document, they will all receive the merged version.

In asynchronous writing context, B opens the document to write, the revision history shows that the document was modified by C. B wants to know what changes C made to the document, she can either just view the current version written by C, or, if there are many changes, she can use the system to produce a change report that displays the differences between her last revision and C's version, and read the comments from C to know why he made such modifications.

After the draft of the document was done, the three agree that B is in charge of the reviewing work. So B reviews the document, and makes changes and comments to the document. C reads the document revised by B, he uses the change representations to follow the changes and comments made by B, and incorporates her comments to the newly revised version, then passes it to A. A does the same work as C did, so the final report is finally done.

Therefore, the functions of change representations are not just to represent changes between two revisions, but to help co-authors cope with changes, especially understand why the other person made them. Producing differences between two documents is a technical issue that has received much attention and research results, but with change representations -- how to represent the differences produced by the difference-generating tool -- there have been only a few studies on the design of change representation functions [Kim and Eklundh, 2001], especially studies from the user interface design point of view.

In the next chapter, I will go through the major designs that have been done on change representations in collaborative writing systems.

3. Designs of Change Representations

In this chapter, an overview on the major studies that have been done on change representations and the designs of interacting with change representations is presented. It examines the applications used for collaborative writing projects, focusing on how they deal with change representations, and their pros and cons.

Before diving into the evolution of the design of change representations, I would like to point out a phenomenon observed among the studies. Most studies on collaborative writing were conducted in late 1980s and early 1990s. At which time Internet and World Wide Web (shortened to the web or the WWW) were not popular among general users. So the applications developed for collaborative writing were desktop applications instead of web-based applications.

However, with the popularity of the World Wide Web, varying web services have emerged since mid-1990s, for example, web mail, web forum, discussion group, web chatting room. Web services still require a browser, which is a desktop application, but unlike other desktop applications, users can access and operate different applications via a single web browser. Before the WWW, users of personal computers installed and used different desktop applications for accessing different services on the Internet. Take following applications as examples: mail client application to receive / send emails and manage mailing list, news groups; IRC client applications to connect to IRC server; document processor or editor to write and edit documents. But with web-based services, users can access the mentioned services via a browser without installing extra applications on their computers.

Changes of tools and environment will influence the way people do the same thing, such is the challenge faced by designers and developers when porting desktop applications to web-based services. From the technological aspect, are the approaches developed for desktop applications also available on web-based programming techniques? From the user experience point of view, are the interactions designed for desktop applications still valid for web-based applications? Because there are few studies about web-based collaborative writing systems, these issues are rarely discussed, not to mention change representations.

This Chapter begins from desktop collaborative writing software, summarizing their approaches, and then proceeds to the web-based collaborative writing services.

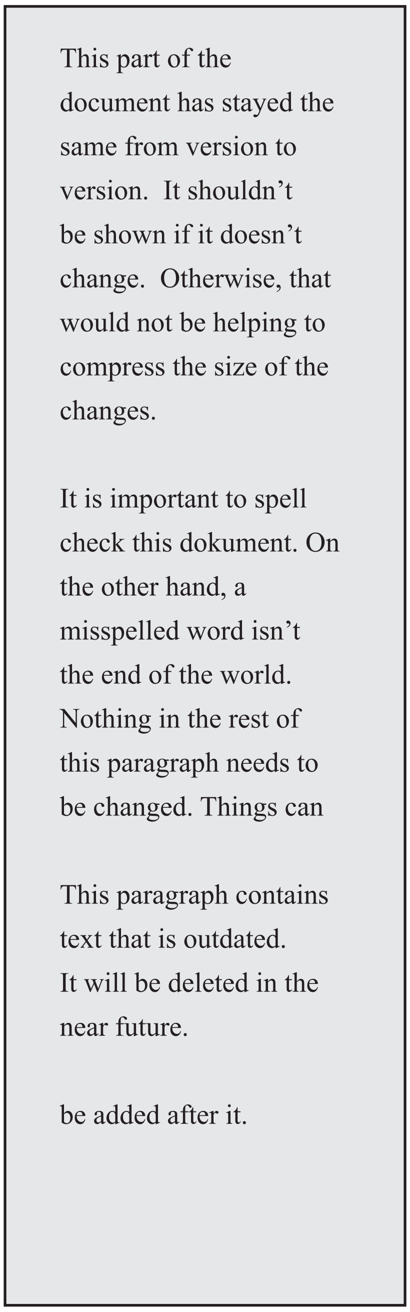
3.1 Change Representations in Desktop Applications

The need of a differential program comes from the need to distinguish the difference between two files. When the research on diff utility began, it was considered a problem in the algorithm field. The researchers focused on how to use space and time efficiently to compare the difference between two files. Gradually, this feature is integrated into word processors to support collaborative writing work.

3.1.1 Diff

As mentioned, the ability to tell differences between two files was first considered as an algorithm challenge, so it is easy to assume that representation was not a main concern at that time. When Unix Diff was officially released in 1974, it displayed the changes made per line for text files with simple visualization.

As a command line tool, the change report generated by Diff is a plain text file, which lists only the changes between two files vertically, see Figure 1 and Figure 2 for examples. When a user wants to use Diff to compare two files, the command is “diff [parameters] old_file new_file”. The default output is terminal,



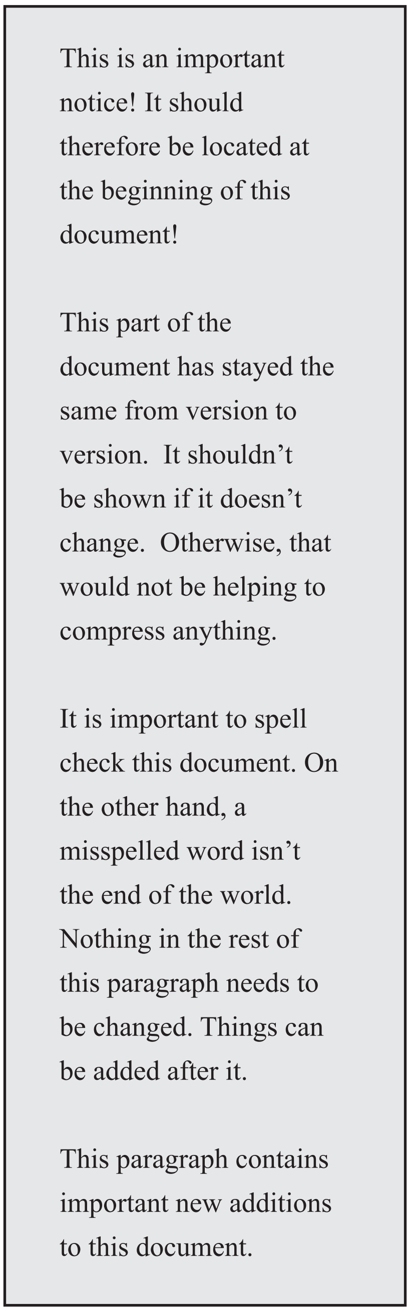
```
This part of the
document has stayed the
same from version to
version.  It shouldn’t
be shown if it doesn’t
change.  Otherwise, that
would not be helping to
compress the size of the
changes.

It is important to spell
check this dokument. On
the other hand, a
misspelled word isn’t
the end of the world.
Nothing in the rest of
this paragraph needs to
be changed. Things can

This paragraph contains
text that is outdated.
It will be deleted in the
near future.

be added after it.
```

Figure 1. Original document



```
This is an important
notice! It should
therefore be located at
the beginning of this
document!

This part of the
document has stayed the
same from version to
version.  It shouldn’t
be shown if it doesn’t
change.  Otherwise, that
would not be helping to
compress anything.

It is important to spell
check this document. On
the other hand, a
misspelled word isn’t
the end of the world.
Nothing in the rest of
this paragraph needs to
be changed. Things can
be added after it.

This paragraph contains
important new additions
to this document.
```

Figure 2. Revised document

```
0a1,6
> This is an important
> notice! It should
> therefore be located at
> the beginning of this
> document!
>
8,9c14
< compress the size of the
< changes.
---
> compress anything.
12c17
< check this dokument. On
---
> check this document. On
18c23,24
< be changed. Things can
---
> be changed. Things can
> be added after it.
21,23c27,28
< text that is outdated.
< It will be deleted in the
< near future.
---
> important new additions
> to this document.
25d29
< be added after it.
```

Figure 3. A default change report produced by Diff, divided into two columns.

and the result is as shown in Figure 3.

There are three types of changes: added, deleted and changed text. For added and deleted text, the change representation in the change report includes two parts: a line describing the change type and the position where the change is made, the added or deleted text. For changed text, the change representation in the change report includes four parts: a line describing the change type and the position where the change is made, the original texts, the separation mark "---", and the revised text.

The one-line description is at the beginning of every modified part, "a" stands for added, "d" for deleted and "c" for changed. Line numbers of the original file appear before a/d/c and those of the modified file appear after. Angle brackets appear at the beginning of lines that are added, deleted or changed, ">" means the text is added, "<" means deleted texts. Addition lines are those added to the original file to appear in the new file. Deletion lines are those deleted from the original file to be missing in the new file.

There has not been much improvement on the Diff algorithm since its release, but there have been efforts on providing more formats of the change report to make it suitable for various needs. In addition to the default option that reports all changes made to the document, Diff provides other formats for the users to indicate what changes to report in the change report. In a "diff [parameters] old_file new_file" command, the format of the change report is decided by parameters: "-e" means to display only the edited part in the report, without the original text; "-c" means context format which not only reports all changes but adds more description to the changes between two documents, which not only gives more readability for humans, but also helps Unix program Patch to apply patches to a program; “-u” means unified format which is improved from context format and is used mostly for Unix program Patch.

In the edited script format, for added and changed text, there is a line describing the change type and the position where the change is made on the old document; following the description is the edited content in the new document. But for deleted text, there is no following content after the description because the deleted part is meant to be invisible in the new document.

In the context format, any changed lines are shown alongside unchanged lines before and after. The inclusion of unchanged lines provides a context to the reader. The context consists of lines that have not changed between the two documents, so it can be used as a reference to locate the position of cchunks in the modified documents.

```
*** temp00.txt 2009-05-11
13:42:03.000000000 +0300
--- temp01.txt   2009-05-11
13:41:22.000000000 +0300
***************
*** 1,3 ****
--- 1,9 ----
+ This is an important
+ notice! It should
+ therefore be located at
+ the beginning of this
+ document!
+
  This part of the
  document has stayed the
  same from version to
***************
*** 5,25 ****
  be shown if it doesn’t
  change.  Otherwise, that
  would not be helping to
! compress the size of the
! changes.

  It is important to spell
! check this dokument. On
  the other hand, a
  misspelled word isn’t
  the end of the world.
  Nothing in the rest of
  this paragraph needs to
! be changed. Things can

  This paragraph contains
! text that is outdated.
! It will be deleted in the
! near future.


- be added after it.
--- 11,29 ----
  be shown if it doesn’t
  change.  Otherwise, that
  would not be helping to
! compress anything.

  It is important to spell
! check this document. On
  the other hand, a
  misspelled word isn’t
  the end of the world.
  Nothing in the rest of
  this paragraph needs to
! be changed. Things can
! be added after it.

  This paragraph contains
! important new additions
! to this document.
```

Figure 4. Diff change report in context format, divided into three columns.

The user can define the number of unchanged lines shown above and below a change chunk, three lines is typically the default. If the context of unchanged lines in a chunk overlaps with an adjacent chunk, Diff will avoid duplicating the unchanged lines and merge the chunks into a single chunk.

There is a two-line header at the beginning of the change report, which includes the paths to the old and new documents and their timestamps respectively. There are five parts in a cchunk: a line of asterisk marks (*) as an indication of the beginning of a cchunk; a line that tells the change information in the original document; the changed content alongside unchanged content before and after in the original document; a line that tells the change information in the modified document; the changed content alongside unchanged content before and after in the modified document.

In the line that tells the information of changes in the document, the first number is the line number indicating where the change begins in the document; the second number is the range of the change. The line that begins and ends with three asterisks refers to the original document, while the line that begins and ends with three dashes (–) refers to the modified document. In the change chunk, an exclamation mark (!) represents a change between lines that correspond in the two files, a plus sign (+) represents the addition of a line, while a blank space represents an unchanged line. The illustration is in Figure 4.

The unified format starts with the same two-line header as the context format, except that the original document is preceded by three dashes and the modified document is preceded by three plus signs. Following this are one or more cchunks that contain the line differences in the file. There are two parts in a cchunk: a line begins with two at marks (@) telling the information of the changed content.

The format of the change information line is "@@ -R +R @@". The one preceded by a minus sign (-) tells the change information in the original document, and the change information in the modified document is preceded by a plus sign. Each cchunk range, R, contains two numbers, the first number is the starting line

```
--- temp00.txt  2009-05-11
13:42:03.000000000 +0300
+++ temp01.txt  2009-05-
11 13:41:22.000000000
+0300
@@ -1,3 +1,9 @@
+This is an important
+notice! It should
+therefore be located at
+the beginning of this
+document!
+
 This part of the
 document has stayed the
 same from version to
@@ -5,21 +11,19 @@
 be shown if it doesn't
 change.  Otherwise, that
 would not be helping to
-compress the size of the
-changes.
+compress anything.

 It is important to spell
-check this dokument. On
+check this document. On
 the other hand, a
 misspelled word isn't
 the end of the world.
 Nothing in the rest of
 this paragraph needs to
-be changed. Things can
+be changed. Things can
+be added after it.

 This paragraph contains
-text that is outdated.
-It will be deleted in the
-near future.
+important new additions
+to this document.

-be added after it.
```

Figure 5. Diff change report in unified format, divided into three columns.

number of the change, and the second number is the number of lines of the change. In the change content, a space character precedes the unchanged, contextual lines, addition lines are preceded by a plus sign, and deletion lines are preceded by a minus sign. The illustration is in Figure 5.

Line-based changes means the program parses a text file by newlines, which is proper for comparing two program files, because programmers usually write one program sequence per line. In plain text documents such as essays or articles, newlines are usually used for separating paragraph. In an article level, although separation by paragraph fits the cognitive understanding of an article, with a long paragraph it adds cognitive load to readers. Although an improved front-end program Wdiff, which can compare files on a word per word basis. The output is still plain text. Plain text output does not provide good readability of the report. Readers have to know the meaning of numerous keywords and signs so to know what the change is.

In sum, the change report produced by Diff is lacking readability, which makes it not only difficult for the readers to know the reasoning behind changes, but also difficult to figure out what changes were made.

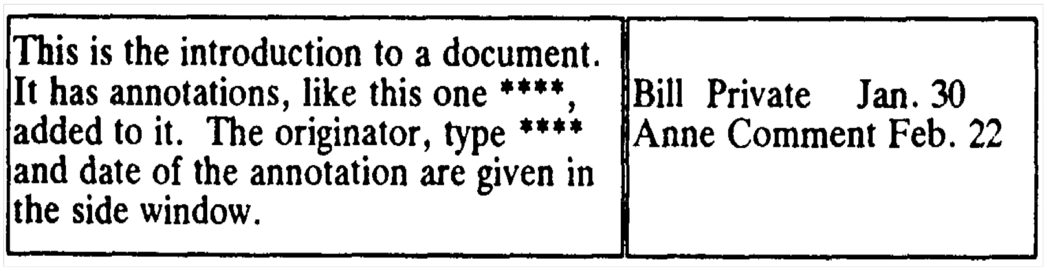
3.1.2 Quilt

Quilt was a collaborative writing tool developed in 1988 [Fish *et al.*, 1988]. Unlike collaborative writing tools developed at the same time period which concentrated mainly on document access control for multiple authors, the Quilt team thought that all types of documents and degrees of collaboration require communication among the collaborators, and that co-authors need communication to maintain a pleasant and productive working relationship. Therefore, in addition to access control, Quilt provides structured mechanisms for annotation of document, including revision suggestions, public comments, and direct or private messages.

Quilt relies on roles and collaboration styles to support collaborative writing projects. Roles are predefined and cannot be changed; they are co-author, commenter and reader. In addition to three default collaboration styles: exclusive, shared, and editor, the project creator can also customize predefined styles or define new collaboration styles from scratch. The style of collaboration determines the types of annotations permitted on documents and the social roles played by the collaborators.

A draft of the document in Quilt consists of three elements: a current base document, which is the text and other material that the writers consider can be publicly visible portion of their work; suggestions for revision in a form that users with appropriate permissions can swap with a current paragraph in the base document; voice or text comments. Although there is no comprehensive versioning system in Quilt, but for better coordination and communication, in addition to creating and reading drafts, users can save a history version of the base document, complete with its associated links. Quilt automatically records the date and time a co-author changes the document and can automatically compare the versions before and after the changes. Through an automatic process of paragraph comparison, readers can use the history version to see side-by-side comparisons of changes between versions of a draft. If there is a revision suggestion, Quilt allows examination of the difference between the two versions and swapping in of the revised version.

There are not explicit illustrations demonstrating how Quilt displays differences between two versions, the only clear part is that the users can see "side-by-side" comparisons of changes. But from examples provided by the published paper [Leland *et al.*, 1988], we can derive a rough idea. In the reading mode, when a co-author accesses a draft via Quilt with proper permissions and reads through the draft, if there are annotations, the annotation list is displayed on a side window. See Figure 6 for an example.

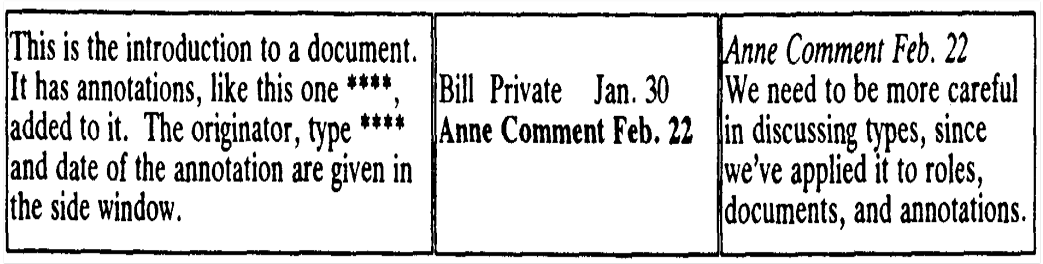


This is the introduction to a document. It has annotations, like this one ****, added to it. The originator, type **** and date of the annotation are given in the side window.

Bill Private Jan. 30
Anne Comment Feb. 22

Figure 6. Quilt in reading mode [Leland *et al.*, 1988]

If the co-author selects an annotation from the annotation list, another side window appears with the content of selected annotation. See Figure 7 for an example.

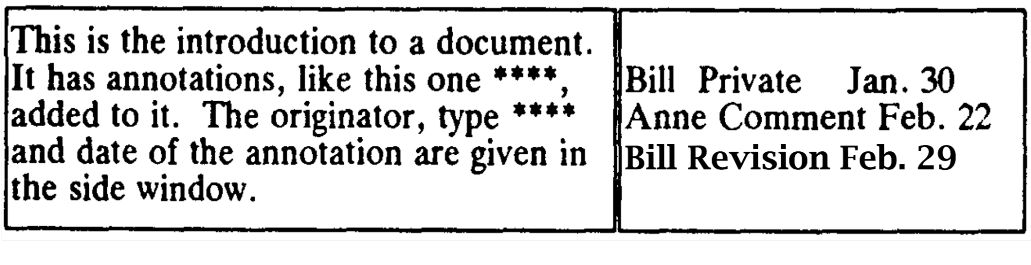


This is the introduction to a document. It has annotations, like this one ****, added to it. The originator, type **** and date of the annotation are given in the side window.

Bill Private Jan. 30
Anne Comment Feb. 22

Anne Comment Feb. 22
We need to be more careful in discussing types, since we've applied it to roles, documents, and annotations.

Figure 7. The selected annotation is displayed in another side window [Leland *et al.*, 1988]



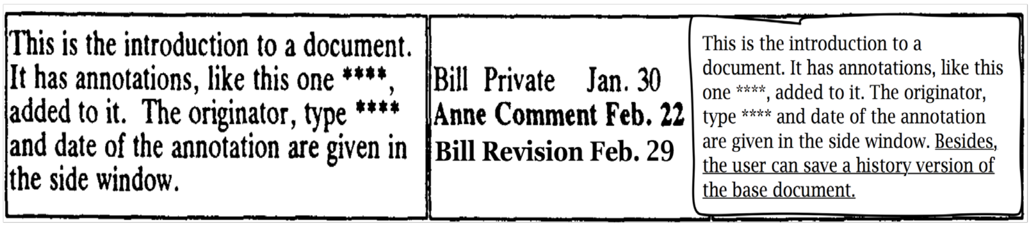
This is the introduction to a document. It has annotations, like this one ****, added to it. The originator, type **** and date of the annotation are given in the side window.

Bill Private Jan. 30
Anne Comment Feb. 22
Bill Revision Feb. 29

Figure 8. Quilt in reading mode with a revision as an annotation, based on material from [Leland *et al.*, 1988]

Following this design convention, if there are revisions in a draft, the information will be displayed at the side window as illustrated in Figure 8.

When the co-author selects a revision from the side window, another side window appears next to the annotation side window. The content of the selected revision with change representations is displayed in the side window for the co-author to read and accept / reject contents. See Figure 9 for an example.



This is the introduction to a document. It has annotations, like this one ****, added to it. The originator, type **** and date of the annotation are given in the side window.

Bill Private Jan. 30
Anne Comment Feb. 22
Bill Revision Feb. 29

This is the introduction to a document. It has annotations, like this one ****, added to it. The originator, type **** and date of the annotation are given in the side window. <u>Besides, the user can save a history version of the base document.</u>

Figure 9. Possible side-by-side comparisons of changes in Quilt, based on material from [Leland *et al.*, 1988]

3.1.3 PREP

PREP was developed in 1994. It was basically for asynchronous collaborative writing [Neuwirth *et al.*, 1994]. Inheriting from the idea of Quilt, PREP as well emphasized social communication issues in collaborative writing, especially co-authoring and commenting. There are three issues that the PREP team intended to address: support for social interactions among co-authors and commenters, support for cognitive aspect of co-authoring and external commenting, and support for practicality in both types of interaction [Neuwirth *et al.*, 1990].

PREP agrees with the function of roles and flexible collaboration styles in Quilt, but the team also observed the insufficiency of Quilt. Roles such as "co-author" and "commenter" substantially underspecify the activities involved in coordinating complex tasks such as collaborative writing. Writers also need support for coordination activities that fall outside role boundaries. An acute example is support for the communication about comments. Comments are meant to help co-authors understand the comments from commenter, however, comments are not always easy to understand, and moreover, the lack of consistency in comments and contradictory comments can be frustrating to authors.

At the co-authoring part, it has been observed that "edit-review-incorporate" cycle is one of the most common events in a co-authoring relationship. The cycle is described in the scenario in Chapter 2. Because unexpected and unexplained changes to texts can cause frustration for co-authors, to solve this, communication about changes to texts should be supported by collaborative writing systems.

PREP focused on the design of interfaces, specifically on the visual representation of the draft, and interaction with the draft to achieve its goal to support communication on co-authoring and commenting. Based on the purpose, a critical concept developed by PREP is versioning, which allows revisions to exist as distinct versions of the draft. Though versioning and history log are not new concepts in developing collaborative writing systems, PREP is the first one that not only implements the versioning mechanism, but also devotes effort in designing and developing interfaces for representing differences between revisions for collaborative writing systems [Neuwirth *et al.*, 1992; Kim and Eklundh, 2001]. The design features of PREP such as comment history and the pinpointing of change representations are still used by nowadays word processors.

A flexible text differencing system "flexible diff", allowing collaborative authors to customize change reports to their various social and cognitive needs, is embedded in the PREP editor. Flexible diff intends to answer three questions about change reports: what changes should be reported, how should changes that are reported be pinpointed, and what should the user interface to the change report be like.

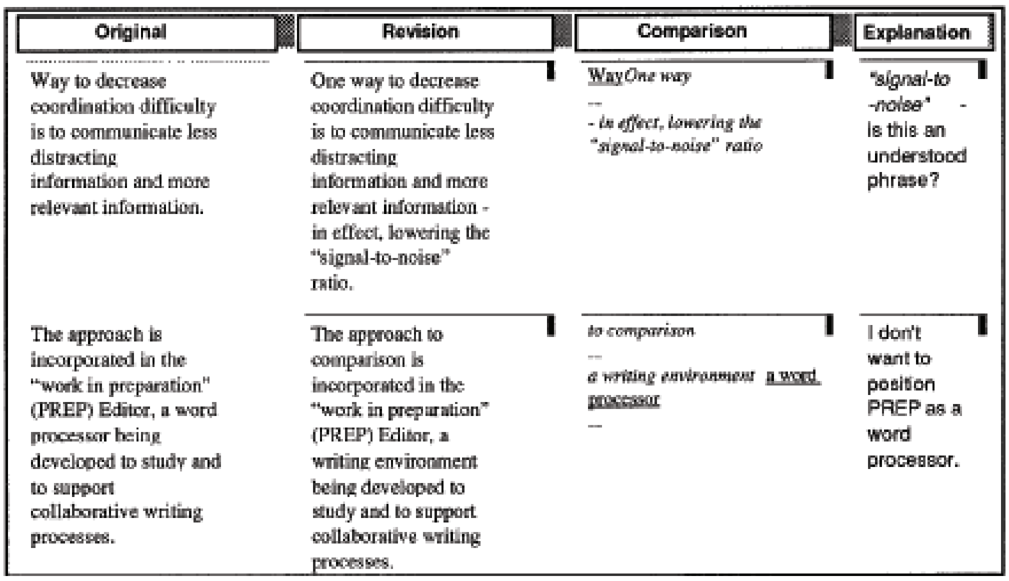
Regarding "which changes should be reported", instead of "reporting all changes", Nachbar argued that for some tasks, reporting all changes is inappropriate [Nachbar, 1988]. Neuwirth *et al.* argued that there are factors that influence how co-authors think of what changes should be reported. The trust level the writer has toward co-authors and reviewers is one of the factors. If a more trusted member reviews the draft, the writer may not want to review all changes. Another is the development phase of the document. If a document is at early-drafting phase, the changes may be dramatical every time it is revised. Some writers may prefer reporting all changes at this phase because they want to see what happened to especially their written parts, but some writers may have opposite preference, because reporting huge amount of dramatical changes with improper change representation can be distracting and can reduce the readability of the document. At some point, the writers may want to see only the added parts, the deleted parts or the moved sentences or paragraphs, depending on if they find the reports useful or not. Since trust level, distraction level and usefulness level are hard to evaluate objectively, a differencing program for collaborative writing should be flexible, to allow writers and readers to specify what changes to ignore.

For "how should changes that are reported be pinpointed", it is considered whether the changes should be pinpointed at its exact position, or pinpointed according to the number, density and complexity of changes. Again, for the readability and distraction level when a reader reads a revised document, the flexibility to represent changes is required for collaborative writing systems.

To offer flexibility on change reports to users of the collaborative writing system, PREP applies heuristics and parameters to its differencing program. The co-author can set a "change threshold", so that differences between two units are ignored if some percentage of their parts are equal. Setting the percentage to 100% will report all changes. Other parameters are for determining how changes in a text are pinpointed. The "coarseness" defines at which level the changes are to be pinpointed: character, word, phrase, sentence, or paragraph. Three parameters are used to define how precisely replacements are pinpointed: maximum distance to look for commonalities, maximum percent of differences, and maximum distance to concatenate.

The PREP team implemented an interface for the flexible diff, which is embedded in the PREP editor. The interface supports side-by-side columns of text, with horizontal alignment that enables “at a glance” viewing of large numbers of annotations and related texts. The “side-by-side” design is the same as in Quilt, but the horizontal annotation history is pioneering. As shown in Figure 10, there are four columns when a change report is produced, starting from the left: the first column is the original text, the second is the revision, the third is the comparison, a.k.a the change report, the fourth is the explanation to the changes made to the original text. PREP reports changes sentence by sentence, so every sentence is a row with four columns.

For readers accustomed to horizontal reading and writing, displaying changes in a side column fits to their cognitive process when dealing with reading tasks. Compared to traditional Unix diff that displays changes by line [Hunt *et al.*, 1975], cchunking and displaying changes by sentence is more logical for an article, and it helps readers to understand the meanings and context in an article.



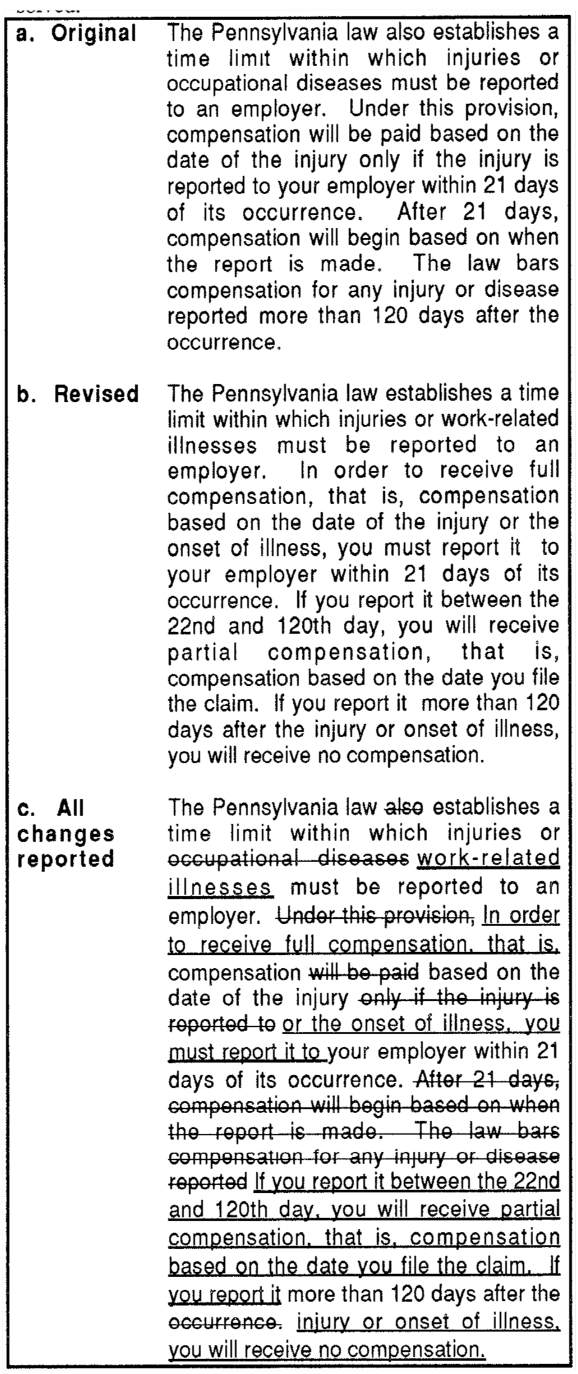
Original	Revision	Comparison	Explanation
Way to decrease coordination difficulty is to communicate less distracting information and more relevant information.	One way to decrease coordination difficulty is to communicate less distracting information and more relevant information - in effect, lowering the “signal-to-noise” ratio.	Way *One way* — *- in effect, lowering the “signal-to-noise” ratio*	"signal-to -noise" - is this an understood phrase?
The approach is incorporated in the “work in preparation” (PREP) Editor, a word processor being developed to study and to support collaborative writing processes.	The approach to comparison is incorporated in the “work in preparation” (PREP) Editor, a writing environment being developed to study and to support collaborative writing processes.	*to comparison* — *a writing environment* a word processor —	I don't want to position PREP as a word processor.

Figure 10. Side by side change report in PREP [Neuwith *et al.*, 1992]

Ideally, a fine-tuned combination of change threshold and parameters that are appropriate to reader’s cognitive and social needs can help readers understand a revised document more effectively and efficiently. Therefore, the users should have the motivation to adjust various parameters according to their needs. However, as Noël and Robert revealed in their empirical study on collaborative writing, too many difficult functions offered by collaborative writing systems is ironically a cause that stops people from using them [Noël and Robert, 2004], so is the PREP users’ attitude toward complicated parameters. To compensate for this shortcoming, PREP provides default parameters for its flexible differencing program based on predefined heuristics. Neuwirth *et al.* [1992] also recognize that most users do not change the defaults.

From the perspective of cognitive load, figuring out how to adjust various parameters and change thresholds maybe more distractive to users, and require more cognitive effort than understanding the context of changes; because it takes time and trials for co-authors to find out the best configurations for their needs.

For the visual cue to represent changed text in the change report of PREP, PREP uses italic text for inserted texts and underlined text for deleted texts, as shown in Figure 10. This is a bit odd format because in English writing, italic text has its own function. The convention used in the models of reading process is that strike-through corresponds to deleted texts and underline corresponds to inserted texts, as shown in Figure 11.



a. Original	The Pennsylvania law also establishes a time limit within which injuries or occupational diseases must be reported to an employer. Under this provision, compensation will be paid based on the date of the injury only if the injury is reported to your employer within 21 days of its occurrence. After 21 days, compensation will begin based on when the report is made. The law bars compensation for any injury or disease reported more than 120 days after the occurrence.
b. Revised	The Pennsylvania law establishes a time limit within which injuries or work-related illnesses must be reported to an employer. In order to receive full compensation, that is, compensation based on the date of the injury or the onset of illness, you must report it to your employer within 21 days of its occurrence. If you report it between the 22nd and 120th day, you will receive partial compensation, that is, compensation based on the date you file the claim. If you report it more than 120 days after the injury or onset of illness, you will receive no compensation.
c. All changes reported	The Pennsylvania law ~~also~~ establishes a time limit within which injuries or ~~occupational diseases~~ work-related illnesses must be reported to an employer. ~~Under this provision,~~ In order to receive full compensation, that is, compensation ~~will be paid~~ based on the date of the injury ~~only if the injury is reported to~~ or the onset of illness, you must report it to your employer within 21 days of its occurrence. ~~After 21 days, compensation will begin based on when the report is made. The law bars compensation for any injury or disease reported~~ If you report it between the 22nd and 120th day, you will receive partial compensation, that is, compensation based on the date you file the claim. If you report it more than 120 days after the ~~occurrence.~~ injury or onset of illness, you will receive no compensation.

Figure 11. An example revision and change report with all changes reported [Neuwirth *et al.*, 1992; Samuels and Kamil, 1984]

Instead of developing a new system for collaborative writing from scratch, Malcolm and Gaines hypothesized that the main potential users of collaborative writing systems would be current users of standard commercial word processors. Therefore, the other approach is to develop functions that support collaborative writing on existing word processors [Malcolm and Gaines, 1991].

Based on the hypothesis, the advantage of merging collaborative writing support into standard word processors is obvious. The potential users of collaborative writing systems are already used to the writing environment and functions available in the word processors they are using currently, therefore, they would not be ready to accept any degradation in facilities in using an experimental system and neither would they be willing to make major changes in their work practice in a short term. In addition, the rich formatting functions in the word processor can help with the representations of changes in the change report.

The specifications of requirements for supporting collaborative writing in word processors focus on version control, document control methods that support synchronous writing, and communication that supports comments, annotations and their logs. Commercial products such as Adobe FrameMaker and Microsoft Word do adopt this approach by adding collaborative writing functions to their products.

A study on reviewing practice in collaborative writing supports this hypothesis and approach as well. In Kim and Eklundh's study toward fifteen collaborative writing groups, seven groups used Microsoft Word as their writing tool, five groups used Latex and three used Adobe FrameMaker [Kim and Eklundh, 2001].

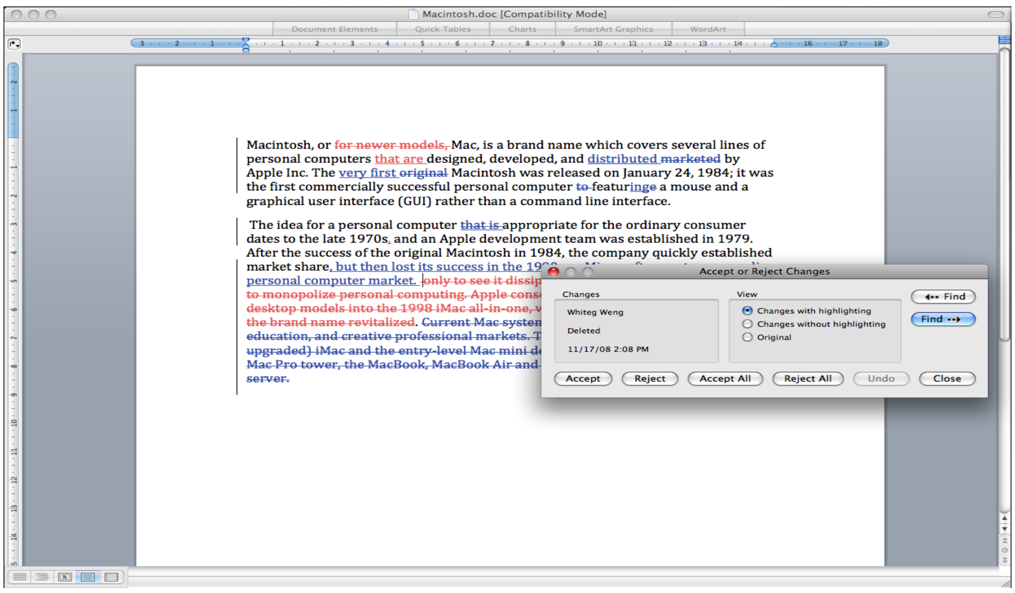
3.1.4 Microsoft Word

In Microsoft® Word 2008 for Mac, the interaction functions with changes and change representation are called "Track Changes", which can be found under Tools on the menu bar. There are three change representation functions. The first one is the function called "Highlight Changes" by which users can start or stop recording changes of text, and make the changes shown on the screen or hidden while editing.

Microsoft Word does not have versioning. When the user activates "Track changes while editing" in "Highlight Changes", all changes are logged on one single document without revision number; in other words, there is not a related version saved for every revision of the document. If "Track changes while editing" is off, then, no changes will be recorded in the document. If the user activates "Highlight changes on screen" as well in "Highlight Changes", both recorded changes and modifications that are being edited are displayed on the document, there is indication at the border of each line that is changed. Hovering on the changed text will bring up a small pop-up box with information: changed by who if the User Information is available from Word configuration, when the change is made, and change type (deleted or inserted).

The second function in "Track Changes" is "Accept or Reject Changes", which enables the user to accept or reject a change that has been made. To accept or reject a change, hover on the changed text, choose "Accept or Reject Changes" from "Track Changes" on menu bar, which brings out a dialogue box as shown in Figure 12. The user can then choose whether to accept or reject a change from the dialogue box, and find previous or next changes on the document. For an accepted change, the format of changed text will become the same as the general text, no longer highlighted as changed text; for a rejected change, the changed text will disappear from the document, the result is similar to undoing a change.

The third option in "Track Changes" is "Compare Documents", which allows the user to compare two documents. The result is displayed as a new Word document with all contents, where differences are highlighted. "Accept or Reject changes" is also available on the document generated by "Compare Documents".



Macintosh, or ~~for newer models,~~ Mac, is a brand name which covers several lines of personal computers that are designed, developed, and distributed ~~marketed~~ by Apple Inc. The very first ~~original~~ Macintosh was released on January 24, 1984; it was the first commercially successful personal computer ~~to~~ featur~~e~~inge a mouse and a graphical user interface (GUI) rather than a command line interface.

The idea for a personal computer ~~that is~~ appropriate for the ordinary consumer dates to the late 1970s, and an Apple development team was established in 1979. After the success of the original Macintosh in 1984, the company quickly established market share, but then lost its success in the 19
personal computer market. ~~only to see it dissip~~
~~to monopolize personal computing. Apple cons~~
~~desktop models into the 1998 iMac all-in-one, v~~
~~the brand name revitalized~~. ~~Current Mac systen~~
~~education, and creative professional markets. T~~
~~upgraded) iMac and the entry-level Mac mini de~~
~~Mac Pro tower, the MacBook, MacBook Air and~~
~~server.~~

Accept or Reject Changes

Changes: Whiteg Weng, Deleted, 11/17/08 2:08 PM

View: [✓ Changes with highlighting] Changes without highlighting Original

Figure 12. Options of "Accept or Reject Changes" in Word

Without saving every revision as an individual document, the user is not able to view a revision made by a specific co-author. In addition, comments are separate from changes, so the user is not able to read a change and add a comment to the change at the same time. One can argue that in reviewing a revised document it is more important what changes was made, not who made changes. In that way, logging all changes in the same document makes it convenient for the co-author to see what changes are made. It is still possible to get the co-author information by hovering on the changes.

For the representation of changed text, Word uses color to indicate changes made by different co-authors. There are two types of changes: inserted and deleted. For inserted texts, the default style of text is underlined with color; for deleted text, the default style of the text is strike-through with color. Both styles can be configured in Word preferences. Especially for deleted texts, there is a style called "hidden", which allows the deleted texts to be hidden from the document.

Although it is easy to position the changed text in a Word document, however, the clutter of texts with mixed colors and strikethroughs like in Figure 12 may cause difficulties in reading and revising a revised document. Because in order to revise a document, one has to sense the flow of the text to feel how the parts to be revised harmonize with the unchanged text, but scattered texts make it difficult to extract meaning from original context -- especially for a document which has been reviewed back and forth for a few times. A subject said that he alternatively switched on and off the mode of "Highlight Changes" on the screen about ten times so that he could avoid the problem of cluttering text [Kim and Eklundh, 2002].

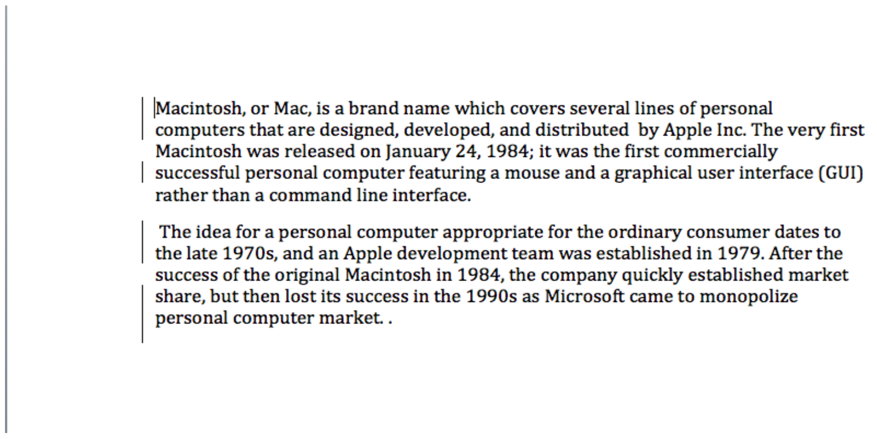
3.1.5 FrameMaker

As a desktop publishing program and word processor for professional publishing, Adobe FrameMaker is equiped with various features. In Adobe FrameMaker, there are three functions related to changes on document revisions: "change bars" under "Format → Document" or "Format → Style" menu, "compare documents" under "File → Utility" menu, and "track text edit" under "Special" menu.

A change bar is a vertical line (usually in the margin of a column) that visually identifies new or revised text on the document. The user can choose whether to automatically indicate all changes made on the document with change bars or manually add change bars to specific changes (texts or paragraphs), so the user can flag only the most important changes to the document rather than flag every change. This can be considered as a corresponding implementation to the parameters of "what to report in a change report" in PREP, but with simpler interactions.

Unlike Word, change bars do not display changes word by word, but only indicate which part of text has been modified. The function corresponding to "Highlight changes" of Word is "Track Text Edit". When "Track Text Edit" is activated, the added and deleted text is highlighted for visual distinction. The user can navigate through the edited sections and accept or reject specific edits. The user can also preview the document to see its original or final state. By default, this function can only be activated by editor and reviewer.

Like in Microsoft Word, the user can compare two documents with "Compare documents" function to receive a detailed change report. When running "Compare


Macintosh, or Mac, is a brand name which covers several lines of personal computers that are designed, developed, and distributed by Apple Inc. The very first Macintosh was released on January 24, 1984; it was the first commercially successful personal computer featuring a mouse and a graphical user interface (GUI) rather than a command line interface.

The idea for a personal computer appropriate for the ordinary consumer dates to the late 1970s, and an Apple development team was established in 1979. After the success of the original Macintosh in 1984, the company quickly established market share, but then lost its success in the 1990s as Microsoft came to monopolize personal computer market. .

Figure13. Change indications with Change bars in FrameMaker

Documents" function, FrameMaker generates two documents as results: composite document and summary document.

The composite document is a conditional document that combines the newer and older versions; it shows the differences side by side. The co-author can specify the condition tag to apply to changed text, and whether changes should be flagged with change bars. A conditional document in FrameMaker is a document containing conditional texts that are output selectively by the author.

The summary document contains a general summary and a revision list for each type of item being compared. The co-author can then create the summary as a hypertext document, with links to the actual pages where the changes occurred. By creating a hypertext summary document, the co-author can quickly display changed pages for reading or editing.

For the visual representation of changes, when working with "Change Bars", the co-author can decide the style of change bars, including: thickness, color, and the distance from the column of text to the change bar.

When working with "Track Text Edit", if the co-author starts typing text in a document where the "Track Text Edit" feature is switched on, the string "(FM8_TRACK_CHANGES_ADDED)" or "(FM8_TRACK_CHANGES_DELETED)" appears on the left side of the status bar of the document window. Text additions appear in a green color, and deletions appear in a red color with a strikethrough.

When comparing documents, there are three parameters for the user to choose on how and what changes should be reported: Mark Insertions With, Mark Deletion With, and Mark Changes with Change Bars. For example, if the co-author wants to see only inserted texts, it can be achieved by specifying how to display inserted text in the Mark Insertions With area, then specifying "Replacement Text" in the Mark Deletion With area and leaving the text box empty. If not specified, the inserted texts are marked by default condition tag (Inserted) while the deleted texts are marked by default deletion tag (Deleted).

FrameMaker is ambitious to support variant scenarios in collaborative writing: change bars to indicate changes but not disturb the context of writing, track text edit to display all changes for revising and reviewing process, separated change list for an overview of changes. However, different functions have to be activated from different menus and with different procedures. The lack of grouping related functions constructs a barrier for the users to effectively use them.

3.1.6 Summary of Desktop Applications

To sum up from the applications discussed, there are three aspects related to the interaction with change representations: which changes to report, how to represent changes, and how to interact with changes.

For what to report, it was argued that under certain circumstances, not all changes should be reported. But still, the instinctive answer to the question is that all changes should be reported. Therefore, the control of what to report is not common on collaborative writing systems. PREP supports it with a heuristic -- differences between two units are ignored if some percentage of their parts are equal and lets the user decide the percentage. FrameMaker lets the user to decide the place where the changes should be reported. The former may work well with a good setting, but it is not straightforward for the user, and the result is not easy to predict. The way FrameMaker provides for the user to choose what changes to report is easier. Because for every change, the user can manually decide whether to report it or not, the user does not have to predict the result. But it may be too much work: if the document is long, there may be many changes to be flagged.

Representing changes includes two parts: highlighting and layout. How to highlight changes has been a debating issue. There are pros and cons for both representation by indication (such as change bars in FrameMaker) and representation by display (such as highlight changes in Word). The indication mode is suitable for reading the whole document but not so helpful in understanding and checking changes in detail. On the other hand, display mode works well in understanding and checking changes, but its effectiveness decreases as the amount of changes increases. For a document, especially a draft in the early stage, with over three revisions, the clutter of text makes it not only difficult to read, but also difficult to track the relation among changes. Kim and Eklundh concluded from their interview that the users actually have different purpose for the two representation modes [Kim and Eklundh, 2002].

When the co-author wants to read a whole text or paragraphs, indication mode is favored, because it gives fewer disturbances in reading and understanding the context of a document. For example, indication mode can be suitable when revising a rough draft that requires many modifications, because understanding every change in detail may not be so important at this stage, but understanding the flow and structure of text is more valued. On the other hand, if content in the document is almost set, display mode is useful for proofreading, reviewing or editing. Because the density of changes is low, it is clear to see the changes and track how the changes are made.

PREP favors display mode, because it helps understand the changes. It seems that both FrameMaker and Word support two mode at the same time, therefore the user can choose the mode they need according to their needs. In Word, the user can either choose Balloon, or change the representation settings to achieve display mode. But Balloon mode still displays all changes, just some text are moved to the balloon at the side of the document, which is even more disturbing than usual. Both FrameMaker and Word allow the user to choose the visual cues of the changes from a set of predefined parameters such as color, bold, italic, underline, strike-through, none, or hidden...etc.

For "how to represent changes?" early Diff omitted the unmodified text in the report, and vertically displays only the original text and changed text. Although it provides the line number so the user can identify the changed position in the original document, still it is difficult to understand the context of changes without the ability to see full documents conveniently.

Sdiff, Quilt and PREP provide side-by-side columns. One column displays the original text, another displays the changed text with changes highlighted, and more columns displaying other information such as annotation and comments. The argument here is that the readers accustomed to horizontal writing read faster in the horizontal direction than in the vertical. The problem with side-by-side columns is, that when revising a document, extra columns reduce the space of writing on the current document. In that way, it is not so preferred by word processors, because full screen is considered less distracting for writing. Nowadays, side-by-side columns are visible in version control systems, and web-based interfaces, but not common in word processors with collaborative writing functions.

As for interaction with changes, three interactions are considered: browse a series of change histories, accept a change, and reject a change. Change histories means being able to browse the evolution of a change on every revision of a document. It is obvious that now the information that comes with a change is not the change itself, but with the time, the name of the co-author / reviewers who made the change, and comment to the made change. In a collaborative writing process, change histories with assistant information helps both co-authors and reviewers to understand the context how changes are made, and form conventions on how to develop the document. To achieve this purpose, a version control system that stores every revision identically is required. PREP is equipped with such design but it is not further developed in other collaborative writing systems.

Almost all systems recognize the importance of the ability to accept and reject a change that is made in the previous revisions, FrameMaker and Word use a dialogue box to find, accept and reject changes, Quilt and PREP use menu options to swap changes. However, with normal version control systems such as CVS or Subversion, the user has to manually merge documents.

3.2 Web-based Collaborative Writing Tools

Tim Berners-Lee created the idea of World Wide Web, which refers to a system of interlinked hypertext documents accessed via the Internet [Berners-Lee and Fischetti, 1997], in 1989. The tool to access the data on the WWW is a web browser, which allows the user view web pages that contain text, images, videos, and other multimedia and navigates among them using hyperlinks. The release of the first web browser in 1992 opened the door of the prosperity of web. Ever since then, a variety of services has emerged around web, including collaborative writing tools.

3.2.1 Wikis

The barrier of conducting collaborative writing on the web is how to write web pages. For a long time, it was required to understand HTML in order to write web pages. The user had to firstly write HTML code with a text editor, save the document, and then upload the document to a web server. So the content could be seen from a web browser. The idea of "Wiki Wiki Web" (now simplified to "Wiki") -- writable web, was created by Ward Cunningham in 1995 [Leuf and Cunningham, 2001]. A wiki is a web-based software that allows all viewers to change the content by editing the page and to add new pages online in a browser. Only a common web browser is required in order to do so, and the user does not have to download any special plug-ins or software, as the text is edited in a common HTML <textarea>. The user does not have to know any HTML syntax to write a wiki page, plain text or simple wiki markup language can do the trick.

The simplicity and convenience of wiki turn the web into a desirable tool for collaborative work on text and hypertext. The trend can be seen from an analysis of the usage of CoWeb, an early wiki system. CoWeb identifies four categories of how it is used: collaborative artifact creation, review activities, case library creation, and distributing information [Dieberger and Guzdial, 2002].

There has been a lot of enhancements and improvements on wiki systems since the first Wiki web was created in 1995. Similarly, the requirements of a wiki system have evolved from simple "writable web" to a platform for numerous collaborative missions on web upon the goals of the users. So far, there have been 110 wiki software registered in wikimatrix.org. Despite the varieties, there are common characteristics among wikis summarized by researchers: editing, history, links, recent changes, sandbox, and search functions [Ebersbach *et al.*, 2008].

Editing. In general, there is an edit button on every wiki page. The edit button on a wiki page indicates that "this is page is editable", so the users know they can write on it. With permission control, not all wiki pages are by default editable, only users with proper permissions can access correspondent wiki pages, this is the same concept of roles in collaborative writing systems.

History. This function is similar to revisions in a versioning system, which basically saves all revisions of a wiki page. Clicking on the "history" button will bring a page with available revisions of a specified wiki page. Every revision can be opened, edited and saved again. Saving a previous revision of a wiki page means to revert the content of the page to its previous version. In this way, the history function allows users to accept and reject changes made to a wiki page. More advanced wiki systems provide "Diff" function with history, so the user can compare two revisions and view their differences with visual representations

Links. In a wiki system, each page can be linked to other pages with specified wiki markup languages. The link is like a hypertext link on the web. If a respective linked page within a wiki does not yet exist, clicking on that link will bring the user to an edit window for creating content of the empty page.

Recent Changes. This function varies in wiki systems. For some wiki systems, "recent changes" provides a list of certain number of recent changes to a wiki system. For other wiki systems, it provides a list of all changes within a predefined time period. The list is produced automatically and cannot be changed by users.

SandBox. Wikis provide SandBox or PlayGround for newbie to learn how to use the wiki syntax and try the results without having to use a regular page. A Sandbox is just a predefined wiki page without particular content, and its content is cleared regularly.

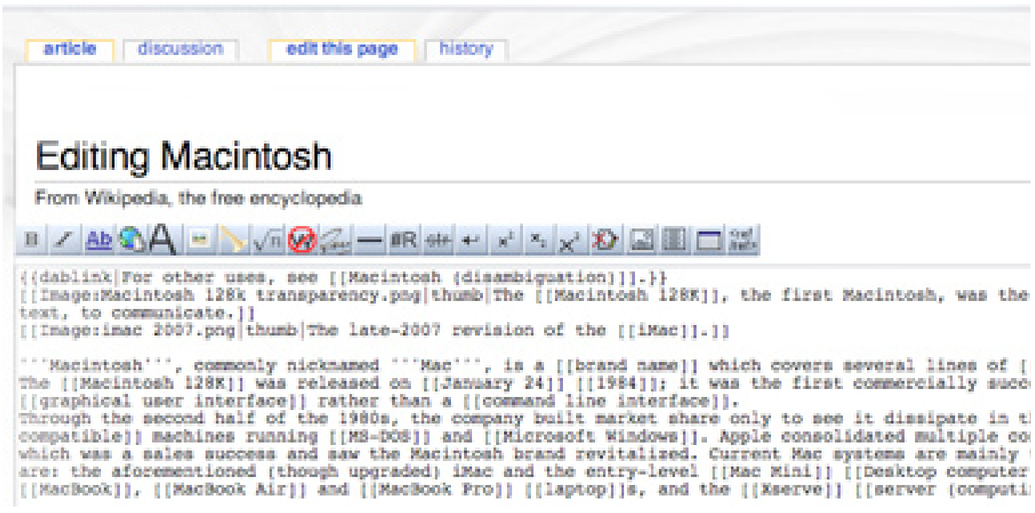
Search function. Search function is provided for the user to search either full-text or titles within a wiki site. So the user can access a wiki page quickly.

Due to the variety of wikis, it is impossible to find one wiki to represent all wiki systems. MediaWiki is used in this thesis, bacause it is one of the most popular wikis used on the web. Other wiki systems are mentioned as comparisons.

As described before, a typical editable wiki page has an "Edit" button on the page. When the user clicks on it, the content of the page becomes a <textarea> element in HTML standard, which allows the user to edit content within it. In MediaWiki, this button is "Edit this page". In addition to "Edit" button, there is another button called "History", which lists all revisions of the page. See Figure 14 for an example. The "History" button may be called "Revisions" in other wiki systems. The information displayed on the history page changes with different wiki systems.

For MediaWiki, a history list contains the following information: time when the page was modified, the user who modified the page, size of the page, comment from the user, and undo function which enables the user to revert back to a specific revision. To view changes between two revisions, the user can select them and click "Compare selected versions" to read the differences of the revisions. See Figure 15 in the next page for an example.

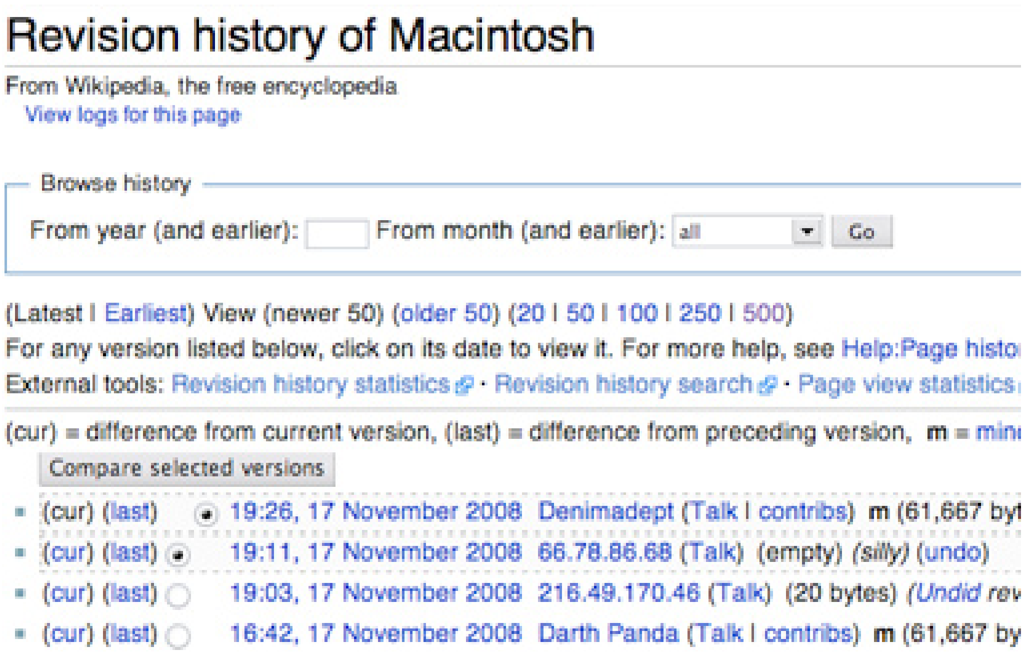
Diff is the most common algorithm and program used by wiki systems. For the three issues that are relevant to change representations, the change report displays all changes; there are no options for the user to choose what to report. The changes



```
{{dablink|For other uses, see [[Macintosh (disambiguation)]].}}
[[Image:Macintosh 128k transparency.png|thumb|The [[Macintosh 128K]], the first Macintosh, was the
text, to communicate.]]
[[Image:imac 2007.png|thumb|The late-2007 revision of the [[iMac]].]]

'''Macintosh''', commonly nicknamed '''Mac''', is a [[brand name]] which covers several lines of [
The [[Macintosh 128K]] was released on [[January 24]] [[1984]]; it was the first commercially succ
[[graphical user interface]] rather than a [[command line interface]].
Through the second half of the 1980s, the company built market share only to see it dissipate in t
compatible]] machines running [[MS-DOS]] and [[Microsoft Windows]]. Apple consolidated multiple co
which was a sales success and saw the Macintosh brand revitalized. Current Mac systems are mainly
are: the aforementioned (though upgraded) iMac and the entry-level [[Mac Mini]] [[Desktop computer
[[MacBook]], [[MacBook Air]] and [[MacBook Pro]] [[laptop]]s, and the [[Xserve]] [[server (computi
```

Figure 14. A typical wiki page in edit mode, "edit" and "history" buttons are at top.



Revision history of Macintosh

From Wikipedia, the free encyclopedia
View logs for this page

Browse history
From year (and earlier): From month (and earlier): all Go

(Latest | Earliest) View (newer 50) (older 50) (20 | 50 | 100 | 250 | 500)
For any version listed below, click on its date to view it. For more help, see Help:Page histo
External tools: Revision history statistics · Revision history search · Page view statistics

(cur) = difference from current version, (last) = difference from preceding version, m = min
Compare selected versions

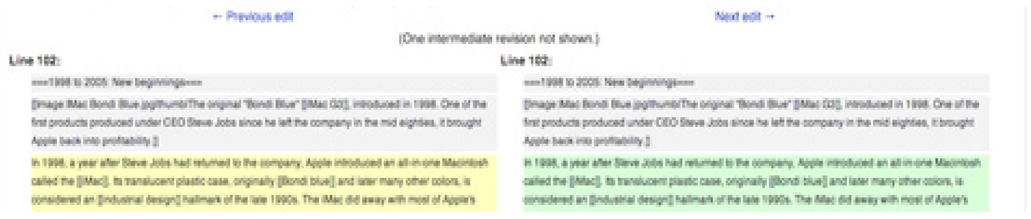
- (cur) (last) ● 19:26, 17 November 2008 Denimadept (Talk | contribs) m (61,667 byt
- (cur) (last) ● 19:11, 17 November 2008 66.78.86.68 (Talk) (empty) (*silly*) (undo)
- (cur) (last) ○ 19:03, 17 November 2008 216.49.170.46 (Talk) (20 bytes) (*Undid rev*
- (cur) (last) ○ 16:42, 17 November 2008 Darth Panda (Talk | contribs) m (61,667 by

Figure 15. History of a page in MediaWiki

are represented with two columns and the output format is the context format in Diff, where any changed paragraphs are shown alongside unchanged paragraphs before and after. The left column is the original text, while the right column is the changed text. For the visual highlight, the background color of the changed paragraph in the original page is light yellow, while it is light green in the revised version. The color of the texts that are exactly changed is red.

There is a "Previous edit" link above the text in the left column, clicking on it will move the comparing revision one revision older if it is not the first revision and display the change report. Similarly, there is a "Next edit" link above the text in the right column, and a click on it will move the comparing revision one revision newer if it is not the last revision and display the change report. An example is given as Figure 16.

Wiki as a web-based collaborative writing tool has changed the usual understanding about collaborative writing. The wiki way favors: content over form, open editing over security and control, free form content over structured content, and incremental growth over upfront design [Désilets *et al.*, 2005]. It seems conflicting to some requirements of collaborative writing systems, however, with the improvement of wiki systems, the conflicting parts are solved. Permission



(One intermediate revision not shown.)

← Previous edit	Next edit →
Line 102:	Line 102:
===1998 to 2005: New beginnings===	===1998 to 2005: New beginnings===
[[Image:iMac Bondi Blue.jpg|thumb|The original "Bondi Blue" [[iMac G3]], introduced in 1998. One of the first products produced under CEO Steve Jobs since he left the company in the mid eighties, it brought Apple back into profitability.]]	[[Image:iMac Bondi Blue.jpg|thumb|The original "Bondi Blue" [[iMac G3]], introduced in 1998. One of the first products produced under CEO Steve Jobs since he left the company in the mid eighties, it brought Apple back into profitability.]]
In 1998, a year after Steve Jobs had returned to the company, Apple introduced an all-in-one Macintosh called the [[iMac]]. Its translucent plastic case, originally [[Bondi blue]] and later many other colors, is considered an [[industrial design]] hallmark of the late 1990s. The iMac did away with most of Apple's	In 1998, a year after Steve Jobs had returned to the company, Apple introduced an all-in-one Macintosh called the [[iMac]]. Its translucent plastic case, originally [[Bondi blue]] and later many other colors, is considered an [[industrial design]] hallmark of the late 1990s. The iMac did away with most of Apple's

Figure 16. Change report of MediaWiki

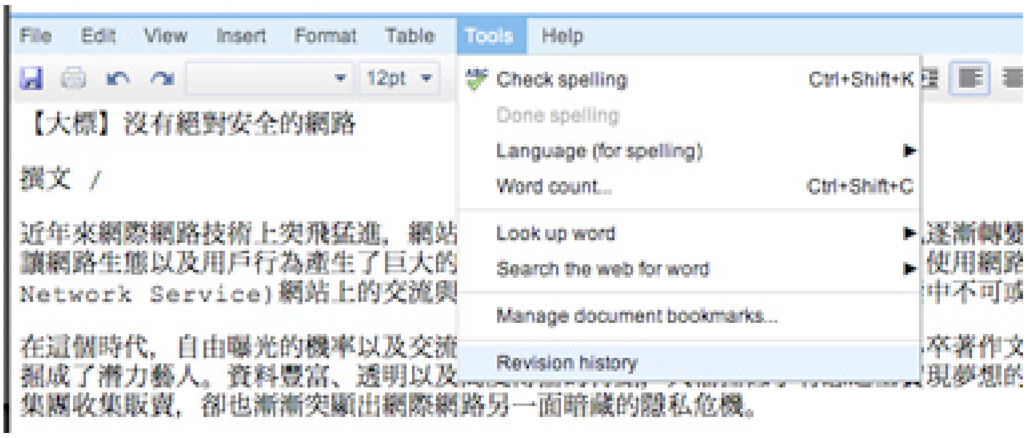
control allows control on editing content; with careful design and control, the structure of content can be maintained. The wiki way may favor doing collaborative writing in some way, but it doesn't mean that the user cannot do collaborative writing in the traditional ways.

Despite of the conceptual challenges, there are challenges in using wiki systems. Before web, users have been accustomed to standalone word processors, and switching to web browser means switching from a familiar environment to a new one. For example, edit button may sound simple but it is actually a barrier for using wikis, especially to wiki novices. The procedure to edit a document with word processors is open the file, then edit. But with wikis, the procedure is: open the link to the document, click edit button, then edit. It is a pity that there are few studies focusing on wiki usability, and among them, even fewer concern the change representations in wiki systems [Désilets *et al.*, 2004; Reitmayr, 2006; Paul, 2006].

One observation by Wei *et al.*[2005] is related to change representations: wiki editing can also intimidate users new to the collaborative environment. If collaborative writers and editors are accustomed to the visual cues offered by Microsoft Word as they edit documents using tracking and comment boxes, the opaque nature of these activities in wiki editing may be unsettling. It may take demonstrations to reassure the novice editor that edits are recorded, if not denoted visually, and can be compared in the revision history [Wei *et al.*, 2005].

3.2.2 Google Docs

Google intends to solve this usability gap by providing user interfaces similar to Microsoft Word for its web-based collaborative writing service: Google Docs. When the user opens a document in Google Docs, it is editable already; no "Edit" button is required, as shown in Figure 17. Google Docs provides the common formatting functions available on Word to Google Docs users as well. However, there is not much improvement in the change representations.



【大標】沒有絕對安全的網路

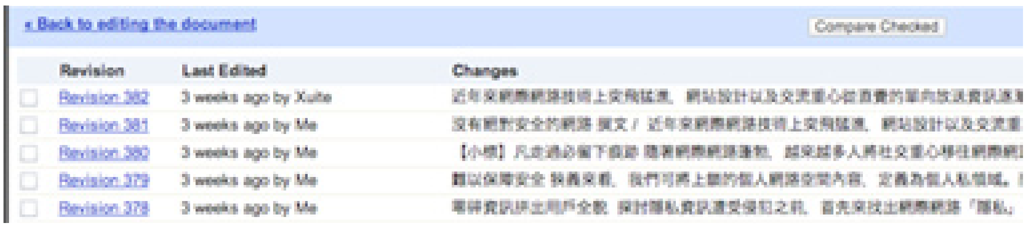
撰文 /

近年來網際網路技術上突飛猛進，網站 逐漸轉變
讓網路生態以及用戶行為產生了巨大的 使用網路
Network Service)網站上的交流與 中不可或

在這個時代，自由曝光的機率以及交流 卒著作文
掘成了潛力藝人。資料豐富、透明以及 現夢想的
集團收集販賣，卻也漸漸突顯出網際網路另一面暗藏的隱私危機。

Figure 17. Google Docs provides UIs similar to Microsoft Word.

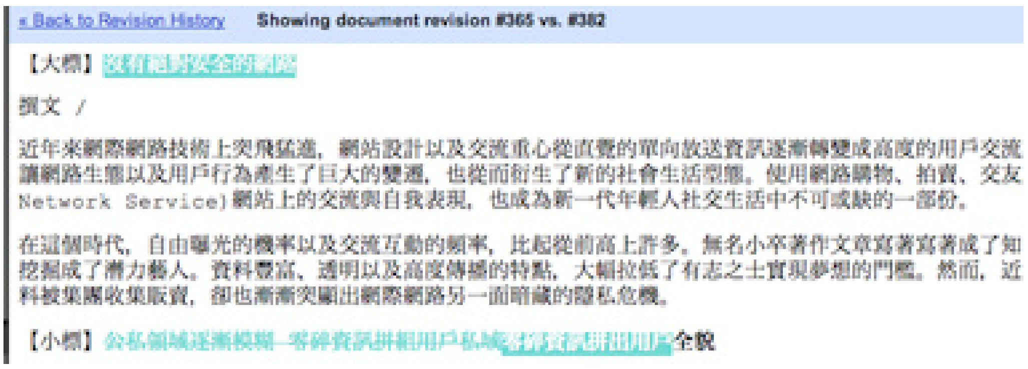
The change representation functions are accessed from “Revision History” in “Tools” on the menu bar. Like in general wiki systems, a list of revisions is displayed after clicking on “Revision history” (Figure 18). The information displayed on the list for every revision includes: revision number, the time the revision is made, the user who edits this revision, and the changed text. The user can choose two arbitrary revisions and see their difference by clicking “Compare Checked”.



	Revision	Last Edited	Changes
☐	Revision 382	3 weeks ago by Xuite	近年來網際網路技術上突飛猛進，網站設計以及交流重心從直覺的單向放送資訊逐漸
☐	Revision 381	3 weeks ago by Me	沒有絕對安全的網路 撰文 / 近年來網際網路技術上突飛猛進，網站設計以及交流重
☐	Revision 380	3 weeks ago by Me	【小標】凡走過必留下痕跡 隨著網際網路蓬勃，越來越多人將社交重心移往網際網
☐	Revision 379	3 weeks ago by Me	[illegible]以保障安全 [illegible]，我們可將上網的個人網路空間內容，定義為個人私領域。
☐	Revision 378	3 weeks ago by Me	零碎資訊拼出用戶全貌 探討隱私資訊遭受侵犯之前，首先來談談網際網路「隱私」

Figure 18. Revision history list in Google Docs

Unlike the two-column comparison report in wikis, Google Docs generates the change report similar to a Word document with “Highlight changes on screen” option switched on (Figure 19). There are few interactions with a change report. To compare with other revisions, the user has to go back to the Revision history page, to revert to a specific revision; the user then opens that revision and saves it.


« Back to Revision History
Showing document revision #365 vs. #382

【大標】沒有絕對安全的網路

撰文 /

近年來網際網路技術上突飛猛進，網站設計以及交流重心從直覺的單向放送資訊逐漸轉變成高度的用戶交流
讓網路生態以及用戶行為產生了巨大的變遷，也從而衍生了新的社會生活型態。使用網路購物、拍賣、交友
Network Service)網站上的交流與自我表現，也成為新一代年輕人社交生活中不可或缺的一部份。

在這個時代，自由曝光的機率以及交流互動的頻率，比起從前高上許多。無名小卒著作文章寫著寫著成了知
挖掘成了潛力藝人。資料豐富、透明以及高度傳播的特點，大幅拉低了有志之士實現夢想的門檻。然而，近
料被集團收集販賣，卻也漸漸突顯出網際網路另一面暗藏的隱私危機。

【小標】~~公私領域逐漸模糊—零碎資訊拼湊用戶私域~~零碎資訊拼出用戶全貌

Figure 19. Changes report in Google Docs

3.3 Summary

Observing the interaction with change representations in desktop collaborative writing systems and web-based collaborative writing systems in detail, it is obvious that the main problem of change representation in web-based collaborative systems is how to write a document with change representations at the same time. The users are not able to view changes while editing a document whether it is a wiki page or a Google Docs document. This is very different from the way of revising a document with desktop collaborative writing applications.

In general wiki systems, the user has to explicitly click “Edit” or “Edit this page” to edit a wiki page, which brings the user away from the original list of revision history or the change reports generated by the compare function. In Google Docs, after reading a change report, the user has to go back to the Google Docs homepage, and select the document again in order to edit it. During the editing or reviewing process, there are no functions to see the changed text unless the user opens another browser window or browser tab. In this way, the performance of change representation in collaborative writing is weakened, which is the reason why I derived the concept design of interacting with change representation in web-based collaborative writing systems. The goal of my design is to make it possible for the users to access the change report and edit a document at the same time within the same browser window in a web-based collaborative writing tool.

In the next chapter, I will explain this concept design in detail.

4. The Concept Design

As discussed in the previous chapter, lacking the possibility to view the change report and edit a document simultaneously in a web-based collaborative writing environment is a weakness for collaboration in the web-based environment. Thus, the design proposed here is aimed at compensating for this weakness in current web-based collaborative writing systems, providing a more intuitive way for the user to browse changes between revisions while editing a document simultaneously. This goal can be easily achieved on nowadays-desktop word processors with collaborative writing features, but is difficult to achieve in a web-based environment.

4.1 The Concept Design in Detail

My approach is to add change bars and pop-up windows to the current edit area in the web-based collaborative writing environment. When the user wants to edit a page/document, clicking on the "Edit" button or double clicking on the document will bring the currents of the current window to a browser window in which the texts are editable within the <textarea>. If there are revisions before the current version, change bars are displayed at the left side of the paragraphs that have been modified. Clicking on the change bar will bring out a pop-up window, which displays the change report of that selected paragraph compared to the latest revision.

The change representation in the change report is to display all changed text like the one in Microsoft Word with "Highlight changes on screen" option turned on: inserted texts are indicated by color with underline style, deleted text are represented by color with strike-through style. In addition to changed text, there is information about the compared revision on the pop-up window: author, comment to the changes, and revision number.

For interactions, there are two buttons in the pop-up window: previous revision and next revision. If the currently compared revision is neither the latest one nor the first one, both buttons are clickable. Clicking on the "Previous revision" button will compare the currently edited revision to the revision which precedes than the current one and display the new change report. Clicking on the "Next revision" button will compare the currently edited revision to the revision that follows the current one and display the new change report.

In general collaborative writing, especially back and forth reviewing process, what the co-author/reviewer wants to know is the differences between the current revision and its previous (few) revisions, so there may not be much help with the ability to browse difference reports among all revisions. But it can help if the user wants to know the evolution of a document with time.

4.1.1 The Benefit of Pop-up Window

Since there are different purposes for indicating changes and displaying changes, it has been suggested that there should be easy transition between indication and display mode in representing changes [Kim and Eklundh, 2002]. The pop-up window is aimed to fulfill the requirement of easy transition. This idea is from the dialogue box in desktop word processors. A dialogue box is a small window or message box that appears temporarily in a GUI (graphical user interface) to alert the user to a condition and/or to request information. Microsoft Word deploys a dialogue box to help users find changes in a revised document. In the web environment, a pop-up window can function as a dialogue box in the Windows environment.

Displaying the change report in a pop-up window enables users to access the change report of the currently edited document at the same time, and within the same main window, so it can provide a similar user experience as revising a document with word processors in the Windows environment. In this way, it is error preventing, less distracting and more efficient for users when they need to see the change report while reviewing or editing a document online.

Under the design of Wikipedia and Google Doc, if users want to see the change report of a document that is being edited, they have to explicitly open another browser window or tab that displays the revision history, and then choose the desired revisions to gain the compared change report. Take MediaWiki for example, clicking on the “History” button will immediately change the current editing page to the page of a revision history, without saving the content or asking the user to confirm such operation beforehand. Therefore the user suffers from the loss of content that has not been saved.

The way to keep a change report available while editing is to either open another browser window, or another tab within the same browser window, and use this window / tab to manipulate revision history. In this way, the user has to switch between windows / tabs; this approach not only increases the operations but also distracts the users’ concentration. The following figures display the steps for accessing a change report while editing a document in the web environment with MediaWiki (5 steps), Google Docs (6 steps) and my design (3 steps) respectively.

The reason to adopt change bars (highlight changes by indication) in the

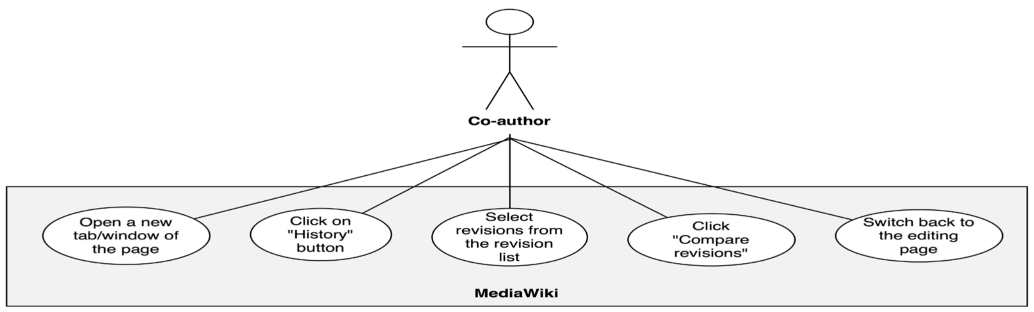

Co-author
Open a new tab/window of the page
Click on "History" button
Select revisions from the revision list
Click "Compare revisions"
Switch back to the editing page
MediaWiki

Figure 20. Steps to access to change report while editing in MediaWiki

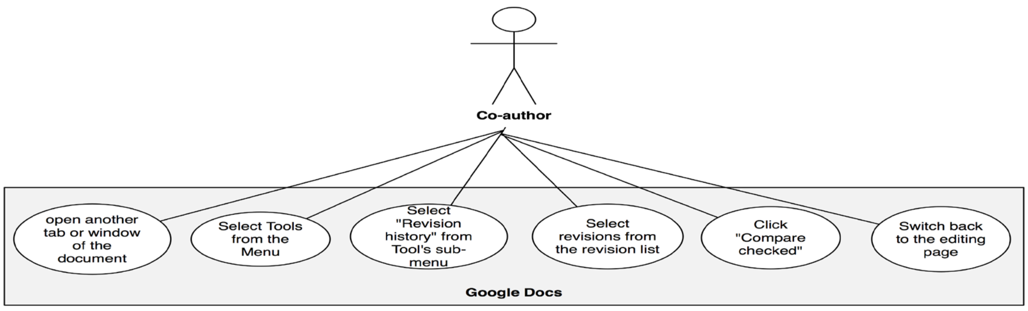

Co-author
open another tab or window of the document
Select Tools from the Menu
Select "Revision history" from Tool's sub-menu
Select revisions from the revision list
Click "Compare checked"
Switch back to the editing page
Google Docs

Figure 21. Steps to access to change report while editing in Google Docs

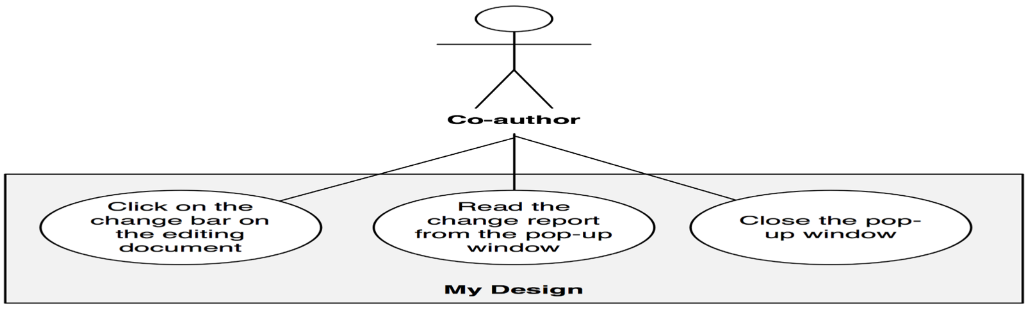
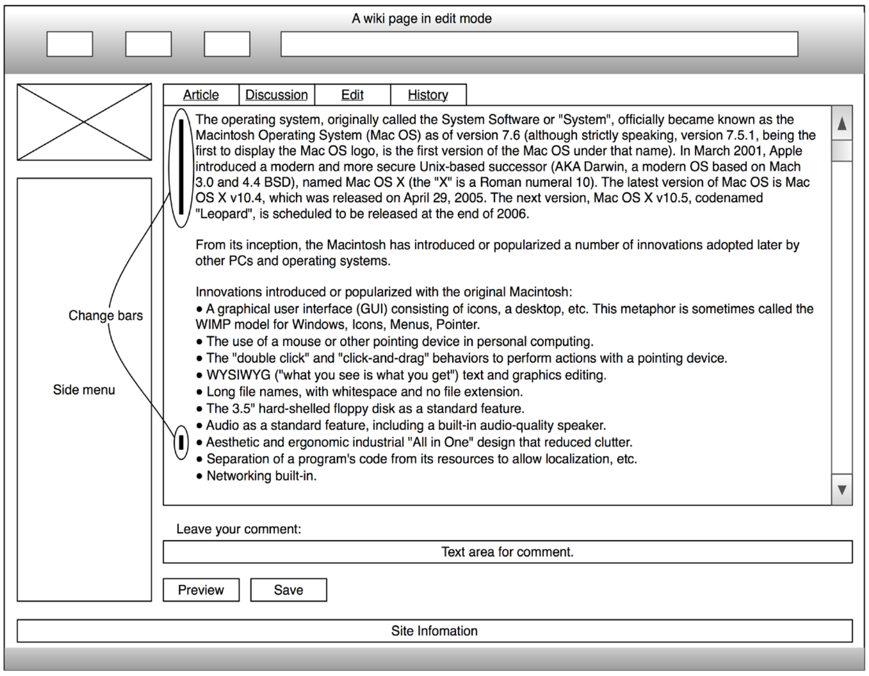

Co-author
Click on the change bar on the editing document
Read the change report from the pop-up window
Close the pop-up window
My Design

Figure 22. Steps to access to change report while editing with my design

<textarea> is to reduce the distraction to co-authors or reviewers when they are reading the document. As it is known that the “edit-review-incorporate” cycle is common in a co-authoring relationship, it is reasonable to assume that a co-author reads a text before making changes to it. When the user wants to further understand the rationales of changes, the purpose can be reached by clicking on the change bar to read the changes in detail from the pop-up window. If the user simply wants to revise the document without putting too much attention on understanding the changes, change bars are less annoying than displaying all changes, but still provide the possibility for the user to check changes in case he or she wants to. In this way, the design supports both needs of representing changes by indication and by display [Kim and Eklundh, 2002].

4.1.2 Display Changes by Paragraph

As for the unit of granularity of changed text, there are several choices: by word, by sentence, by paragraph, and others. The paragraph is considered suitable when dealing with documents [Malcolm and Gaines, 1991]. The basic structure of a document consists of word, sentences, paragraphs, sections and chapters, in which paragraph is a natural conceptual unit [Halliday and Hasan, 1976]. From the cognitive point of view, a paragraph can provide enough information for the reader to understand the context of a piece of writing without being too long, so that it prevents the pop-up window from occupying too much space on the browser window. In addition to cognitive advantage, because the difference programs used by existing wiki systems are either line-based (diff) or word-based (wdiff), it requires more effort to develop a sentence-based, section-based or chapter-based difference program. The Diff program decides the ending of a line by an escaped character called “new line”, which is defined as “\n” in programming languages. When the user presses an “Enter” to start a new paragraph, “\n” is added to the document. Therefore, it is easier to parse a document into paragraphs than to parse it into sentences or other structures.

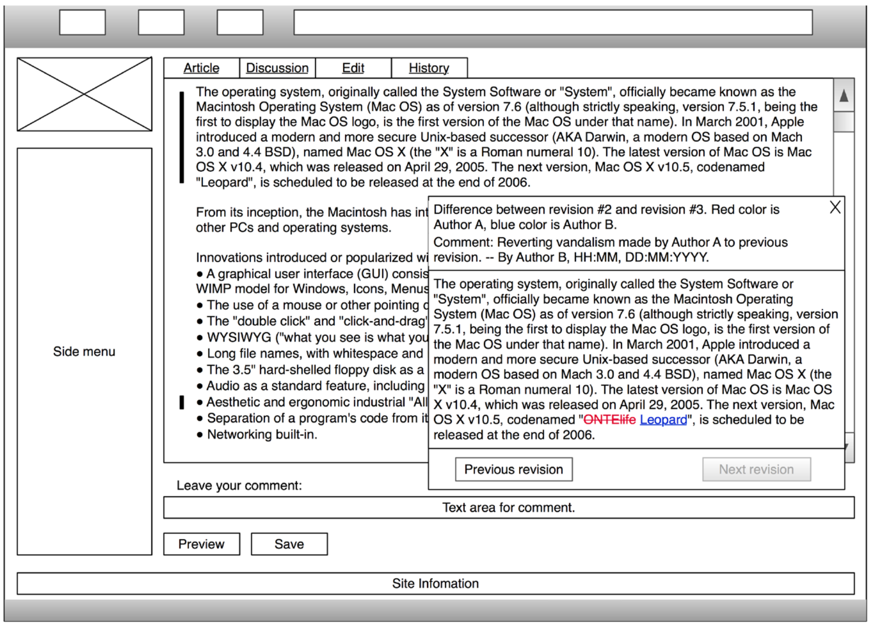


The operating system, originally called the System Software or "System", officially became known as the Macintosh Operating System (Mac OS) as of version 7.6 (although strictly speaking, version 7.5.1, being the first to display the Mac OS logo, is the first version of the Mac OS under that name). In March 2001, Apple introduced a modern and more secure Unix-based successor (AKA Darwin, a modern OS based on Mach 3.0 and 4.4 BSD), named Mac OS X (the "X" is a Roman numeral 10). The latest version of Mac OS is Mac OS X v10.4, which was released on April 29, 2005. The next version, Mac OS X v10.5, codenamed "Leopard", is scheduled to be released at the end of 2006.

From its inception, the Macintosh has introduced or popularized a number of innovations adopted later by other PCs and operating systems.

Innovations introduced or popularized with the original Macintosh:
- A graphical user interface (GUI) consisting of icons, a desktop, etc. This metaphor is sometimes called the WIMP model for Windows, Icons, Menus, Pointer.
- The use of a mouse or other pointing device in personal computing.
- The "double click" and "click-and-drag" behaviors to perform actions with a pointing device.
- WYSIWYG ("what you see is what you get") text and graphics editing.
- Long file names, with whitespace and no file extension.
- The 3.5" hard-shelled floppy disk as a standard feature.
- Audio as a standard feature, including a built-in audio-quality speaker.
- Aesthetic and ergonomic industrial "All in One" design that reduced clutter.
- Separation of a program's code from its resources to allow localization, etc.
- Networking built-in.

Figure 23. Wiki page at editing condition

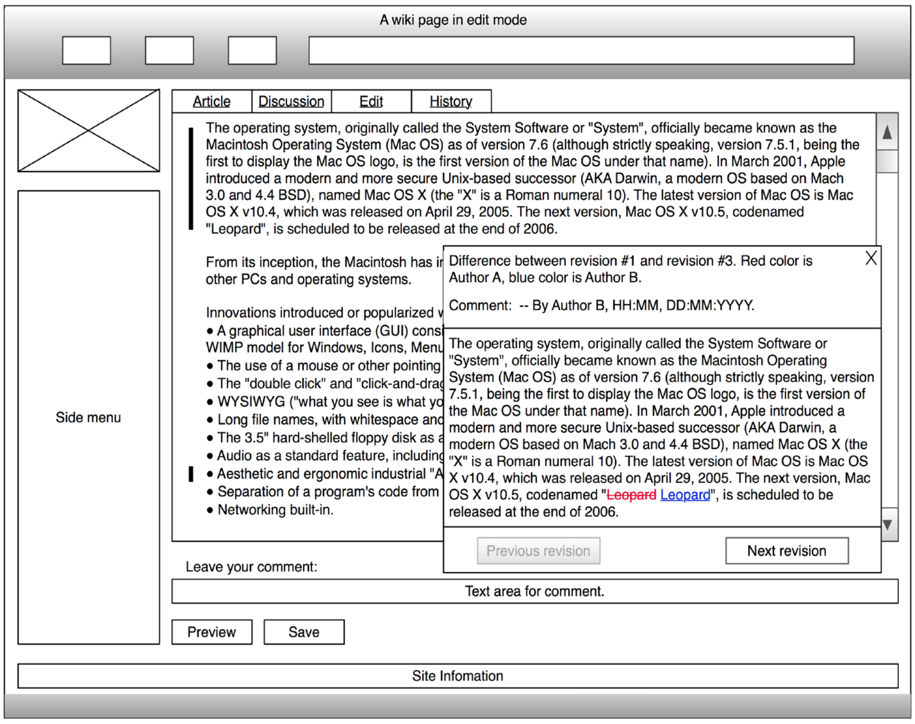


Difference between revision #2 and revision #3. Red color is Author A, blue color is Author B.

Comment: Reverting vandalism made by Author A to previous revision. -- By Author B, HH:MM, DD:MM:YYYY.

The operating system, originally called the System Software or "System", officially became known as the Macintosh Operating System (Mac OS) as of version 7.6 (although strictly speaking, version 7.5.1, being the first to display the Mac OS logo, is the first version of the Mac OS under that name). In March 2001, Apple introduced a modern and more secure Unix-based successor (AKA Darwin, a modern OS based on Mach 3.0 and 4.4 BSD), named Mac OS X (the "X" is a Roman numeral 10). The latest version of Mac OS is Mac OS X v10.4, which was released on April 29, 2005. The next version, Mac OS X v10.5, codenamed "~~ONTElife~~ Leopard", is scheduled to be released at the end of 2006.

Figure 24. A Wiki page at editing condition and a pop-up window. When a change bar is clicked, the pop-up window appears and displays the change report of that selected paragraph compared to the latest revision. The inactive “Next revision” button indicates there is no newer revision, while the active “previous revision” button indicates there are older revisions.



The operating system, originally called the System Software or "System", officially became known as the Macintosh Operating System (Mac OS) as of version 7.6 (although strictly speaking, version 7.5.1, being the first to display the Mac OS logo, is the first version of the Mac OS under that name). In March 2001, Apple introduced a modern and more secure Unix-based successor (AKA Darwin, a modern OS based on Mach 3.0 and 4.4 BSD), named Mac OS X (the "X" is a Roman numeral 10). The latest version of Mac OS is Mac OS X v10.4, which was released on April 29, 2005. The next version, Mac OS X v10.5, codenamed "Leopard", is scheduled to be released at the end of 2006.

From its inception, the Macintosh has i
other PCs and operating systems.

Innovations introduced or popularized v
• A graphical user interface (GUI) cons
WIMP model for Windows, Icons, Menu
• The use of a mouse or other pointing
• The "double click" and "click-and-drag
• WYSIWYG ("what you see is what yo
• Long file names, with whitespace and
• The 3.5" hard-shelled floppy disk as a
• Audio as a standard feature, including
• Aesthetic and ergonomic industrial "A
• Separation of a program's code from
• Networking built-in.

Difference between revision #1 and revision #3. Red color is Author A, blue color is Author B.

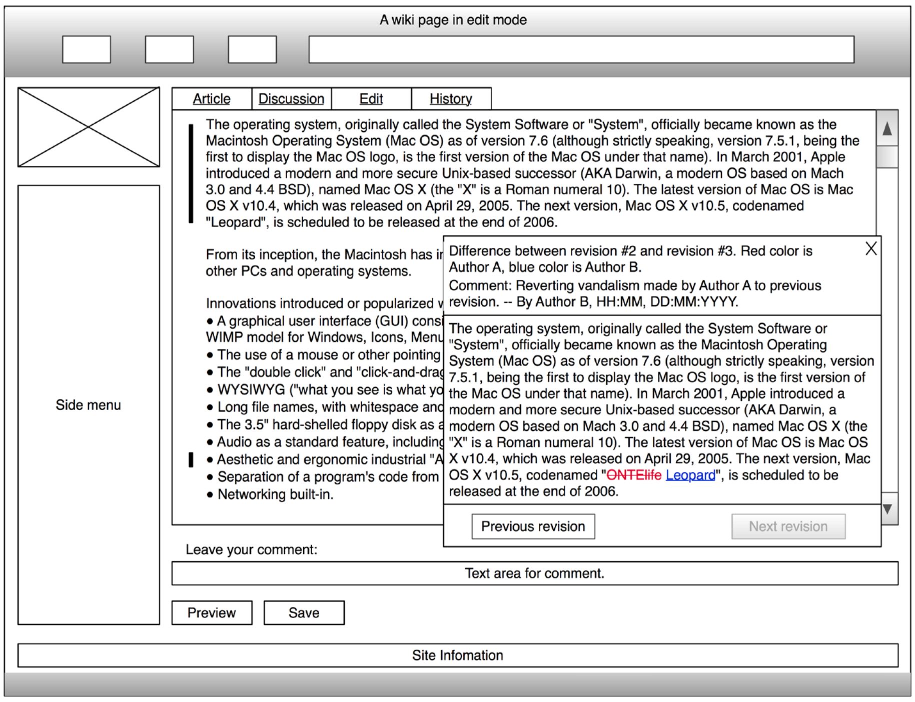
Comment: -- By Author B, HH:MM, DD:MM:YYYY.

The operating system, originally called the System Software or "System", officially became known as the Macintosh Operating System (Mac OS) as of version 7.6 (although strictly speaking, version 7.5.1, being the first to display the Mac OS logo, is the first version of the Mac OS under that name). In March 2001, Apple introduced a modern and more secure Unix-based successor (AKA Darwin, a modern OS based on Mach 3.0 and 4.4 BSD), named Mac OS X (the "X" is a Roman numeral 10). The latest version of Mac OS is Mac OS X v10.4, which was released on April 29, 2005. The next version, Mac OS X v10.5, codenamed "~~Leopard~~ Leopard", is scheduled to be released at the end of 2006.

Leave your comment:

Text area for comment.

Figure 25. A Wiki page at editing condition and a pop-up window. When the "Previous revision" button is clicked, the system compares the currently edited revision to the revision which precedes than the current one and display the new change report in the pop-up window. The inactive "Previous revision" button indicates there is no older revision, while the active "Next revision" button indicates there are newer revisions. The filed at top of the pop-up windows displays the information of revisions.



The operating system, originally called the System Software or "System", officially became known as the Macintosh Operating System (Mac OS) as of version 7.6 (although strictly speaking, version 7.5.1, being the first to display the Mac OS logo, is the first version of the Mac OS under that name). In March 2001, Apple introduced a modern and more secure Unix-based successor (AKA Darwin, a modern OS based on Mach 3.0 and 4.4 BSD), named Mac OS X (the "X" is a Roman numeral 10). The latest version of Mac OS is Mac OS X v10.4, which was released on April 29, 2005. The next version, Mac OS X v10.5, codenamed "Leopard", is scheduled to be released at the end of 2006.

From its inception, the Macintosh has i
other PCs and operating systems.

Innovations introduced or popularized v
• A graphical user interface (GUI) cons
WIMP model for Windows, Icons, Menu
• The use of a mouse or other pointing
• The "double click" and "click-and-drag
• WYSIWYG ("what you see is what yo
• Long file names, with whitespace and
• The 3.5" hard-shelled floppy disk as a
• Audio as a standard feature, including
• Aesthetic and ergonomic industrial "A
• Separation of a program's code from
• Networking built-in.

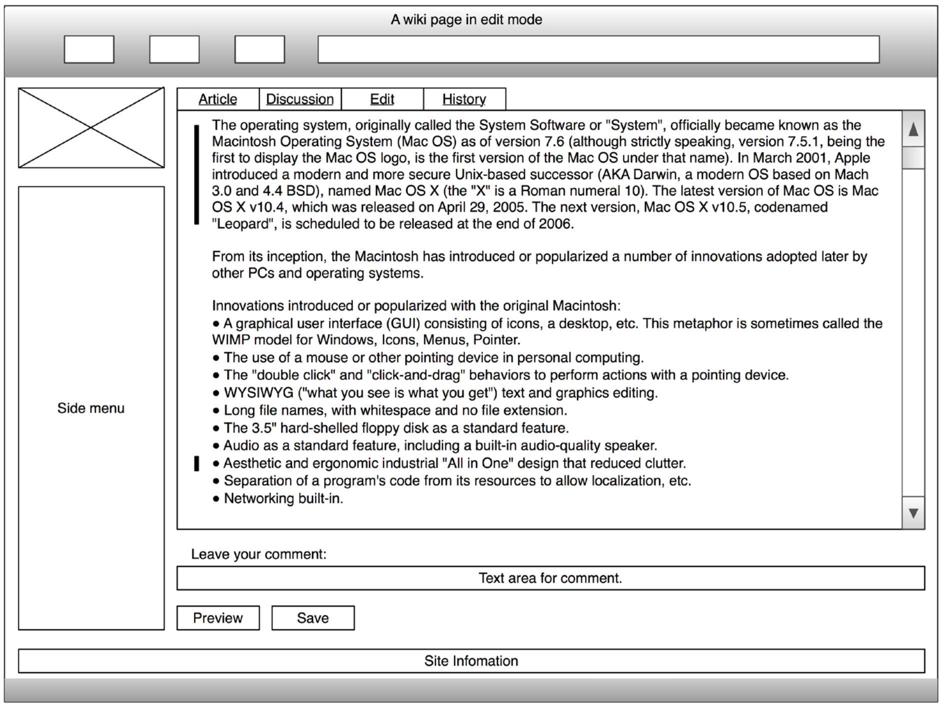
Difference between revision #2 and revision #3. Red color is Author A, blue color is Author B.
Comment: Reverting vandalism made by Author A to previous revision. -- By Author B, HH:MM, DD:MM:YYYY.

The operating system, originally called the System Software or "System", officially became known as the Macintosh Operating System (Mac OS) as of version 7.6 (although strictly speaking, version 7.5.1, being the first to display the Mac OS logo, is the first version of the Mac OS under that name). In March 2001, Apple introduced a modern and more secure Unix-based successor (AKA Darwin, a modern OS based on Mach 3.0 and 4.4 BSD), named Mac OS X (the "X" is a Roman numeral 10). The latest version of Mac OS is Mac OS X v10.4, which was released on April 29, 2005. The next version, Mac OS X v10.5, codenamed "~~ONTElife~~ Leopard", is scheduled to be released at the end of 2006.

Leave your comment:

Text area for comment.

Figure 26. A Wiki page at editing condition and a pop-up window. When the "Next revision" button is clicked, the system compare the currently edited revision to the revision that follows the current one and display the new change report in the pop-up window. The inactive "Next revision" button indicates there is no newer revision, while the active "Previous revision" button indicates there are older revisions. The filed at top of the pop-up windows displays the information of revisions.



The operating system, originally called the System Software or "System", officially became known as the Macintosh Operating System (Mac OS) as of version 7.6 (although strictly speaking, version 7.5.1, being the first to display the Mac OS logo, is the first version of the Mac OS under that name). In March 2001, Apple introduced a modern and more secure Unix-based successor (AKA Darwin, a modern OS based on Mach 3.0 and 4.4 BSD), named Mac OS X (the "X" is a Roman numeral 10). The latest version of Mac OS is Mac OS X v10.4, which was released on April 29, 2005. The next version, Mac OS X v10.5, codenamed "Leopard", is scheduled to be released at the end of 2006.

From its inception, the Macintosh has introduced or popularized a number of innovations adopted later by other PCs and operating systems.

Innovations introduced or popularized with the original Macintosh:
- A graphical user interface (GUI) consisting of icons, a desktop, etc. This metaphor is sometimes called the WIMP model for Windows, Icons, Menus, Pointer.
- The use of a mouse or other pointing device in personal computing.
- The "double click" and "click-and-drag" behaviors to perform actions with a pointing device.
- WYSIWYG ("what you see is what you get") text and graphics editing.
- Long file names, with whitespace and no file extension.
- The 3.5" hard-shelled floppy disk as a standard feature.
- Audio as a standard feature, including a built-in audio-quality speaker.
- Aesthetic and ergonomic industrial "All in One" design that reduced clutter.
- Separation of a program's code from its resources to allow localization, etc.
- Networking built-in.

Figure 27. A Wiki page at editing condition. Clicking the x symbol in the pop-up window closes the pop-up window and brings back the Wiki page at editing condition.

The mockups of the web pages at every step are illustrated as following:

It is important to keep in mind that this design is not meant to replace the function of "History"; its aim is to provide extra functionality that cannot be achieved by the current "History" function. The revision history keeps a revision as a whole without any changes recorded; it fulfills the requirement if the user wants to read a revision of a specific page/document. The pop-up window displays the change report of a specific paragraph in a page/document that is being edited on the fly; it fulfills the requirement if the co-author/reviewer wants to know the change history of a paragraph while modifying a page/document.

4.2 Technical Analysis

To implement this design, there are several technical challenges to overcome. Although the actual implementation is beyond the scope of this thesis, the following analysis provides necessary information to accomplish the implementation.

4.2.1 Front End

Making change bars visible and clickable in a wiki page under edit mode is a challenge for implementation, because most wiki pages are edited with plain text. For a wiki page, it is straightforward to think of representing change bars with the standard HTML (HyperText Markup Language) and CSS (Cascading Style Sheet). However, although visual codes and formatting are common in current WYSIWYG (What You See is What You Get) editors, they are not widely available in wiki systems.

The traditional way to edit a wiki page is to edit the content within a <textarea> element, in which all texts are plain text without visual effects; specific wiki markup is used if the user wants to add formats or hypertext to the content. The visual effects and formats are only visible after the user clicks "Preview" or "Save". The wiki markup will be converted to HTML format and rendered as HTML pages. The visual effects are not available in a wiki page under edit mode.

Current web development technologies have made it possible. The <textarea> used to receive input can be substituted by a frame element. HTML allows a browser window to be split into several frames. Each of these frames forms a window of its own within the content window (or within other frames). In this way, content in a frame is displayed as html format instead of plain text, with the help of JavaScript and DOM (Document Object Model); the content in a frame can be set to editable and retains its formatted representation.

A common approach is to use an iframe as a container of the content, then set iframe.document.designMode = "on" in JavaScript, which controls the behavior of the iframe. Control of mouse event on the change bars and actions within the pop-up window should be possible as well via JavaScript.

4.2.2 Back End

Other challenges are the data structure of the document, and the structure of the document version. The ability to view differences of a single paragraph in a document implies that the Diff program is applied to a paragraph in a document, not a whole document. However, this is not how the Diff program works. The Diff program reads two files and compares differences between them, it does not compare specific parts in the files, but compares the whole files.

To solve this problem, a data structure to store every document revision by paragraph is required, so that every individual paragraph can be retrieved by the system, and can be sent as input to the Diff program for comparison.

Following the document structure is the structure of document versions. Every paragraph has to know its previous revision and next revision if it has revisions, so the system knows where to obtain the target for comparing.

Since design and development of data structures for this purpose is beyond my goal, I do not go further into this. For those who are interested in the data structure design and development, a data structure for solving this problem proposed by Malcolm and Gaines can be considered as a reference [Malcolm and Gaines, 1991].

5. Pilot Usability Evaluation and Discussion

A small pilot usability evaluation was conducted to gain feedback on the concept design. This chapter will briefly describe the pilot test, its results, discussion and possible alternatives to the design.

5.1 The Pilot Test

The pilot test was done with a front-end prototype called ViewRevision without actual data, because the backend data structure was not available. There were four participants; two were computer science students, and two were general users with basic computer operation knowledge. The subjects were asked to complete three tasks: choose a document, find any part of the document they were interested in and browse its changes, and view changes in older revisions of the chosen paragraph. An interview was conducted after the usability test.

As for the result, all participants gave positive support to the idea. The participants had no difficulty clicking on change bars to bring up the pop-up window. However two participants found it confusing to relate the text in the edit window to the text in the pop-up window: one reason was because of the position of the pop-up window, the other reason was because of the text in the pop-up window. One participant suggested that instead of displaying the changes report directly to the users, the system should display a list of revisions for the users to choose which revision they want to compare.

5.1.2 Introduction to the Pilot Test Plan

In the Pilot Usability evaluation of ViewRevision, the usability of ViewRevision (a stand-along program) was evaluated. Especially, the evaluation focused on the convenience of browsing changes of selected paragraphs in different versions of one document. The usability of this program was evaluated using usability testing. The goal of the evaluation was to find out whether it is useful to users browsing revisions as well as ideas for developing the program to be more usable.

In the usability test, the users were provided with pre-formulated test tasks (prepared so that they will target the issues that are on the focus of the evaluation). Think-aloud approach was used to illustrate the problem descriptions. In addition, data was collected via interviews with predefined questions.

The measures that were used to evaluate different aspects of the usability of ViewRevision are effectiveness, efficiency and satisfaction, described as follows:

1. Effectiveness
   - % of tasks successfully completed
2. Efficiency
   - Task times
   - Errors per task and the time that is spent overcoming the errors

3. Subjective satisfaction
   - How often and in which circumstances does the participant express signs of frustration or pleasantness (the behavior of the participant)?

These three measures were decided based on the definition of usability provided by ISO 9241-11[1998]. Depending on the goals of the evaluation, there are other measures available for usability evaluation.

5.1.3 ViewRevision and its Users

ViewRevision is a method for users to browse differences between revisions of the same document. There are mainly four functions:

1. Users choose the file and revision they want to view. The program displays the chosen version and compares it with the last version, and highlights different parts with change bars.
2. When the user moves the mouse to the change bar, the cursor changes to a clickable image for the user to click. When the user clicks on the change bar, it brings up a pop-up window; the paragraph chosen by the user will be displayed in the pop-up window. There are two buttons in the pop-up window: Previous and Next.
3. When the user clicks on the "Previous" button, the window should display the paragraph of the previous revision.
4. When the user clicks on the "Next" button, the window should display the paragraph of the next revision.

The target user groups of ViewRevision are those whose work is related to collaboratively written documents, for instance: authors, proofreaders, and editors.

For a document that is to be published, it has to be reviewed and revised by other editors and/or co-authors, so there will be revisions of the same document; if there are more editors, reviewers and authors, then there will be more revisions.

However, the testers in this pilot evaluation test would be general users.

5.2 Usability Testing

The pilot test was conducted on Wednesday, November 22nd, 2006 in the computer room of the Department of Computer Sciences of the University of Tampere.

5.2.1 Technical Context

The operating system used in the test was Windows XP in the computer room. The size of the screen was 15 inches and the resolution of the screen was 1024x768.

5.2.2 Participants

Test participant 1: A (male), a technical writer with computer sciences background, who is also familiar with wiki and version control systems.
Test participant 2: B (female), communication background, who has basic computer operation knowledge and working experience in media.
Test participant 3: C (male), general end users.
Test participant 4: D (female), computer sciences student.

5.2.3 Test Tasks

Practice task	Find news about Finland today
Start state	No application open
Rationale	The purpose of this task is to practice test procedure in general. The task is not related to the evaluated system.
End state	The moderator closes the Web browser and the test can begin. If the participant does not find the information, the moderator will show where the information can be found from.
Estimated task time	Less than 3 minutes.

Task 0	Open iTunes and play a song
Start state	No application open
Rationale	The purpose of this task is to relax the participant.
End state	iTunes plays a song.
Estimated task time	Less than 1 minute.

Task 1	Find a document and revision you want to read.
Start state	Main window of ViewRevision
Rationale	The purpose of this task is to find out how the users will choose the file.
End state	Document with the version chosen by the user is displayed on main window.
Estimated task time	Less than 30 seconds.

Task 2	Find a modified part on the document you are interested in and browse its changes.
Start state	Main window with the document.
Rationale	The purpose of this task is to know if the user can click on the change bar to bring the pop-up window.
End state	Pop-up window shows on the screen.
Estimated task time	Less than 1 minute.

Task 3	Find other versions of the paragraph.
Start state	Pop-up window with the paragraph chosen by the user.
Rationale	The purpose of this task is to know if the user can use Previous and/or Next buttons on the pop-up window.
End state	Pop-up window with the paragraph in another revision.
Estimated task time	Less than 1 minute.

5.2.4 Interview

Following the usability test, the users were interviewed. The type of the interview was face-to-face interview with pre-prepared questionnaires, but the participants did not have the questionnaire. The interview took 10 minutes at most.

The questions were:

1. What did you like about the project?
2. What did you not like about the project?
3. What was difficult to use during the exercise?
4. What did you find confusing?
5. What can be done to improve upon the project?

5.2.5 Collecting and Analyzing the Data

The data was collected through interviews with predefined questions. The interviewer took notes about the test to support the analysis. Quantitative (numerical) data was presented as graphs.

5.3 Report of the Pilot Test

This report is on the pilot usability evaluation of ViewRevision, the result is described in the following sections.

5.3.1 Task 1

This task required the users to find a document and revision they want to read. The purpose was to find out how the users would choose the file and the estimated time period for the task was 30 seconds. Two participants finished within 27 seconds, one participant within 22 seconds and another participant used 30 seconds.

5.3.2 Task 2

In task 2 the users were asked to find any part of the document they were interested in and browse its changes. The purpose was to know if the user could click on the change bar to bring up the pop-up window. One minute was allocated to this task. Three participants finished within 50 seconds, and one participant used 51 seconds.

5.3.3 Task 3

Task 3 required users to find other revisions of the chosen paragraph. The purpose of the task was to know the users' ability to use the Previous and Next buttons on the pop-up window. The results show complete 100% task completion. Two participants finished within 20 seconds, one within 10 seconds and the other used 15 seconds.

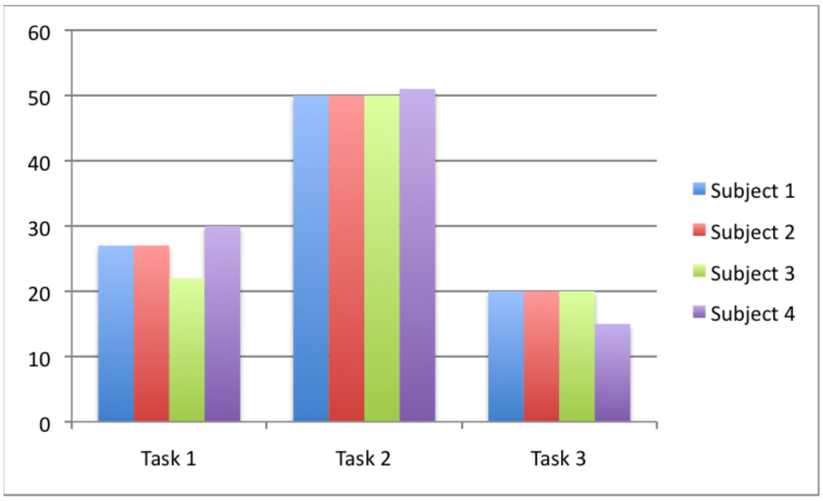


Figure 28. The time taken by every subject to finish each task.

5.3.4 Interview Analysis

In addition to the usability test, an interview was conducted. Questions asked were:

1. What did you like about the project?
2. What did you not like about the project?
3. What was difficult to use during the exercise?
4. What did you find confusing?
5. What can be done to improve upon the project?

All the four participants said that what they liked about the design was the idea, and they found nothing difficult in using it.

However, two participants complained about the same thing. The appearance of the selected text in the pop-up window was confusing: there seemed to be no connection between where the text was selected and where it appeared in the pop-up window.

Regarding the question what to do to improve on the project. three participants said that the idea is good so they hoped such a thing could come into existence.

One participant suggested that it would be good if there could be modifications to the UI. There is redundant use of words between the main title and the subtitle of the UI. In addition, the revision button should be inactive until a file is selected, and when there is only one version the button should not be active at all.

Another suggestion was that instead of displaying changes directly to the user, display a list of revisions for the users to choose which revision to compare.

5.3.5 Conclusion

The analysis shows that all the participants completed the tasks within the estimated given time. This shows an achievement of both efficiency and effectiveness. The chart on top of this page tells the time used to complete every task.

5.4 Discussion

The goal of the concept design was to improve the usability of interacting with the change representations in web-based collaborative writing systems. As analyzed in Chapter 3, two modern and well-known web-based collaborative writing systems are MediaWiki and Google Doc. For both systems, if the user wants to view thhe change report of a currently edited document, and edit the document at the same time, the user has to open one more browser window/tab and switch between two browser windows/tabs to achieve the purpose (as described in Chapter 4).

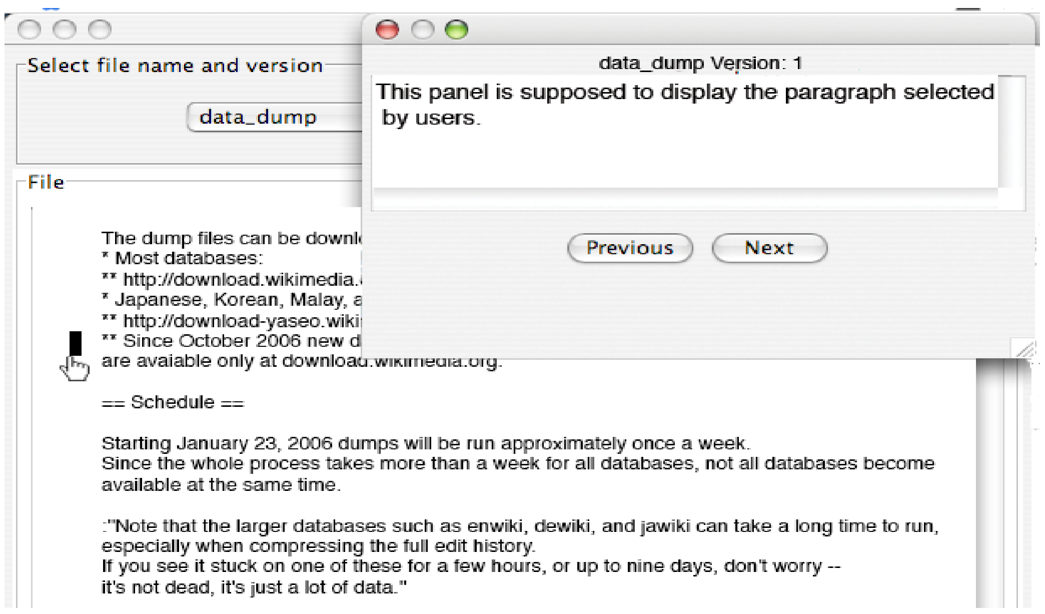
The design intends to increase the effectiveness and efficiency by reducing the steps of accessing the change report. By adopting the idea of pop-up windows from desktop word processors, the design reduces the number of steps required to access the change report from at least five steps to three steps. In addition, it as well reduces the number of steps required to view change reports of different revisions.

However, because it is only a concept design and is only evaluated with the pilot usability test, there is room to improve on this design. From the feedback of the pilot usability test, it is clear that there are more detailed issues about this design worth further studies as well.

Changing the design from a full web page to a pop-up window may cause some functions lost in the pop-up window, for example, choosing a specific revision to compare by user name. The pop-up window can help users access the change report more efficiently, but the improvement to its context-awareness is required. To improve the context-awareness, three aspects can be considered: position of the pop-up window, visual cues of changes in edit window, and alternatives to display changes in the pop-up window.

5.4.1 Limitation of the Pop-up Window

By adopting the pop-up window to display the change report and provide interaction with changes to users of web-based collaborative writing systems, the user can view differences of a paragraph of a document between two revisions immediately, without losing the ability to edit the document within the same browser window. This concept is inspired by the desktop word processor: Microsoft Word uses the pop-up window to let users choose which changes to accept or reject. In addition to simplify the procedures for accessing the change report, the pop-up window



This panel is supposed to display the paragraph selected by users.

The dump files can be downl
* Most databases:
** http://download.wikimedia.
* Japanese, Korean, Malay, a
** http://download-yaseo.wiki
** Since October 2006 new d
are avaiable only at download.wikimedia.org.

== Schedule ==

Starting January 23, 2006 dumps will be run approximately once a week.
Since the whole process takes more than a week for all databases, not all databases become available at the same time.

:"Note that the larger databases such as enwiki, dewiki, and jawiki can take a long time to run, especially when compressing the full edit history.
If you see it stuck on one of these for a few hours, or up to nine days, don't worry -- it's not dead, it's just a lot of data."

Figure 29. The position of the pop-up window in the pilot usability test

reduces the user's cognitive load as well, because the user does not have to switch between tabs or windows to view differences and edit a document.

What is neglected here is that the changing of the design restricts the way the user can do in the traditional design, for instance, access the revision history, or select a revision by specific criteria. In my opinion, access to the revision history and access to the change report between revisions are separate functions, so there should be different user interfaces designed for them respectively. The pop-up window is for access to the change report, not for access to the revision history.

5.4.2 Position of the Pop-up Window

In the pilot test, the position of the pop-up window was fixed to the up-right corner of the edit window as shown in Figure 29. But the edit area was actually in the middle of the window; therefore it caused confusion. Ideally, the pop-up window should be near the target paragraph but should not overlap, so it is within the user's visual range but it does not stop the user from reading on the edit window. But whether the pop-up window should be right above, right below the target paragraph, or to the down right corner or to the down left corner of the target paragraph remains unanswered. Previous researches proposed that side-by-side is cognitively fit to users who are accustomed to writing from left to right. Therefore I would assume that positioning the pop-up window to the down right corner of the target paragraph is a suitable choice. Still, more tests need to be done to verify this.

5.4.3 Change Bars or Color Coding on the Changed Text

The change indications on the edit window have space for improvement as well. The change bar indicates there are changes in the paragraph, but it can not point out the exact position where the change happens. Therefore, when there is a long paragraph with many modifications in different sentences, the user may not be able to see the connection between the texts on the edit window and the texts on the pop-up window. Should the position of change bars be changed from paragraphs to sentences? Or should the color-coding be used on changed texts to substitute for change bars? Because the first priority is to keep the integrity of the flow of the text so it helps reading and revising the text, so it is not suggested to display all changes on the edit window.

Instead of displaying change bars at the side of the changed paragraph, it is also possible to apply color-coding to the changed text. In that way, it is easier for the co-authors to tell the location of the changes in a paragraph they are focusing on in both the edit window and the pop-up window. However, it requires more effort to find a better solution.

5.4.4 Alternatives to Displaying Changes on the Pop-up Window

The last issue worth further investigation is the usage with revisions, which is related to the interaction with representing changes on the pop-up window. The requirement for the version control mechanism is the ability to gain changes between revisions [Posner and Baecker, 1992]. Therefore, the current design assumes that what the user expects to see is the differences between the currently edited document and its last revision. But it does not consider the frequency of usage of even older revisions. A more recent study actually shows that the revisions are used for reuse of deleted parts although the frequency of reusing them is low, but it gives the users feeling of security. On the other hand, six out of eleven interviewees in the study expressed their idea of the uselessness of playback function, which means being able to review revisions back and forth. Three out of the eleven interviewees expressed that they had difficulty to conceptualize the use of playback functions [Kim and Eklundh, 2001].

Is it really required to have “Previous revisions” and “Next revisions” on the pop-up window? Or is it necessary to allow the users to retrieve the change report between random revisions? Studies on how participants in collaborative writing projects use revisions can help solve the questions.

6. Conclusion

The goal of this thesis was to present a design for interacting with change representations on web-based collaborative writing systems such as wiki or Google Docs, and to explain in which part they improve the use of those systems.

The design aims to improve the efficiency of browsing the change report while editing a document at the same time for users of web-based collaborative writing systems. Compared to traditional design, which requires the users to open another browser window or tab in order to see change differences during editing a document online, the design introduces a pop-up window on the editing window to support the requirement of interacting with change reports.

By adopting a pop-up window, the steps to display a change report of a currently edited document on a web-based collaborative writing system are reduced, which implies the improvement of efficiency. Moreover, this design transplants the user experience on current desktop word processor software to web-based collaborative writing systems; therefore it is expected to increase the familiarity of desktop users when they are transformed to web-based environment.

The technical challenges and possible solutions are analyzed for reference to evaluate the possibility to implement the design in the real world. A pilot test was conducted to evaluate the usability and to collect user feedbacks; participants gave positive feedback to the design idea, but had opinions on improvements as well.

Based on the feedbacks and observations from the pilot test, few questions are proposed for further study on the design of interacting with change representations: position of the pop-up window, indication of the change parts on edit area, and the usage of revisions in real contexts.

It is expected that this thesis can contribute to better design and development of change representations in web-based collaborative systems.

References

[Berners-Lee and Fischetti, 1997] Berners-Lee, T. and Fischetti, M. *Weaving the Web*. Harper San Francisco, 1997.

[Cross, 1990] Cross, G. A. A Bkhtinian exploration of factors affecting the collaborative writing of an executive letter of an annual report. In *Research in the Teaching of English* **24** (2), 1990, 173–203.

[DeSanctis and Gallupe, 1897] Desanctis, G. and Gallupe, R. B. A foundation for the study of group decision support systems. *Manage. Sci*. **33** (5), 1987, 589–609.

[Désilets *et al.*, 2005] Désilets, A., Paquet, S., and Vinson, N.G. Are Wikis Usable? In *The 2005 International Symposium on Wikis*. October 17-18, 2005. San Diego, California, USA. NRC 48272.

[Dieberger and Guzdial, 2002] Dieberger, A. and Guzdial, M. CoWeb – Experiences with Collaborative Web Spaces. In *From Usenet to CoWebs: Interacting with Social Information Spaces*. Springer-Verlag, 2002.

[Ebersbach *et al.*, 2008] Ebersbach, A., Glaser, M., and Heigl, R. *Wiki Web Collaboration*. Springer, 2008.

[Ede and Lunsford, 1990] Ede, L. and Lunsford, A. Singular Texts/Plural Authors: Perspectives on Collaborative Writing. *Southern Illinois University Press*, 1990.

[Fish *et al.*, 1988] Fish, R.S., Kraut, R.E., Leland, M.D.P., and Cohen, M. Quilt: a Collaborative Tool for Cooperative Writing. In *Proceedings of COIS'88*, 1987, 30–37.

[Grudin, 1994] Grudin, J. Groupware and Social Dynamics: Eight Challenges for Developers. *Communications of the ACM* **37** (1), 1994, 92–105.

[Hawley, 2003] Hawley, A. *A Manual to the GNU Revision Control System (RCS)*: https://agave.garden.org/~aaronh/rcs/manual/html/

[Halliday and Hasan, 1976] Halliday, M.A.K. and Hasan, R. *Cohesion in English*. London: Longmans, 1976.

[Hunt *et al.*, 1975] Hunt, J. W., and McIlroy, M.D. An Algorithm for Differential File Comparison, Bell Laboratories, N.J., Computing Science Technical Report

41, 1975.

[Kim and Eklundh, 1998] Kim, E. and K. Severinson Eklundh. How Academics Co-ordinate their Documentation Work and Communicate about Reviewing in Collaborative Writing. Interaction and Presentation Laboratory, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm University. Technical Report TRITA-NA-P9815, NADA, August 1998.

[Kim and Eklundh, 2000] Kim, E. and K. Severinson Eklundh. Change Representations in Collaborative Writing. Interaction and Presentation Laboratory, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm University. Technical Report TRITA-NA-P0005, NADA, March 2000.

[Kim and Eklundh, 2001] Kim, H. C. and Severinson Eklundh, K. Reviewing practices in collaborative writing. *Computer Supported Cooperative Work: The Journal of Collaborative Computing* **10** (2), 247–259.

[Kim and Eklundh, 2002] Kim, H. and Eklundh, K. Collaboration between Writer and Reviewer through Change Representation Tools. *In Proceedings of the 35th Annual Hawaii international Conference on System Sciences* (Hicss'02)-Volume **1** - Volume **1** (January 07 - 10, 2002). HICSS. IEEE Computer Society, Washington, DC, 39.

[Leland *et al.*, 1988] Leland, M. D. P., Fish, R.S. and Kraut, R.E. (1988). Collaborative document preparation using Quilt. In *Proceedings of the Conference on Computer-Supported Cooperative Work*, 1988, 206–215. Portland, Oregon (September).

[Leuf and Cunningham, 2001] Leuf, B. and W. Cunningham. *The Wiki Way -- Quick Collaboration on the Web*. Boston, MA, Addison-Wesley, 2001.

[Malcolm and Gaines, 1991] Malcolm, N. and Gaines, B. R. A minimalist approach to the development of a word processor supporting group writing activities. *SIGOIS Bull*. **12**, 1991, 2–3, 147–152. DOI= http://doi.acm.org/10.1145/127769.122846

[Nachbar, 1988] Nachbar, D. Spiff – A Program for Making Controlled Approximate Comparisons of Files. In *Proceedigns of the Summer 1988 USENIX Conference*, 1988, 73–84.

[Newman and Newman, 1993] Newman, R. and Newman, J. Social writing: Premises and Practices in computerized contexts. In Sharples, M. (Ed.) *Computer Supported Collaborative Writing*. London: Springer-Verlag, 1993, 29–40.

[Neuwirth *et al.*, 1990] Neuwirth, C. M., Kaufer, D. S., Chandhok, R., and Morris, J. H. Issues in the design of computer support for co-authoring and commenting. In *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work* (Los Angeles, California, United States, October 07 - 10, 1990). CSCW '90. ACM, New York, NY, 183–195. DOI= http://doi.acm.org/10.1145/99332.99354

[Neuwirth *et al.*, 1992] Neuwirth, C. M., Chandhok, R., Kaufer, D. S., Erion, P., Morris, J. and Miller, D. Flexible Diff-ing in a Collaborative Writing System. In *Proceedings of CSCW '92*, 1992, 147–154.

[Neuwirth *et al.*, 1994] Neuwirth, C. M., Kaufer, D. S., Chandhok, R., and Morris, J. Computer Support for Distributed Collaborative Writing: Defining Parameters of Interaction. In *Proceedings of CSCW '94*, 1994, 145–152.

[Noël and Robert, 2004] Noël, S. and Robert, Jean-Marc. Empirical Study on Collaborative Writing: What Do Co-authors Do, Use, and Like? In *Computer Supported Cooperative Work*. **13**, 2004, 63–89.

[Paul, 2006] Paul, L. C. Wikipedia Usability Presentation. In WikiMania Hacking Days, August, 2006.

[Posner and Baecker, 1992] Posner, I.R., & Baecker, R.M. How people write togethr. In *Proceedings of the 25th Hawaii International Conference on System Sciences*, Vol. **4**, Hawaii, 1992.

[Reitmayr, 2006] Reitmayr, E. Usability Test Results: Editing Information in the German Wikipedia. In *http://www.openusability.org*, March, 2006.

[Samuels and Kamil, 1984] Samuels, S.J,, and Kamil, M.L. Models of the reading process. In *Handbook of Reading Research*, D. P. Pearson, Ed. Longman Inc., N. Y., 1984,
pp. 185-224.

[Miles *et al.*, 1993] Miles, V.C., J.C. McCarthy, A.J. Dix, M.D. Harrison and A.F. Monk. Reviewing Designs for a Synchronous-Asynchronous Group Editing Environment. In M. Sharples (ed.) *Computer Supported Collaborative Writing*. London: Springer-Verlag, 137–160.

[Wei *et al.*, 2005] Wei, C., Maust, B., Barrick, J., Cuddihy, E., and Spyridaki, J. H. Wikis for Supporting Distributed Collaborative Writing. In *Proceedings of the Society for Technical Communication 52nd Annual Conference*, Seattle, 2005.

Diff on Wikipedia: http://en.wikipedia.org/wiki/Diff

ECMAScirpt Specification: http://www.ecma-international.org/publications/standards/Ecma-262.htm

Help Manual of Adobe Framemaker: http://help.adobe.com/en_US/FrameMaker/8.0/help.html?content=Chap14-Revision-Mgmt_03.html

ISO Standards 9241-11: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16883

W3C DOM Specification: http://www.w3.org/DOM/

Wdiff Website: http://www.gnu.org/software/wdiff/