New Techniques for Merging Text Versions

Darius Dadgari and Wolfgang Stuerzlinger

Dept. of Computer Science & Engineering
York University, Toronto, Canada
`http://www.cse.yorku.ca/~wolfgang`

Abstract. Versioning helps users to keep track of different sets of edits on a document. Version merging methods enable users to determine which parts of which version they wish to include in the next or final version. We explored several existing and two new methods (highlighting and overlay) in single and multiple window settings. We present the results of our quantitative user studies, which show that the new highlighting and overlay techniques are preferred for version merging tasks. The results suggest that the most useful methods are those which clearly and easily present information that is likely important to the user, while simultaneously hiding less important information. Also, multi window version merging is preferred over single window merging.

Keywords: Versioning, version merging, document differentiation, text editing, text manipulation.

1 Introduction

Versioning allows users to compare two (or more) versions of a single document and to create a new version, which presumably synthesizes the "best" features of the old ones. This is a common activity when jointly writing an essay or conference paper, where each participant may modify an original document in parallel to improve it. Then is a necessary for the task-leader to compare and merge text from these multiple versions towards the common goal of creating a final version. We focus on (English) text documents and do not evaluate methods for source code, alphanumeric strings, highly structured poetry, or other forms of writing. Moreover, we ignore formatting, layout and other meta-data.

However, versioning and version merging is not used universally today, due to the perceived difficulty to use, or the users' lack of knowledge of the tools. Even though text versioning systems have been available for more than a decade, there is no published comparative examination of version merging methods. Moreover, multiple window versioning is rarely used, too. Most common version merging methods use a single window and show information as necessary within that single window. Multiple window version merging uses two or more separate windows, where each uses some (potentially different) visualization method to differentiate between versions. This reduces visual clutter.

The primary objective of this paper is to explore a variety of methods for text version differentiation and merging. More precisely, we investigate techniques that enable users to select the desired parts from among a set of different versions, i.e.

version merging. For this task, we present new techniques for both single window and multiple window versioning. Moreover, we validate these techniques with user studies in comparison to other published and commercially available techniques.

We examine whether version merging with multiple windows is superior to single-window when examining many documents, and present new interaction methods unique to multiple window text version merging.

2 Related Work

A fundamental operation in text editing is to compare two versions and to output which and where data has been added, changed, and deleted. Classical command line tools, such as `diff` [7], compute and output this list directly, thus providing a method to determine differences between documents. The output format of `diff` is very terse. This does not allow end-users to easily see how those changes relate to the context within the document and we assume that this is the main reason why non-technical people rarely use this tool. Providing the same information with overview exposes versioning to a much larger audience. `xdiff` [15] and similar graphical differencing utilities, show versions side-by-side in a scrollable display and by using colored backgrounds to highlight changes. Such tools target highly structured text, such as source code, and support often only the direct merging of code changes.

Microsoft Word 2000, and similarly WordPerfect 12, provides a Track Changes option, which denotes insertions by underlining added text and visualizes deletions by applying strikethrough to the deleted text. In these applications, versioning is often referred to as "Revisions" or "Track Changes". Insertions and deletions are colored based on which user made the change. This method is known as 'redlining' or 'strikethrough', and is called in this paper '*Strikeout/Underline*'.

Word 2003 utilizes balloons containing the deleted text as opposed to showing strikethrough text. This is called the *Balloon* method in this paper, which can lead to an overwhelming number of annotations. Each individual portion of changed text may be accepted or rejected relative to the previous version individually, rather than all at once. Also, a reviewing pane is provided which shows all changes in a linear top-to-bottom order. But this pane uses a lot of space even for minor changes. Finally, the variety of available options for revising can be confusing to end users and has led to the accidental public release of sensitive document data on multiple occasions. Third party plug-in software, such as iRedline [8], aim to fix some of these issues.

Word 2007 adds a one-click methods to view only comments and other formatting in balloons inline within the document. Another significant addition is the Compare Documents feature that, similar WinMerge [14], takes two documents and visualizes their similarities and differences. One issue is that four separate frames with different views of the information are shown at once, together with the title bar and other menus, which makes each pane relatively small.

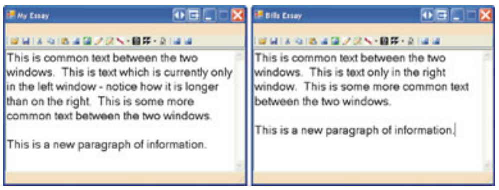
Research on collaborative writing is also related to our efforts. Several studies investigated collaborative writing, with a wide variety of results. Some found that users rarely collaborate and only 24% of users write together [1]. Yet, in other studies, 87% of all documents were produced by at least two authors [3]. In cases where collaborative writing is used, one study found that the favorite function of users was the one that allowed them to scan through the revisions made by others [12]. Users prefer version control functions as well. We speculate that such findings apply also to recently introduced collaborative writing systems, such as Google Documents.

Although source code versioning is outside of the focus of this paper, the topic is still related to general text versioning. We point the reader to the "Compare with Local History" feature of Eclipse, WinMerge [14] in combination with xdocdiff [16], the Concurrent Version System (CVS), and Subversion (SVN).

In summary, and while there are several existing implementations of text versioning methods, their usability has not been formally evaluated.

3 Version Differentiation Techniques

First, we discuss existing and two new version differentiation methods, all targeted at version merging. Later, we will evaluate these techniques with user studies. As discussed above, the most popular current document version merging techniques focus on showing all modifications to the text in a single window, which can negate the potential usefulness of multiple window techniques.



This is common text between the two windows. This is text which is currently only in the left window - notice how it is longer than on the right. This is some more common text between the two windows.

This is a new paragraph of information.

This is common text between the two windows. This is text only in the right window. This is some more common text between the two windows.

This is a new paragraph of information.

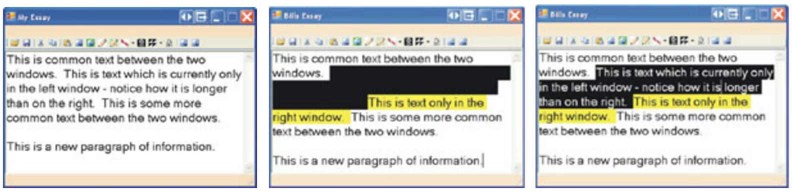
Fig. 1. Base condition: side-by-side view of two different versions of a text document

This paper examines multiple version differentiation methods that enable users to identify and compare multiple versions of a text document on-screen at the same time. Users then can select which portions of text they wish to retain in the result. This last part is most frequently implemented via appropriate GUI widgets. The cycle repeats until completion of the document. Here we describe the version differentiations methods we investigated. The base condition for our work is a side-by-side view of two versions. See Figure 1.

3.1 Highlighting Method

To differentiate text modified relative to a previous version, we present a new *Highlighting* method, which uses highlighting for changes. This focuses attention to some objects and/or diverts from others [6] and provides a simple visualization of what has and what has not been modified relative to a previous version. Considering the ubiquitous use of highlighting and highlighter markers in everyday life, this is an intuitive affordance. Our implementation highlights inserted text with a (user configurable) soft-palette background color, by default light yellow. Assuming black text on a white background, removed text is shown with a black background, which hides deletions from the user. This highlight method focuses the user on items that likely require user attention, while diverting attention from those parts that may not require consideration. Although user attention is diverted from the deleted text, users can still easily see that text. For that, he or she can hover the cursor over the hidden text, which will turn the text color within the deleted text section white, while keeping the background color black. The user can lock or unlock this state by left-clicking on the text. This provides a simple way to view deleted text, without the added clutter of viewing all deleted text when it is not necessary. See Figure 2.

Highlighting is used to visualize differences in a variety of work, usually by highlighting important items to focus attention [10]. But, it is also recognized that too much color differentiation confuses users [17]. Thus, we believe that using too many colors for text highlighting are not positive, e.g., when the colors relate to several authors with edits that have to be reconciled. The *Highlighting* differentiation method is similar to the methods used in systems such as WinMerge, save for the addition of whitespace and the inability to hide information not in both documents. Google Documents also recently added highlighting in their revision feature.



This is common text between the two windows. This is text which is currently only in the left window - notice how it is longer than on the right. This is some more common text between the two windows.

This is a new paragraph of information.

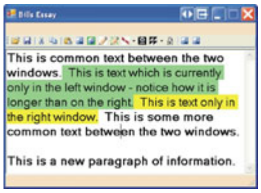
This is common text between the two windows. This is text only in the right window. This is some more common text between the two windows.

This is a new paragraph of information.

This is common text between the two windows. This is text which is currently only in the left window - notice how it is longer than on the right. This is text only in the right window. This is some more common text between the two windows.

This is a new paragraph of information.

Fig. 2. Highlighting Method. Left – Original version. Middle – changed version with blackened text. Right – same version during mouse hover.



This is common text between the two windows. This is text which is currently only in the left window - notice how it is longer than on the right. This is text only in the right window. This is some more common text between the two windows.

This is a new paragraph of information.

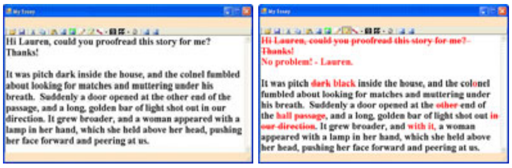
Fig. 3. The two windows of the Base Conditions overlaid with the Overlay method

3.2 Overlay Method

Transparency or translucency enables users to understand directly how two documents correlate by layering them, similar to using transparencies to overlay information onto a page of text. Based on this idea, we present the new *Overlay* method, where users can make windows transparent via alpha blending [13]. Users can then drag one window on top of another, which snaps the top window to the bottom window and locks it in place. In this state, the toolbar, menu bars and status bar of the top window are shown normally, but the text area background is made transparent, and the text position is synchronized to the uppermost paragraph in the underlying window. Text that is identical in both versions is shown shadowed and bold with a white background. Text that is only included in the top layer is highlighted in light yellow and text only in the bottom layer in light green. In order to prevent the text from becoming garbled due to different spacing, appropriate white space is automatically added. This maintains full visibility of the data. See Figure 3. Dragging the top window off the bottom one restores the normal appearance of both windows. This process gives the user the full illusion of a translucency effect. We note that this process also bypasses the inherent difficulties in reading blended text, which can severely affect readability [5].

3.3 Strikeout / Underline Method

The *Strikeout/Underline* method, also known as 'redlining', mimics the Word Track Changes functionality and is similar to the *Highlighting* method. However, insertions are shown in red and are underlined, whereas deletions are drawn red and have a strikethrough effect applied. Different colors are used to differentiate between different users' modifications. Users can hover their mouse over the text for a balloon that displays the author, date, time and type of change, see Figure 4. This method was extensively used prior to Word 2003 and can scale to multi-document merging.



Hi Lauren, could you proofread this story for me?
Thanks!

It was pitch dark inside the house, and the colnel fumbled about looking for matches and muttering under his breath. Suddenly a door opened at the other end of the passage, and a long, golden bar of light shot out in our direction. It grew broader, and a woman appeared with a lamp in her hand, which she held above her head, pushing her face forward and peering at us.

~~Hi Lauren, could you proofread this story for me?~~
~~Thanks!~~
No problem! - Lauren.

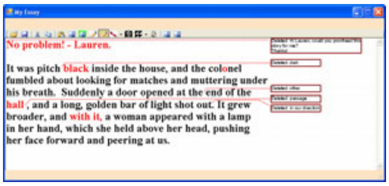
It was pitch ~~dark~~ black inside the house, and the colonel fumbled about looking for matches and muttering under his breath. Suddenly a door opened at the ~~other~~ end of the hall passage, and a long, golden bar of light shot out in ~~our direction~~. It grew broader, and with it, a woman appeared with a lamp in her hand, which she held above her head, pushing her face forward and peering at us.

Fig. 4. Strikeout/Underline Method. Left – Text prior to edits. Right – Text after edits.

A drawback of this method is that deleted text is shown with the same text size as unmodified or inserted text. As the deletion itself suggests that the information will likely not be as important for the next revision this is counterintuitive. In addition, the strikethrough effect negatively impacts readability. Another drawback is that the color of deleted text is the same as for inserted text, which also impacts readability. There are numerous commercial implementations that enhance this basic method [2][9].

3.4 Balloon Method

This method mimics the Word 2003 Track Changes functionality. This is one of the most commonly used text version merging systems today due to the high usage rate of Microsoft Office. In this method, insertions are marked in red. Deletions are shown in the right margin in separate, fixed-position balloons that contain the deleted text. A dotted line and arrow point to the location the text was deleted from. Users can hover their mouse over the text or link to show a balloon that displays more information about the text change, thus allowing users to retrieve additional information as required, see Figure 5.



No problem! - Lauren.

It was pitch black inside the house, and the colonel fumbled about looking for matches and muttering under his breath. Suddenly a door opened at the end of the hall , and a long, golden bar of light shot out. It grew broader, and with it, a woman appeared with a lamp in her hand, which she held above her head, pushing her face forward and peering at us.

Fig. 5. Balloon Method. Same original text as in Figure 4, story after edits.

This method provides a way for users to quickly view deleted information while at the same time removing it from the main text view. It also shows where insertions occur and permits for extra information to be shown as necessary. The primary drawback is that the balloons require a significant amount of screen space. In a 1024x768 window a Word document in Print Layout at 100% zoom uses ~21% of the horizontal width (218 pixels) solely for the balloons. Another drawback is that the balloons always show the complete deleted text. Moreover, the arrows are fairly thin, and it is not always easy to see the original context of a deletion. These lines are also always visible, which creates more visual clutter. Word 2007 gives users the option to turn off the dotted lines, and they appear only if the user selects the balloon.

4 Evaluation

The goal of this study was perform an initial exploration of the topic and to compare the discussed version differentiation methods in terms of performance. We performed a 2x5 within subjects experiment to examine the ease-of-use and efficiency of the version differentiation methods. The intent was to determine which of the five compared methods were preferred. Our independent variables were window type (single and multiple window) and method used (the base condition, *Balloon*, *Strikeout/Underline,* and the new *Highlighting* and *Overlay* techniques). In the base condition we provided users with a physical copy of the older document. Our two dependant variables were error rate and task completion time. The order of the conditions and essays was counterbalanced.

In the single window conditions, one text window was shown on-screen containing a single document with annotations relating to a modification of the first text document. The size of the text workspace window was maximized to the screen. In the multiple window conditions, two equal-sized text windows filling the display were shown on the monitor. The left document window contained the original text document and the right document window contained the modified version of the left essay annotated with the given versioning method. These modifications directly related to the annotations shown in the single window condition. The sizes of the text workspace windows were identical. Furthermore, in both the single and multiple window conditions, an additional, initially minimized, target text workspace window was provided, into which participants were asked to paste information as requested.

Participants were given the following scenario: A student had written an essay a week before their assignment was due. However, the student forgot to save that essay and as such wrote it again. But, a computer-savvy friend was able to retrieve the previously lost essay, so the student now has two similar essays that he or she can potentially hand in. Since the essay format and structure is very similar, the student can pick which portions of the essays to keep and which to throw away. Students were asked to analyze the essays paragraph-by-paragraph and copy paragraphs into the target window based on the criteria that the paragraph chosen must have no spelling errors in it and be the longer paragraph. If a paragraph with a spelling error was selected, this selection was marked as an error. If an entirely new paragraph was seen in an essay version that was not similar to the other version then participants were asked to copy that paragraph, and not doing so was also rated an error. Participants were told that other grammatical or style issues should be ignored. The length requirement is analogous to determining which part of the text is desirable and the spell-checking requirement is analogous to determining whether the specific information within a text part is the correct one. Note that copying and pasting is not representative of version merging in current text editing systems, which often have “Accept/Reject Change” functionality. However, such functionality would be in conflict with our example usage scenario that involves two different documents, which our participants found quite compelling, as almost none of them had experience with versioning. Hence and as the number of paragraphs was fixed, we decided to go with copy/paste, since the overhead for this is likely constant.

All ten used essays were approximately 1000 words in length and between 12 to 15 paragraphs for each ‘old’ and ‘new’ essay. Automatic spellchecking capabilities were not made available. Hence, participants were forced to read through all versions during the test to ensure they had no spelling errors. On purpose, the spelling errors were more obvious than commonly found in draft texts, to address the potential confound of potentially different spellchecking abilities. This ensured that participants used the investigated techniques as fully as possible.

20 participants, six female, aged 18-31, were recruited from the local University campus. All had English as their first language. Participants were paid and took 20 to 30 minutes to complete the tasks. Before the main study began, users were asked to spend five minutes to practice common operations on a test document, such as deleting lines, adding text, cutting, and pasting paragraphs. Finally, participants were given a questionnaire at the end.

4.1 Results for Single-Window Methods

A repeated measures ANOVA identified a significant main effect for task completion time and error rate overall. To simplify the explanation, we discuss the single-window method separately from the multiple-window case in the following and present only significant results of the corresponding ANOVA.

We initially examine the dependant variables of task completion time and error rate for the single window methods. Figure 6 left shows the task completion time in seconds and standard deviation for the techniques in the single-window method. The difference between the five techniques was significant ($F_{4,199}$=0.7121, $p<0.01$). A Tukey-Kramer test reveals that the base condition is significantly worse than all other conditions (pair wise p<0.01, except for *Balloon* $p<0.05$). *Balloon* is also significantly different from all others ($p<0.01$).

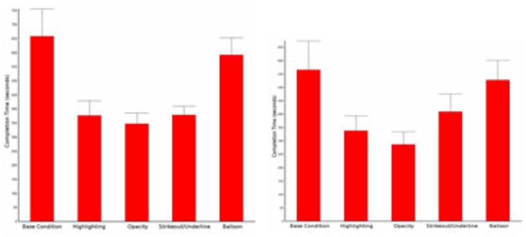


Fig. 6. Average completion time in seconds and standard deviation for Single Window (left) and Multiple Window (right) version merging methods

On a 7-point Likert scale with 7 being the best rating and 1 the poorest, the median user preference for the Base Condition was 2.0, for *Highlighting* 5.0, for *Overlay* 5.0, for *Strikeout/Underline* 5.0, and for *Balloon* 3.0.

The mean number of errors of the techniques was significantly different ($F_{4,199}$=0.6293, *$p<0.01$*), where an error is either a missed paragraph or a paragraph with a spelling error. The mean number of errors for the Base Condition was 3.25, for *Highlighting* 1.7, for *Overlay* 1.7, for *Strikeout/Underline* 1.8, and for *Balloon* 2.7. According to a Tukey-Kramer test, the difference between the base condition and *Highlighting, Overlay,* and *Strikeout/Underline* was significant. Also, the difference between *Balloon* and *Highlighting, Overlay,* and *Strikeout/Underline* was significant. Finally, the difference between *Strikeout/Underline* and *Overlay* was significant.

4.2 Results of Multiple-Window Methods

The right part of Figure 6 illustrates the task completion time in seconds and standard deviation for the multiple window methods. The difference between the techniques in terms of completion time was significant ($F_{4,199}$ =0.8715, *$p<0.01$*). A Tukey-Kramer test reveals that the base condition and *Highlighting, Overlay,* and *Strikeout/Underline* was significant (all $p<0.01$). Also, the difference between *Balloon* and *Highlighting, Overlay,* and *Strikeout/Underline* was significant (all $p<0.01$, except *Strikeout/Underline $p<0.05$*).

On a 7-point Likert scale with 7 being the best rating and 1 the poorest, the median user preference for the Base Condition was 3.0, for *Highlighting* 6.0, for *Overlay* 4.0, for *Strikeout/Underline* 5.0, and for *Balloon* 4.0.

The difference between the mean number of errors between techniques was significant ($F_{4,199}$=1.19, $p<0.05$), where an error is either a missed paragraph or a paragraph with a spelling error. The mean number of errors for the Base Condition was 2.5, for *Highlighting* 1.5, for *Overlay* 1.5, for *Strikeout/Underline* 1.6, and for *Balloon* 2.05. As per a Tukey-Kramer test, the difference between the base condition and *Highlighting*, *Overlay,* and *Strikeout/Underline* was significant.

4.3 Discussion

Overall, the *Highlighting*, *Overlay*, and *Strikeout/Underline* method performed best, with all three techniques performing significantly better than the others. On average, *Overlay* performed best, but not significantly so. The results also show that both the base condition as well as the *Balloon* method seem to be less well suited for version merging. For the *Balloon* method, we attribute this to the fact that deletions are further away from the original document, which makes *using* deleted text difficult. Also, inserting deleted text moves other text around and confuses users. The potential strength of the *Balloon* method, the ability to handle changes by many authors, was not evaluated here.

Several participants stated in the questionnaires and/or during the study that the *Highlighting* method was preferred to *Strikeout/Underline*, primarily due to the ability to hide information that was not relevant to the task. *Strikeout/Underline* also yielded annoyed comments from users regarding readability. More precisely, they found reading deleted text difficult and made comments about eyestrain here.

Comparing across the single and multiple window conditions, we find that multiple windows decrease the task completion time significantly for most methods, except for *Strikeout/Underline*. The number of errors was also reduced significantly for the base and *Balloon* condition in the multiple windows case. And the multiple window conditions are 9 to 20% faster in terms of completion time. These findings are further corroborated by the questionnaire results and comments by participants, which show a marked preference for the multiple window condition. Note that the single window condition showed proportionally more text of the document, whereas the multiple window condition showed less text per window. Even though it puts multiple windows at a relative disadvantage, the ability to view multiple annotated versions simultaneously still seems preferable to a single window. As multiple windows are almost always faster, we believe that it is overall the better choice.

Overall, we find that methods that hide likely irrelevant information (while still providing quick access on demand) and that require less user interaction are preferred and show better results. *Highlighting* is a method that exemplifies this design choice and it is likely the best choice for text comparison and version differentiation.

An interesting result side result of our work is that, with a growing number of text versions, the amount of annotations for single-window version merging becomes unwieldy. We found that expanding this information into multiple windows simplifies the version merging task.

5 Conclusion and Future Work

We examined three existing and two new methods for text versioning and merging. All are designed to help users differentiate between multiple versions of a document. All these version-merging methods were evaluated in a user study. Two different scenarios were examined: single window and multiple windows. We found that the new *Highlighting* and *Overlay* methods provided the largest benefits in completion time, error rate, and had the strongest user preference. We also found that multiple window version merging seems significantly easier for users compared to single-window version merging as the number of text versions increases. In further, currently unpublished, work on version merging on multiple monitors we also found that *Highlighting* is also the fastest and most preferred method.

For future work, we plan to address the issue that deleted sections of text in the *Highlighting* method are always visible as colored “blocks”. As such, we propose further research into a modified *Highlighting* method in which deleted text is shown within the document as a unique, small, symbol, which can be toggled to the "normal" *Highlighting* view of the deleted text via a mouse click and back. We also plan to investigate using multiple symbols in a row to visualize the approximate length of the deletion. Also, we plan to investigate the issue of change granularity [11].

References

[1] Couture, B., Rymer, J.: Discourse interaction between writer and supervisor: A primary collaboration in workplace writing. Collaborative Writing in Industry: Investigations in Theory and Practice, pp. 87–108 (1991)
[2] DeltaView Workshare, `http://www.workshare.com/products/wsdeltaview`
[3] Ede, L., Lunsford, A.: Singular Texts/Plural Authors: Perspectives on Collaborative Writing. Southern Illinois University Press
[4] Grundy, J., Hosking, J., Huh, J., Li, K.: Marama: an Eclipse Meta-toolset for Generating Multi-view Environments. In: Proc. 30th Conference on Software Engineering (2008)
[5] Harrison, B., Kurtenbach, G., Vincente, K.: An Experimental Evaluation of Transparent User Interface Tools and Information Content. In: Proc. UIST 1995, pp. 81–90 (1995)
[6] Horton, W.: Overcoming Chromophobia: A Guide to the Confident and Appropriate use of Color. IEEE Transactions on Professional Communication 34(3), 160–171 (1991)
[7] Hunt, J.W., McIlroy, M.D.: An Algorithm for Differential File Comparison. Computing Science Technical Report #41, Bell Laboratories (1975)
[8] iRedline: Enhanced Word Document Comparison, `http://esqinc.com/section/products/3/iredline.html`
[9] Litera Change-Pro, `http://www.litera.com/products/change-pro.html`
[10] Manber, U.: The Use of Customized Emphasis in Text Visualization. In: Proceedings of the IEEE Conference on Information Visualization
[11] Neuwirth, C., Chandhok, R., Kaufer, D., Erion, P., Morris, J., Miller, D.: Flexible Diffing in a Collaborative Writing System. In: CSCW 1992, pp. 147–154 (1992)
[12] Noël, S., Robert, J.-M.: Empirical Study on Collaborative Writing: What do co-authors do, use and like? In: Proc. Computer Supported Cooperative Work, pp. 63–89 (2004)
[13] Porter, T., Duff, T.: Composting Digital Images. Computer Graphics 18(3), 253–259 (1984)
[14] WinMerge differencing and merging tool, `http://winmerge.sourceforge.net`
[15] xdiff: X-Windows file comparator and merge tool, `http://reality.sgiweb.org/rudy/xdiff`
[16] xdocdiff plugin for WinMerge, `http://freemind.s57.xrea.com/xdocdiffPlugin/en/index.html`
[17] Yang, W.: How to merge program texts. Journal of Systems and Software 27(2) (1994)