

Designing and Evaluating a Web-Based Collaboration Application: A Case Study

Wenli Zhu

Microsoft Corporation, One Microsoft Way, Redmond, WA 98052 USA

ABSTRACT

The Web has evolved from a simple browsing environment to an interactive, goal-oriented, task-driven application and script-enabling platform. More and more Web applications are being developed to allow users to create, edit and manipulate objects in the browser, just like in traditional graphical user interface-based applications. SharePoint™ Team Services from Microsoft® is an example of such a Web application. It is a Web site designed to help a team to collaborate and share information. But it also has commands and dialog-like pages that allow users to modify the content of the Web site in the browser. This paper describes the design and evaluation process of the application and presents results and design solutions in the areas of: Navigation, browsing vs. editing, providing context and feedback, selection model, processes and interacting with other applications. Our findings confirmed and expanded previously identified issues with Web application design, and provided fresh data and design solutions to address these problems.

1. INTRODUCTION

In recent years, more and more Web-based applications have been developed to offer functionalities that go beyond simple browsing. As indicated by Fellenz, Parkkinen & Shubin (1999), there may be several different categories of Web usage: browsing, performing transactions and running applications. Web applications are usually developed out of a desire to reduce the cost of software development and distribution, especially for applications that need to run across multiple hardware and software platforms. As explained by Rice and his colleagues (1996), Web applications allow users to create, edit and manipulate objects in the browser just like in traditional graphical user interface-based (GUI) applications.

While general Web page and Web site design guidelines (such as those identified by Nielsen, 2000) and traditional GUI design guidelines (for example, Shneiderman, 1987) can be applied to Web applications, unique characteristics of Web applications have been suggested and observed (Fellenz et al., 1999; Shubin & Meehan, 1997; Rice et al., 1996), including:

- Navigation: As Shubin & Meehan (1997) pointed out, Web browsers use the page metaphor which is appropriate for browsing Web pages with static text and hyperlinks. The navigation model is simple. Users follow hyperlinks on the page to view more information or use the Back button to retrace steps. With Web applications, however, this navigation model may create problems. When users' main goal is to complete a task in Web applications, links that take people away from the task may be distractive and may result in users taking a detour or losing their work. The Back button could also be problematic. Because the Web browser "caches previously seen pages and allows users to revisit these pages without communicating with the server," as Rice et al. (1996) put it, the browser is essentially allowing users to "travel back in time to an earlier interaction state and attempt to execute the commands as they were presented then." This may result in users executing the same command more than once (ordering more products than intended, for example) or misperceive the Back button as "undo." Some Web applications attempt to solve this problem by opening up a new browser window without the browser controls. However, users may not understand the new window's dependency on the browser and may intentionally kill the browser and thereby inadvertently kill the application.
- Browse vs. edit: Web applications need to support both browsing and editing. Designers of Web applications need to weigh the balance of the two. In the applications that Rice and his colleagues developed (1996), the editing environment was made to look as much like the browsing environment as possible, because they wanted to "minimize the number of different-looking pages to which the user would be exposed." In the end, however, they concluded that it might have been better "not to use the edit-in-place model but rather to use pages just for editing."
- Lack of context or feedback: As Rice et al. (1996) pointed out, "there is a fundamental difference" between the type of interactions supported by the browser and the type of interactions supported by the traditional GUI. In traditional GUI, users select an operand or operands (for example, a paragraph of text) through direct manipulation and then apply an operator by means of a menu selection or a keyboard command. The operand remains visible during the operation and provides context. After the operation, the operand changes and therefore provides feedback. In Web applications, however, the page that the user is on could be an

implicit operand and the links the user chooses will operate on the page. During and after the operation, users may be on different pages and therefore may lose the context of the operation or may not be able to see the feedback.

- Limited interaction model: Compared to traditional GUI applications, the interaction models in Web applications are much more limited. For example, it is hard to implement selection of a single object, menu bars, graying-out options, direct manipulation on objects, right-mouse menus, etc., in a Web application.
- Response time: Slow network connections or limited bandwidth may cause delays in the system response time in Web applications. Navigating from one page to another in a Web application may take longer than moving from a main window to a dialog box in a traditional application.
- Exit: Many Web applications do not have a way to stop “running” without closing the browser. Shubin & Meehan (1997) suggested having a “clearly marked exit” in a Web application to help users remember to finish what they are doing before leaving the application. Interestingly, Rice et al. (1996) confirmed the perceived importance of having a clear exit. They discovered that many of their Web application users had a strong desire to be able to log out of their system. Logging out was not necessary. But users felt a need to log out, even when they were told explicitly that it was not necessary.

2. DESIGN RATIONALE

SharePoint Team Services from Microsoft is a Web-based collaboration application designed to help a team to share information. During the design phase, several assumptions and decisions were made (Van Tilburg, 2000):

- It is a Web site. It should look like a Web site. The underlying data structure of the Web site is a list structure, and there are three types of lists that users can use: Documents, discussions and lists. A list that holds documents is called a document library. A list that holds discussions is called a discussion board. A list that holds any other items is called a custom list. Users can create multiple document libraries, discussion boards and lists. Within each list, users can create multiple items. See Figure 1 for an overview of the site structure and Figure 2 for a snapshot of the Web site home page.

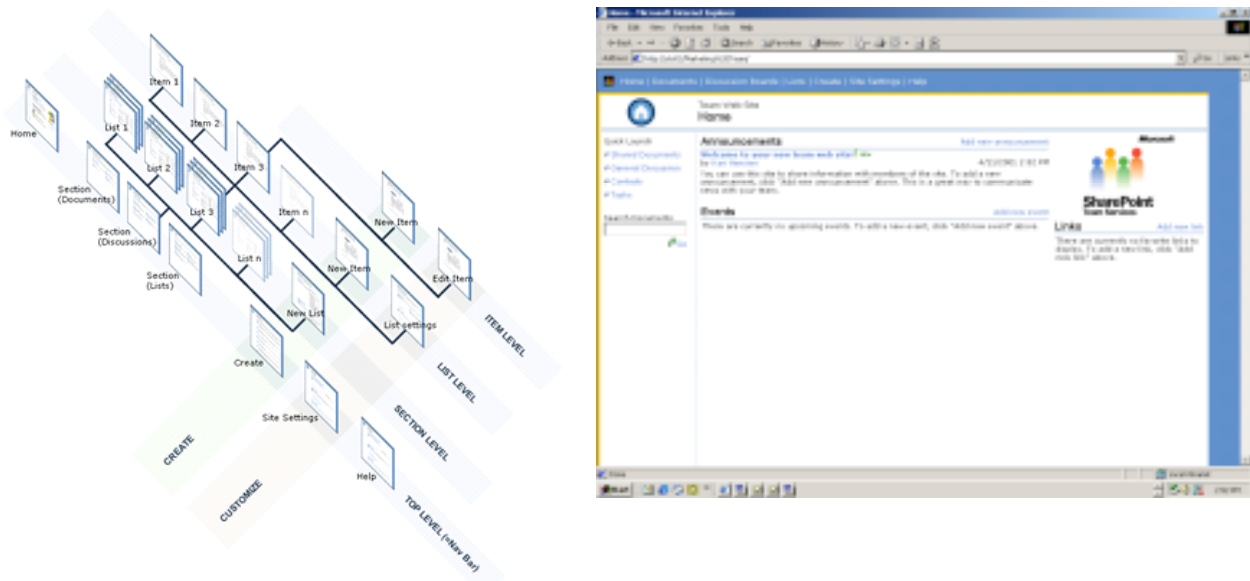


Figure 1. Site Structure (Adapted from Van Tilburg, 2000) Figure 2. Site Home Page

- The design of the Web site is based on assumptions about different user activities and the relative frequencies of these activities. First and foremost, it is a Web site, designed to help people collaborate and share information. It is therefore assumed that users will most often consume information, rather than add or edit information. The Web site is therefore optimized for the browsing behavior. Edit controls are more hidden and only become visible after a deliberate action from the user. Also, no selection model is implemented. To delete an item in a list, users have to choose to “edit” the item first and then delete it.
- There are four basic types of pages: The home page (see Figure 2), the list view page (the page that shows all the items in a list, see Figure 3), the item view page (the page that shows the details of a list item, see

Figure 4), the customization summary page (the page that shows all the options for a list) and the customization controls page (see Figure 5). The list view page and the item view page both have editing controls in a special toolbar area above the content area. The customization controls page looks like a traditional GUI dialog, but is not implemented as a pop-up dialog because we want to maintain a consistent “Web” feel of the application.

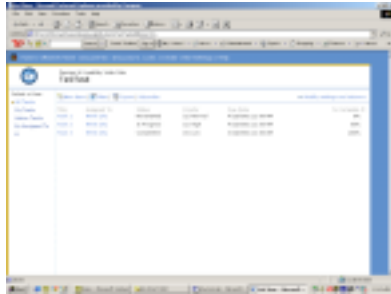


Figure 3. List View



Figure 4. Item View



Figure 5. Control Page

- Each page has 4 areas: Navigation bar, title area, left pane and content area. The Web site has one main navigation bar on the top which is always visible. It links to the different sections on the Web site: Documents, discussions, lists, create, site settings and help. On the list view page, the left pane contains links that filter the list (also called views) and on the home page, the left pane contains quick launch links.

3. EVALUATION

A series of usability studies was conducted to understand what potential usability problems users might have. The first study focused on lists, site customization and navigation. The second study focused on “getting started” or “new site creation” experience. The third study focused on document creation, uploading and editing. The fourth study collected quantitative performance measures. The evaluation process was a typical iterative design process, i.e., results from a usability study were used to make design changes and the new design was tested again. A core set of tasks such as those involving users interacting with lists and documents as well as navigation were repeated across all the studies. The following sections briefly review the methodology.

3.1 Subjects

A total of 42 subjects participated in the four usability studies mentioned above. All of them were intermediate to advanced Microsoft Office® users (user experience was classified using a questionnaire developed by Draine and McClintock, 1999). All the subjects were regular Internet users and had intranet in their organizations. All the subjects were working on projects with deadlines and deliverables. Half of the subjects were project leads and the other half were project participants. The profile of the subjects was chosen to represent the target user population of the application. Subjects received free Microsoft software for their participation.

3.2 Tasks

Tasks representative of real-world scenarios were given to the subjects. Most of the tasks were close-ended, i.e., there was one clear outcome to judge whether the task was completed successfully. Some tasks were open-ended, i.e., there was no clear end state that the subjects had to achieve. Tasks were chosen to cover a wide range of possible usage scenarios and feature areas. Repetitive tasks were included, within one usability study and across several usability studies, to compare performance.

3.3 Procedure

Each test session took about 1.5-2 hours. In each test, subjects filled out a background questionnaire and performed the tasks sequentially. Subjects self-reported completion of the tasks. They could also stop at any time.

3.4 Measures

Verbal protocol, success/failure rates, task completion time and errors were collected.

4. RESULTS AND DISCUSSION

Results and observations from the usability studies are summarized below.

4.1 What Worked Well

4.1.1 Separation of commands and content. Throughout the user interface, editing commands are distinguished from the content by: (a) the use of a verb in the command name; (b) the use of a different text color for the command name; (c) the use of icons; and (d) placement of commands in a designated area such as the special

toolbar on the list view and item view page with divider lines separating the commands from the content. Results showed that subjects can tell the difference between the commands and the content very well. When subjects needed to accomplish a task using one of the commands, they quickly recognized and used the command. The mean success rate was above 90%, and the mean task completion time was less than one minute. Longer and more specific labels and the combination of icons and labels helped users understand the commands better. For example, a change in the list customization command (from “change list settings” to “change the columns and views”) resulted in an increase in success rate (from ~40% to 100%). The results indicated that by separating the commands and the content successfully, the Web site separated editing and browsing successfully.

4.1.2 Navigation and the home page. The content area and the left navigation area got more attention than the top navigation bar. Subjects would look first in the content area and the left navigation area. They would also look there more often. An analysis of link-following patterns on the home page revealed that throughout a test session, 100% of subjects followed the links in the content area whereas 65% followed the links in the left navigation area (the Quick Launch links) and 50% followed the links on the top navigation bar. The results also revealed that subjects tended to use the home page as a “jump board,” especially at the beginning, to discover what is available and learn what they can do.

4.2 How Did We Do With Known Problems

4.2.1 The Back button. All the subjects frequently used the Back button. Sometimes, they backed out of an editing page or a customization controls page without confirming the changes or finishing the fields on the page. Most of the time, however, subjects were successful at using the dialog-like pages (for example, the customization controls page shown in Figure 5) to fill out the fields, scroll down to the bottom and click on the buttons. Some subjects used the Back button in the midst of filling out the fields to try to get information from other pages. That resulted in a loss of the data they already entered. It was very rare, however, that subjects used the Back button to get back to an editing page or a customization controls page and execute it again. Subjects mainly used editing commands to open up editing pages and customization controls pages.

4.2.2 Browse vs. edit. In general, subjects did not have a problem separating the two modes. However, some subjects at times tried to directly edit text (of a document or a list item) in the browser. This problem is unique in the Web environment because in traditional GUI applications, there is usually no separation between browsing and editing. For example, in a word processor, users can browse and edit at the same time. Another problem found with editing hints at how much users view the browser and the Web application as separate. In one of the usability studies conducted on an earlier design, subjects had to use the Edit command in the browser (on the browser menu and toolbar) to edit a document. Most of the subjects (90%) failed the task. They typically did not think of using a command in the browser to perform an action in the Web application. To solve this problem, we added a command called “Edit in <application name>” to help users edit a document.

4.2.3 Lack of context or feedback. The fact that some subjects used the Back button to back out of an editing page in order to see the context information on other pages indicated the need of maintaining the context of users’ work. Various problems associated with the lack of context or feedback were identified during the iterative design process and design changes were made to address the problems. For example, in earlier designs, users did not get appropriate feedback after adding a new item to a list or adding a new column to a list. Changes were made so that users would be brought to the appropriate pages after executing the editing commands and get feedback on their actions.

4.2.4 Limited interaction model. As mentioned before, no selection model was implemented because we wanted to have a simple design optimized for browsing. This decision made deletion more difficult. Many subjects tried to click or right-click to select a document, a column or a document library. They said that they would really like to be able to directly select an item instead of using the Edit command. They complained that the user interface did not follow the conventional GUI application behavior.

4.2.5 Response time and exit. No problem was found here.

4.3 New Problems Found

4.3.1 Processes and interacting with other applications. For some of the process-oriented or compound tasks, users have to use different pages to accomplish steps in a process or have to jump back and forth between the browser and other applications to finish a process. For example, with an earlier design, when creating a new document library with a custom template and custom columns, users needed to: (a) create a new document library; (b) create a template in an application such as Microsoft Word®; (c) upload the template; and (d) create columns for the document library. When subjects were asked to do this task in a usability study, all of them had difficulties with one step or another. A key problem was that subjects did not understand the steps they had to follow. We did not provide a mechanism (such as a wizard in the traditional GUI) to represent steps taken in a process and walk users through that process. Even if we did, it would have been difficult to lead users through the process of switching back and forth between the browser and another application. In the end, we changed the design so that users no longer needed to know to open an application to create a template. They are now presented with the option to “edit template” which automatically opens up an application for them. The more we can do to help users go through a process, the fewer problems users will have with process-oriented or compound tasks.

4.3.2 Refresh. When subjects used the Back button to go back to a list after customizing it, or when they edited a document in another application and switched back to the browser, they saw the old (cached) page that did not have the changes. None of the subjects understood why. Some thought they forgot to save the changes and tried to repeat the process. Relying on users understanding and remembering to use the Refresh command in the browser was not a good solution. In the end, we implemented automatic refresh in most cases but there were still scenarios where we could not fully solve the problem.

4.3.3 Hyperlink view switcher. As mentioned before, the list view page contains links that filter the list (called views) in the left pane (see Figure 3). For example, when users click on the link labeled “My Tasks,” the list will be filtered to show just the tasks “I” (the user) created. Throughout the usability studies, subjects had difficulties understanding how to use these hyperlink view switchers. They all noticed the links but could not understand how the views were defined, how to edit a view or that there was a default view. We do not exactly know why the hyperlink view switchers were so hard to understand, but there may be two possible reasons: views are hard to understand by nature and hyperlinks are not good UI mechanisms to implement views.

5. CONCLUSION

Our experience confirmed many of the problems reported and discussed in previous research. Data from our usability studies provided fresh evidence that these problems do exist and must be considered in building Web applications. Our data also suggested new issues that may further complicate the design of such applications. While using the Web as a platform to deliver applications is highly desirable and perhaps even inevitable, it seems the current Web Page metaphor and the application model of the browser will continue to pose unique challenges. New technology and new ways of interacting with the Web, however, will help us find solutions to these challenges.

ACKNOWLEDGMENTS

Martijn Van Tilburg and Mike Morton led the user interface design of SharePoint Team Services and contributed to the evaluation process. The author would like to thank Martijn in particular, for his insights and comments.

REFERENCES

- Draine, S. C., & McClintock, M. (1999). Developing a User-Knowledge Assessment Tool for Consumer Software. In the *Proceedings of the Eighth International Conference on Human-Computer Interaction*, Munich, Germany, pp.111-112.
- Fellenz, C., Parkkinen, J., & Shubin, H. (1999). Web Navigation: Resolving Conflicts between the Desktop and the Web. *SIGCHI Bulletin*, Vol. 31, pp. 26-28.
- Nielsen, J. (2000). *Designing Web Usability*. Indianapolis, IN: New Riders Publishing.
- Rice, J., Farquhar, A., Piernot, P., & Gruber, T. (1996). Using the Web Instead of a Window System. In the *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, Vancouver, BC, pp.103-110.
- Shneiderman, B. (1987). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. MA: Addison-Wesley Publishing.
- Shubin, H., & Meehan, M. M. (1997). Navigation in Web Applications. *Interactions*, Vol. 60, pp.13-17.
- Van Tilburg, M. (2000). SharePoint User Interface Guidelines. *Technical Report: Microsoft Corporation*. Redmond, WA.