12-2004

Requirement Elicitation of Large Web Projects.

David E. Frazier
East Tennessee State University

Requirement Elicitation of Large Web Projects

A thesis
presented to
the faculty of the Department of Computer Science
East Tennessee State University

In partial fulfillment
of the requirements for the degree
Master of Science in Computer Science

_____

by
David E. Frazier
December, 2004

_____

Martin Barrett, Chair
Bill Pine
Jeff Roach

Keywords: Web Design, Software Engineering

ABSTRACT

Requirement Elicitation of Large Web Projects

by

David E. Frazier

One of the most important aspects of developing a large Web-based project is getting the correct requirements from the client. Time and money can be lost if the requirements are incomplete or inaccurate. Traditional Web design sources tend to gloss over this important activity.

Software engineering is a mature field that can help in the quest for more complete and accurate requirement gathering. This paper explores the ways that traditional software engineering techniques can be applied to Web projects. A methodology is presented based on both existing and new techniques. Several experiments were conducted to determine the usefulness of each method in the methodology.

The conclusion points out that active participants in the Web development field do perceive there to be a problem in requirement gathering. Most who tested the proposed methodology found that it would be useful in addressing this problem.

CONTENTS

Page

CHAPTER 1

THE PROBLEM WITH WEB DEVELOPMENT

Web development has become a necessity for most businesses and organizations. People expect to find information they need online. Customers expect to be able to interact with businesses via a Web site. These interactions could range from purchasing a product from an e-commerce site, to obtaining customer support from an informational site, to getting pre-purchase product information from an advertising site. A Web presence is no longer optional. The importance of having an accessible, functional Web site can mean the difference between business success and failure. The problem then is, how to develop this kind of Web site.

During the first era of business Web site deployment, the stakes were much lower, both from a technical and a business standpoint. Technologically, the Web was a much simpler place. A site could provide text and graphics and little else. The concept of e-commerce had yet to be developed. All business Web sites were informational sites. A user could only obtain product information or learn about store locations. The Web was still little used by consumers, so a poor Web page would not have been a disaster to the business. In the best case, a company Web site would serve as a good advertising medium for the company.

Early Web developers were often pulled from non-technical or lightly technical professions. At this stage, Web development was much closer to desktop publishing than to programming, so many desktop publishers were the first Web developers. What separated good Web designers from bad was how proficient they were in developing graphics for their Web pages. The Web pages themselves were relatively easy to create in HTML. Larger companies would nearly always create these Web sites in-house, often using personnel from the public relations or marketing departments.

As the Web grew in popularity, is also grew in technical sophistication. The Web had moved beyond plain HTML, and it was soon able to link programming languages such as C++, Java, or Perl to allow Web sites to perform more tasks. It was now possible to sell products online with virtual shopping carts, as well as provide credit card processing and Web page access to the company database. Companies such as Amazon.com showed the business world how lucrative online business could be. Within a few short years, an e-commerce site became essential for all but the smallest businesses (Powell, 1998, p. 11).

However, there was a problem. Early Web developers were from mostly non-technical backgrounds. To be able to create an e-commerce site, one needed to be able to program. Many of these people had been able to elevate their positions in their companies significantly by becoming Web Masters. Whom were they going to be able to hire to get the Web programming done? One possibility, of course, was to hire traditional programmers to do the Web programming, but there were problems with this approach.

The Web presents a unique technical environment in which to program. Traditional programmers are often uncomfortable with the lack of control over user environments that is part of the Web. Also, the highly graphical nature of the Web led to some issues for programmers. Even programmers well versed in software engineering had a hard time using their traditional techniques on the Web. Traditional software projects are developed for use by the clients' employees. If an accounting program, for example, allows the client's employees to enter and process all accounting information accurately and efficiently, the product can be deemed a success. Traditional software engineering tools have been developed to aid in the creation of such projects. Web projects add a new dimension. These are programs that will be run by a client's *customers*. This adds a whole new set of requirements. The product has to be visually pleasing, as the customer is likely to judge the quality of a company by the quality of their web

site. The product has to be easy and intuitive to use. Customers cannot be trained as employees can. Users of a traditional system have to utilize it whether they like it or not. In most cases a customer, however, can just go to other Web sites if they are dissatisfied.

So who is going to create these new Web sites that have become critical to the organizations success? The solution in many cases is a team approach. Programmers will work with a graphic designer and HTML coders to effectively develop a useful Web site. Many businesses do not have this specialized knowledge in house, so they must turn, at least in part, to an outside company. Even worse, there is often a mixed design team of in-house employees and consultants who have vastly different backgrounds and do not normally work together.

To further complicate things, business managers have also been left behind by the fast moving technology of the Web. Whereas almost anyone could comprehend the relative simplicity of an informational-only Web site, it is the rare business leader who can comprehend the technical details of the full-blown e-business site. This lack of knowledge is not just a trivial matter. The business leader needs to know what options are available in order to have a Web site that best suits the business. Also, in instances when credit card information is being passed around, there are legal requirements as to how a businesses' Web site handles this sensitive information. How is a manager to know if his or her Web site is up to the task?

The real problem in Web development is one of communication. When the Web was simpler, a lone Web developer who worked in-house was able to keep track of the one Web site that he/she managed. Since he/she was often an existing employee of the company, he/she likely possessed some domain knowledge. In the current Web environment, the manager must be able to communicate to both the in-house and outside Web teams. HTML coders need to be able to communicate with programmers. Programmers need to be able to glean business needs from business managers. HTML coders, who used to do the whole job, struggle with the lack of

control and the necessity of coordinating efforts with others. Programmers often feel constrained by the Web environment, where the user cannot be precisely determined. Programmers are also unfamiliar with the need to provide a more open user environment. There is also a mindset among many companies that Web projects are in some ways less formal than other software design projects, and therefore are not given the same funding and employee attention. How are Web developers handling this problem?

Several Web developers were interviewed on these topics. They ranged from small developers who do some consulting work on the side to developers for relatively large Web design firms that create e-commerce sites as their main business. Several themes developed during these conversations. All developers have faced problems in communicating effectively with clients. Most have horror stories of what can happen when clients expect one thing and the developers deliver something else. Most developers have tried to solve this problem with a variety of methods. Some developers simply conclude that it is up to the client to determine all content for the Web site. They ask the client for a list of needed features and content, and then have the client sign off that that is all the site will do. Others try to guess what the client really needs. Either approach can lead to hard feelings or even legal action.

This thesis proposes a new design methodology that offers several tools to address the communications problems associated with modern Web design and the overriding need to obtain correct requirements. Since there are a wide variety of Web projects of various sizes, this methodology is flexible. Some of the techniques are useful for all Web projects, while others would be cost-effective only for larger projects.

The first tool will address the problem of a lack of a common language between developers and clients. It is a common technique to ask potential clients what Web sites they like. Most developers interviewed used this tactic and would give the client a list of sites to look

at to get ideas. Most developers also have a portfolio of Web pages that they have created which they let clients view. The problem with this approach is that the client is not given any instruction as to what they should be looking for. An annotated portfolio, customized for each client, is a method to teach the client about specific Web features and provide a common language for discussions.

The second tool in the methodology assists the client in determining what content to put on the Web site and how it should be arranged. Once the manager is relatively educated about the options available, the next step is a tool to help him/her determine which content to put on a Web site. I am proposing an enhancement of the Web card method (Lazar, 2001) used in User-Centered Web design to help meet this need. The improved WebCard system will allow the Web team to obtain content from many sources and keep it organized. The WebCard system will also allow the team to develop a navigational scheme.

The output from the WebCard process is next moved to an online content tracking system. One of the most frequent problems in working in a Web design team is determining where the content is going to come from, and who is authorized to make changes. Gathering this information with the content and organizing it into a system makes it is easy to determine what information has been provided to the Web team and what is still missing. Any team member or the manager can look at the content tracking system to see exactly who is responsible for providing any missing content.

The final tool in the methodology is the modification of the standard SRS, or software requirements specification, to allow for a formal agreement on requirements. The SRS is commonly used in traditional software development projects. I have modified this tool to make it compatible with the special challenges a Web site development project includes.

CHAPTER 2

LITERATURE REVIEW

Overview

A starting point for the development of a new Web Design Methodology is to look at existing methodologies. There are several developing schools of Web design. User-centered Web Design attempts to create Web sites that are dictated by the needs of the actual or potential users of the site. Web Engineering attempts to use some aspects of software engineering in Web Design. Information Architecture deals with the structure and presentation of Information and is only tangentially related to Web Design. In addition, more general-purpose Web development books often have a section that presents some design methodology in at least an embryonic or simplified form.

General Web Design Sources

There are numerous general-purpose Web design books on the market. Even though "Web Design" is part of the title of many of these books, most would be more accurately described as "HTML how-to" books. The backgrounds of the authors of these books tend to be in areas such as graphic design or technical writing. The requirement gathering process is rarely given more than a few paragraphs.

Friedlein's book is a good example of the genre. The author is a former television producer turned Web developer with no discernible experience in software engineering (Friedlein, 2001, p. xiii). The book is similar to other books on Web development that present strategies for producing a successful Web site. An interesting difference is that many of the titles of the sections of this book appear to cover software engineering topics. Upon closer inspection,

however, these sections offer no techniques, formal or otherwise, to accomplish the stated section goals. Instead, the reader is just told that he/she should have, for example, a functional specification that describes the site. There are no guidelines on how to create such a document, or more importantly, how to get the information from the client to create one. The advice of *Web Project Management* on requirements gathering amounts merely to the implication that the author is in favor of it.

User-Centered Design

Lazar (2001) provides a good example of the User-Centered Design school of Web development. In this book, Lazar argues that most clients know the content that needs to be on their Web site. All they need is help in organizing the content so that users of the site will be able to find the information they need. Sample users are gathered, and they are asked how they would organize the information. The sample user's opinions form the basis for this methodology.

Lazar's book does not add anything to traditional Web requirement gathering. No attempt at educating the client about the Web or the developer about the client's business is suggested. Clients and potential users of the site are asked to enter content elements for the site onto index cards. No critical review of the appropriateness of the content is suggested, nor are the cards condensed to incorporate similar ideas from more than one source. The cards are then sorted by different users to determine a navigation scheme.

A more complete development of the card sorting process can be found in Fucella's article (1997). This article introduces the concept of card sorting for Web design and suffers from the same problem as does Lazar's. Card sorting is a valuable tool in Web design, but it is not the only tool. The users of a Web site are not the only ones to have valid input into the

content and structure of the site. The other problem with user-centered design is that the cards are used for navigation only. They are not tied into content management or converted into a requirements specification of any kind. The card methodology is related to the CRC card method (Cockburn, n.d.) often used in software engineering, but hold less data.

An alternate approach using paper prototypes is found in the article by Grady (2000). This article describes a system for using colored pieces of paper to create a prototype. The requirement gathering aspect of this process is mostly left to the reader. Grady recommends that Web content be gathered by looking at similar Web projects. All content should be divided into categories based on a pre-existing purpose statement for the Web site. Users are not asked for their input in the planning stage. The content is then organized into different colored pieces of paper. The user interaction comes in as a usability test, in which potential users are asked to navigate the paper prototype. The feedback gathered from this process is used to create iterations of the paper prototype until the desired level of usability is gained. Grady assumes that the goals of the Web site are established in advance, and thus that the content to include is clear. This technique is really just about navigation design, and does not address the entire Web site development problem.

Web Engineering

Web Engineering is a term applied to the school of Web design that attempts to modify the tools of traditional software engineering to make them more usable in the Web environment. This thesis falls into this category. Web Engineering materials currently available fall into two broad categories: those whose authors come from a software engineering background, whose grasp on Web specifics can be deficient, and those whose authors are from a Web background

and who often lack sufficient background in software engineering. The necessary combination of these two distinct disciplines is stated by Roger S. Pressman:

> But what if the current ad hoc approach to Web development persists? In the absence of a disciplined process for developing Web-based systems, there is increasing concern that we may face serious problems in the successful development, deployment, and "maintenance" of these systems. (Pressman, 2001, p. 770)

Web Engineering (Powell, 1998) is one of the more complete works on the topic. The first few chapters of his book discuss the various development models from software engineering, such as the waterfall model, and make adjustments for each model to enable a closer fit with Web development. This excellent work is beyond the scope of this thesis, as it does not focus on requirement gathering. Chapter 5 on requirements analysis and specification is of most immediate concern. According to Powell (1998), "The main goal of the requirements specification phase is to set boundaries and limits for the project" (p. 115). The problem with this idea is that the focus is at too high a level to be useful. The developer needs to know what information actually to include on a site, not what kind of information *could* be appropriate. Content is to come from two sources, existing content and new content (Powell, 1998, p. 126). No further instruction is given as to how to gather or process either type of content. While useful for its coverage of development models, and other important information, Web Site Engineering falls short of providing the tools for successful content gathering.

Information Architecture

Information Architecture (IA) is one of the newest trends in Web development. It attempts to offer a comprehensive plan for developing large scale Web sites. Rosenfeld and Morville (2002) map out the techniques that form IA:

Information is defined as facts and figures. The main task of IA is to structure, organize and label this information. Further, IA is concerned with how users find information on a site and how the producers of the information can manage it (Rosenfeld & Morville, 2002, p. 5). Using IA, one can develop searching systems, navigation systems and semantic networks to assist users in finding the information they need (Rosenfeld & Morville, 2002, p.14). The deliverables of IA can be wireframes or blueprints of a Web design, or a controlled vocabulary list or a metadata schema (Rosenfeld & Morville, 2002, p.15).

IA advocates a balance between the needs of the user, which is the sole focus of user-centered Web design, and the needs of the organization producing the information, which IA terms as "context". IA lists three parts of the complex system of a Web site as context, content, and users. The context is the business situation that the Web site will be placed in. It encompasses the requirements that lead a business to create a Web site. The content is all of the information that needs to be placed somewhere on the Web site (Rosenfeld & Morville, 2002, p. 25). The users are the people who will actually use the site. There are most likely going to be different groups of users who have different information needs (Rosenfeld & Morville, 2002, p. 26).

IA is most concerned with identifying the structure of the information to be presented in a Web site. It defines techniques for labeling, or giving a name to a group of related information objects. IA offers ideas for the types of structures that Web content can take and discusses theoretical search strategies. The discussions are informative but do not offer techniques for actually gathering information and developing Web sites.

Most Web development books and articles refer to the requirements gathering process, but none offer any concrete solutions as to how to make the process go more smoothly. General Web design books do no more than mention that getting the requirements correct is important.

User-centered Web Design is focused on usability after content is known and on Web site refactoring, and thus is of limited use in new Web design. Web Engineering thus far has been more concerned with the development models of Web design than with requirements gathering. Information Architecture deals primarily with the underlying structure of information, and has little to say about how to gather relevant content. Each school of thought has some important things to add to Web design, but none of them deal adequately with requirements gathering.

CHAPTER 3

TECHNIQUES

The methodology I am proposing is designed to be modular. It begins with general tools such as the annotated portfolio, which helps overcome the communication problems inherent in Web development projects. This tool has wide applicability for most, if not all, Web projects. The methodology then moves on to more specific tools such as the expanded WebCard system to help in content gathering and site navigation design. The system can also be used to aid the programmers if a more robust system is being developed. The WebCard system can also tie in to a content management system used to track from where content is supposed to come, when it should be delivered, and in what form it should be delivered, so that all parties can monitor the progress of the project. A final tool in the methodology is an extension of the widely used SRS. This detailed specification can be used as a legally binding contract between the client and the developer.

Closing the Communication Gap: The Annotated Portfolio

As mentioned previously, the biggest problem facing the design team is lack of a common language. Most businesses have developed a specialized language that relates to their specific environments. Most Web developers are not initially familiar with their clients' business. Web developers themselves use their own language, which is highly technical in nature and far beyond the ken of the normal Web user. Consequentially, the client knows what his business needs are but not what is possible or useful on the Web. The designer knows how to create all kinds of Web pages, but does not know what features might add value to the business. Obviously, means are needed to bridge this gap.

It is also wise at this point to determine the conditions under which the finished site will be hosted. What is the budget for the project? Will the client host the site? If so, what are the technical limitations of the client's server? What server-side programming languages can be used if needed? Are there any storage restrictions? If the client prefers that the designer do the hosting, can the designer handle this? What about domain names? Is there an existing domain name that needs to be moved? What about the minimum screen resolution for the site? Should the site be 600 pixels wide or wider? What kind of performance is expected from the site? What level of security is acceptable? All of these factors play a part in how the site is designed and should be written up and signed by both parties before continuing.

Developer Education

The developer should spend some time with the client to see exactly what it is that the client's business does on a daily basis. The developer should pay special attention to the flow of information throughout the business. Where does the information come from? Where does it go? How does a customer go about making an order? Who needs to be informed of the order? The developer should obtain copies of all forms the business uses. How could a Web site enhance the value of the business? The often non-billable time spent on this phase of the project will almost always pay dividends later in the form of the developer's having a thorough understanding of the client's business.

Client Education

The client might or might not have a general understanding of the Web. Many people now routinely use the Web for entertainment or information. Few clients, however, have a thorough understanding of what the different Web features do, how they could be implemented

to help their business, what the features are called, and how complicated it might be to make it happen. Traditionally, Web developers have shown their clients a portfolio of their work. Typically, this is either online, or a printed brochure whose main purpose is to let the potential client know what visually pleasing Web sites the developer had created. The individual features of the site are not often explained, nor are the rationales for using them included.

A more effective approach would be for the developer to create an annotated portfolio specifically for each client. This portfolio is a Web page that contains descriptive links to different Web sites' features that show examples from different categories of sites. The client should be shown both features from static brochure-ware sites that display only information about a company and from complex e-commerce applications. Ideally, the designer will have a portfolio of his or her own work that covers the desired range. If not, the designer should keep an updated list of good examples of other sites sorted by feature. The designer should also seek out sites of organizations that are similar in scope to the client's, or visit some Web pages of similar types of businesses. The useful features found on these pages should be represented in the client's portfolio.

The first step in developing the annotated portfolio is to determine which features the client might be interested in. This feature list should range outside any preconceived ideas that the developer has about what the client ought to want. The only features to be excluded are those that, in the opinion of the developer, would be inappropriate for the client's site. Once the list of features is set, the designer needs to find good examples of each. The portfolio should explain what the client is going to see. The portfolio should open the linked page in a new, smaller window, so that the client will not become distracted, and forget what they are supposed to be doing. Each link should describe only the page it is linking to. If the designer wants to point to more than one feature of a given Web site, it should be done through separate links.

Items to show the client could include an informational page about the client organization, a product catalog, some detailed product or service information, and a shopping cart. If appropriate, the portfolio might also demonstrate chat or other interactive features. The designer should never show a client a poorly designed site, especially one with too much flashy animation, as s/he might conceivably want one just like it! An example of an annotated portfolio can be found on the Web at http://cscilinux.northeaststate.edu/~defrazier/research/portfolio.html, and in Appendix B.

The annotated portfolio should be presented and explained to the client by the developer. The annotated portfolio should be online, and the client should be encouraged to view the document over several days to fully understand the features. All interested or involved parties in the client's business should view the annotated portfolio and take notes on what they do and do not like. The developer should ask the client to point out any other Web sites that s/he finds interesting. The developer and client should then sit down and discuss the results of this review. It is often the case that the client's vision of what the Web site can and should be has been changed and expanded. The operating assumptions gathered and written up in the first stage of the process might need to be revisited and revised.

After the annotated portfolio has been reviewed, the client should have a better idea of what can be done on a Web site. All involved in the project should have a good sense of where development of the site is headed. There should be a revised sense of purpose for the Web site and a good idea of which features are to be implemented on the site.

Making the Data Object Cards

It is during this education process that the first data object cards will be produced. Each card represents one Web page in the completed site. As the client and designer view other pages,

they use index cards to keep of track of the page ideas that they like. For example, if the project is for a coffee shop that has a stage and regular performances, the client might find a page called Happenings on the Web site of a similar establishment in another town. If the client likes this idea, s/he can make an index card with the title Happenings. A short description would also go on the card. At this stage, that is all the card needs to contain. As the designer and client view more pages, they will just add new cards every time they find Web pages that they like. An example of a Data Object card can be found in Appendix B.

It is important to explain to all involved what constitutes a data object. This can vary by project, but in general it is the amount of information that can appear on one relatively short Web page. If the data objects are too broad, necessary levels of navigation may be missed. If the data objects are too narrow, increased work will be needed to create and organize them. When in doubt, it is usually advisable to split the data object into smaller parts. It is easier to combine cards later than it is to break them apart.

Data object cards are created in several ways. A few cards are developed during viewing of the annotated portfolio. More cards are developed by the client's personnel as they think through what information should be displayed or collected in order to accomplish the features identified for the business. Different units of the business will usually have their own areas for which to create cards. It is also a good idea to have some brainstorming sessions that bring together people from a cross section of the business areas to make sure that cross-departmental information is not being missed. The developer should sit in on some of these sessions to ensure that this process goes smoothly.

At the end of the card development process, the client should have a better idea about what s/he would like to see on the Web site. There also should be a large stack of data object cards, each developed by individuals. The last step in determining content is to bring the

designer(s) and the client(s) together for a final brainstorming session. This meeting provides everyone a chance to display their cards. Cards with little support from the client should be discarded. New cards may be added as necessary. All duplicate cards should be removed. Cards removed from consideration should be separated from the rest, so they will not pop up in the stack later. All business processes and customer transactions should be gone through to ensure there are cards for all the requisite parts.

The designer now should have a large stack of index cards with data objects on them. The next step is for the designer to determine which of the data objects are most likely outside the scope and budget of the project. These cards should be moved to an envelope titled "Future Development." The designer next should schedule a meeting with whomever will make the final decisions for the project in order to finalize the data objects to be included on the site. The client should see which data objects are being marked for future development, to make sure that s/he does not consider the features to be essential. After this meeting, the designer should have a stack of index cards that represent all of the Web pages that will be created for the site. The client should sign off that this represents the content that will be included on the site.

Content Matrix

The next step is to determine where the information contained in the data object cards is located—that is, who has the information and who will provide it to the designer. Problems could arise if the designer assumes that the client is going to provide the information for a data object card, while the client makes exactly the opposite assumption. It might be that a data object will have to be moved to the "Future Development" envelope if no resources are available to create it. A content matrix should be developed as a spreadsheet that lists each data objects on a separate row. For each data object, columns should be included in which to note who is

providing the information (the actual person responsible), the format of the data (electronic, on paper, in Word, in HTML), who has the authority to change the data, how often the data will change, and the date on which the content will be provided to the designer. The latter column will most likely be filled in later in the project.

Once the content matrix is complete, the source and type fields must be completed on the data object cards. The designer also should determine whether the data object can be a static page, or whether it should be made dynamic. This decision will rest on how frequently the information needs to be changed, and by whom. If the page is to be dynamic, it must be determined how changes will take place, and who can make them.

Navigation Design

Now that it has been determined what information will be presented on the site, the data object cards will be used to design the navigational structure. The stack should be given to each team member of the client organization involved in the project. According to Powell, each Web page should have no more than seven links plus or minus two (Powell, 2002, p. 33). The task is, therefore, to divide all of the cards into five to nine piles by placing like cards together. Once the piles are created, the team member must construct a navigation card for each stack. This navigation card should be a different color, so it will stand out. The title of the navigation card should be a descriptive word or phrase that describes the stack. In addition, as these are the first piles created, the cards are numbered by placing a 1.x in the top right hand corner of the card, where x starts with a one for the first pile and moves up to however many cards there are. Once the team member is satisfied with the stacks and the navigation cards, a home page navigation card must be created. The home page navigation has "Home Page" as a title, and the content will be the titles of the other navigational cards. The Home Page card is numbered 1.

This process must be repeated whenever there are more than five to nine cards in a pile. As new piles are created, the team member should create new navigational cards for each pile. The numbering scheme will add a digit at each level. For example, if navigation card 1.3 is divided into 3 smaller piles, these piles will be labeled 1.3.1, 1.3.2, and 1.3.3. When completed, there should be a series of related piles, each having no more than nine cards. Every data object card will be listed on only one navigation card. Once the team member is happy with the piles, the navigation cards can be removed and the cards shuffled. The data object cards are then handed off to the next team member, if there is one, and the process starts over again.

It is useful to have some potential users of the site go through the sorting process as well. The designer should ensure that the audience groups identified earlier are still valid, and make any necessary corrections. The designer should pick representatives from each audience group for this project.

Once everyone has been through the sorting process, the designer needs to look at all of the sets of navigation cards to see whether there is some consensus. It will most often be the case that different audience groups will have different navigational needs. The designer will need to take the raw hierarchy of the Web Cards and mold them into a more complex, cohesive navigational structure, producing a final set of navigation cards. This final set should be presented at another meeting to everyone working on the project. Any differences of opinion should be worked out at this time. The final navigational structure should be presented for signoff to whomever is making the final decisions about the project for their sign-off.

The navigation numbering scheme should now be transferred to the top right of the data object cards. A card's number will refer to the navigational page from which will be linked. For example, if a page is linked from navigational page 1.3.4, that would be its number. These are

not unique identifiers; rather they show the location of the page in question. The cards themselves will become part of the documentation of the project.

Testing the Navigational Scheme

Before any further development is done, the designer should test the navigational scheme to make sure it is workable for all likely visitors to the site. For each audience group, the designer must determine what actions that they are likely to want to take on the site. Each of these user actions should be turned into a standard UML Use Case Scenario. The Use Cases will use the navigation scheme to determine what actions the user will need to take, and what response the system will have. These Use Cases should be reviewed by the designer to make sure that there are no tasks that are made too complex by the existing navigational scheme. Navigational changes might need to be in order to made to make some of the Use Cases less complex. If a change is made, all use cases that utilize the affected pages will need to be redone. The results of this analysis should be presented to the client along with any changes made to the navigation. The Use Cases will enter into the requirements documentation for the project.

WebCard Example

Assume that Ye Olde Bookstore wants to create a Web site. Some sample cards that they might create are listed in Figure 1. The first card describes the mystery book section of the Web page. The second card describes the science fiction book section of the Web site. The numbers for these cards are 1.1.1 and 1.1.2, respectively. Card 1 always refers to the main index page of the site. In this example, after doing a card sort, it was determined that card 1.1 would be the books section of the Web site, as that is Ye Olde Bookstore's main line of business. Cards 1.2, 1.3 and so on would reflect other lines of business for the shop, such as videos or CDs. The

1.1.1 card below for the mystery book section would be the first listing on page 1.1, the books page. Similarly, card 1.1.2 for the science fiction section would be the second listing on page 1.1. For both cards, the person responsible for getting the information together for the developer is Bob Jones. In addition, Bob Jones is the owner of the information, meaning that only Jones can update or make additions to either of these pages. (In some cases, the source and the owner may not be the same.)

Card Title:	Mystery Books	**Number:**	1.1.1
Description:	A listing of mystery books for sale		
Source:	Bob Jones	**Owner:**	Bob Jones
Date Expected:		**Date Received:**	

Card Title:	Science Fiction Books	**Number:**	1.1.2
Description:			
Source:	Bob Jones	**Owner:**	Bob Jones
Date Expected:		**Date Received:**	

Figure 1. Sample of the visual layout of two WebCards.

Requirements Specification Document – The WebSRS

The designer now should have a very good idea of what to include in the project. It is time to create a formal requirement specification document. The standard document for software development outside of Web applications has been the Software Requirements Specification, or SRS (http://kybele.escet.urjc.es/Documentos/ISI/IEEE-STD-830-1998.pdf). This document is often used as a legal contract between software developer and customer. An SRS is a formal description of the software project and includes expected system behavior, performance, design

constraints, validation criteria, and any other special features of the product. Web development projects should use a version of the SRS to assist in communication between the development team and the customer.

IEEE Standard 830 (http://kybele.escet.urjc.es/Documentos/ISI/IEEE-STD-830-1998.pdf) defines the SRS for traditional software development. Starting from the IEEE standard, I propose to develop a Web-specific SRS format, or WebSRS, to address the unique needs of a Web development project. In my interviews with Web developers, the problem of getting the requirements right was the number one concern for all of them. None of the developers had heard of an SRS itself, but thought that a WebSRS would be a good idea. Several developers had created a standard requirements document, although none had the level of detail found in the SRS.

Section 1 of the standard IEEE SRS is the Introduction. This section provides a project overview, and includes subsections on the purpose and scope of the project. There is also a subsection that presents domain knowledge by defining key terms and spelling out any abbreviations or acronyms used. This section is immediately usable for Web Development. It is generic enough to cover Web projects adequately. The type of information contained in this section is extremely important for the success of a Web project but is often not written down. The more experienced developers interviewed always had the client produce a written purpose statement. The less experienced developers reported that many of their projects had been based mostly on verbal requirements. It was assumed that the developer and customer agreed on all aspects of domain knowledge. As a project progresses, however, this often proves to be a false assumption. While this section of the SRS does not need modification, its importance does need to be stressed to the Web development community.

Section 2 of the standard IEEE SRS describes the environment in which the project will be developed and function. This section includes subsections on such topics as user interfaces, product functions, and user characteristics. These subsections will require changes in order to make them more useful for Web projects. For example, it is often far more difficult to know who your users might be on a Web project than on a traditional software project.

Use cases have been used on traditional project management projects, but should be modified for Web use. Techniques from User-Centered Design (Lazar, 2001) can be helpful here. Use cases on a Web project need to address the different audiences that could be using a site. The overly graphical nature of the Web also needs to be reflected in the Use Case. These modifications are minor and could be implemented easily.

Look and feel is of paramount importance in a Web project, and product functions will have to address this importance. The customer cannot just decide what the product needs to do. The customer must also, to some degree, decide how it is to do it. Functions also have to be presented in relation to existing services.

Developers need prototyping tools that show the user interface to the customer. With the complexity of many Web pages, it is not always feasible to develop actual pages for the customer to look at. Another problem with developing actual sample pages is that the client might believe that too much work has already been done to justify requests for changes. In such cases, the client might sign off on the prototype but never be fully satisfied with the end result.

A related topic not dealt with in the IEEE SRS is Navigation. The navigation scheme that results from the WebCard process should be added to the WebSRS. The IEEE SRS also includes a subsection on constraints on the software project. This is an important section for any project, but special considerations are needed for Web projects. Is there an existing Web server that must be used? Who will host the site's pages and run the server? What can or can't the

developer do on the server? These questions have a great potential to sink a Web project if not dealt with early. A beautiful Java-based E-commerce site, for example, will be of no use to a company that must run .Net applications only.

Section 3 of the standard SRS details the project's requirements. This section must be specific enough to allow the developers to design a working system. It includes functional and non-functional requirements and external interfaces. The techniques in this section will need to be modified to make them relevant to Web projects. UML diagrams are often used in a standard SRS to show the application's object-oriented structure. I have found some attempts to modify UML for Web applications (see, for example, Conallen, n.d.); this approach, however, is too program-specific and thus too far removed from the navigational structure of the overall site. A better approach is the enhancement of the WebCard system to include the objects that will create the content. Special consideration must be given to finding or developing tools that will allow the client to express requirements in a way that the developer will understand. As in section 2, care should be taken to show both the technical requirements and the look and feel that the site must exhibit.

Section 3 also contains information about security and performance requirements for the project. Performance measures on the Web are mostly dependent upon network speed; thus care should be taken by the developer not to promise performance levels that are out of his/her hands to deliver. Security, on the other hand, is a much more important area. The security of the WebSRS will in general be more specific and more involved than a standard SRS, due to the inherent risk associated with the project being accessible via the Web. A detailed list of what information is sensitive should be provided in this section, along with a security scheme to protect the data. Table 1 illustrates how the security section of Ye Olde Bookstore would differ in a standard SRS and a WebSRS.

Table 1

Comparison Between SRS and WebSRS

SRS	Customers can view most Web pages without having to log in. If customers want to place an order, they will have to create a username and login.
WebSRS	The server hosting the site should be in a secured location. The server should be maintained regularly to ensure that all security updates are installed. All communications with the server should be done through the secure https protocol. All unnecessary ports should be closed. Transaction processing should be used to ensure that only complete credit card transactions are processed.

Section 4 of the standard SRS is for supporting information, and is less structured than other sections. Anything that makes the SRS easier to understand can be included in this section. There is no particular Web enhancement needed for this section.

Once the WebSRS has been completed, the client should review it. This is a formal representation of what the developer plans to do for the client. Anything that is unclear to the client should be explained or rewritten. In many cases, the SRS becomes a legally binding contract between developer and client which fully specifies what services will be provided. If the WebSRS is to be used in this way, both the client and developer should consult legal advice before proceeding.

CHAPTER 4

EXPERIMENTAL DESIGN

The starting point for my experimental design was pre-project interviews with several Web designers. I discussed with them the techniques they used to get requirement information from their clients. As part of this process, I discussed the problems that I myself had encountered as a Web developer in order to see whether they have had similar difficulties. Everyone I talked to had experienced problems in the same areas, such as getting the client to tell them what s/he wanted the Web site to do, lack of a commitment to provide needed materials in a timely fashion, lack of a commitment of necessary resources to get the job done, and a general lack of a shared vocabulary between client and developer in order to discuss what needed to be done. This, along with a search of the computer science literature, convinced me that I was identifying a widespread problem in need of a solution.

Difficulty of Testing

The testing of a methodology can be difficult to accomplish. To obtain a true test of the methods contained in my proposed Web methodology, two Web development teams with equal skills would need to be deployed to the same, or at least very similar, clients. One team would use a more traditional approach to Web development. The other team would use the proposed new methodology. Interviews could then be conducted to determine which group produced the better end results. I attempted such an approach with students of a Web Programming class that I taught at Northeast State Technical Community College in Fall 2003. I asked a faculty member who was planning to implement a Web site for the faculty senate to act as client. I divided the class of eight students into two teams. The first team was instructed to use traditional techniques, such as finding similar Web sites to show the client, and asking him or her what s/he would like to see in a Web page. The second team was instructed to use the extended WebCard

system to elicit requirements. The teams were instructed not to communicate with each other. The faculty member was instructed to disregard what he had heard from Team Number One when Team Number Two arrived. Unfortunately, the faculty member already had a very good idea of what he wanted, which was confirmed by team one showing him some Web sites. By the time Team Number Two arrived, his mind was already made up, and thus he had no interest in completing any cards.

This aborted attempt shows the difficulty in testing this kind of methodology. It is nearly impossible to create teams of equal skill. Further, it is difficult to use the same client more than once. If the client hears a good idea from one group, s/he likely cannot just forget it. Another problem in testing this approach is that it is best suited to large projects. I could find no one willing to allow me to test my methodology on his or her real-life Web projects. Obviously, alternate means of testing needed to be developed. The annotated portfolio method, however, did lend itself to objective testing.

Annotated Portfolio Experiment

The annotated portfolio is designed to inform a relatively naive Web user of some of the features that can be incorporated into a Web page. It also creates the basis for a shared vocabulary to allow clients and developers to communicate more effectively with each other. Thus, it was not mandatory to find actual clients willing to take the time to look at an example of an annotated portfolio. Any supply of relatively naive Web users would show the efficacy of the method. In the spring of 2004, I was teaching three sections of an Introduction to Computers course, intended for the general first-year community college student. Over 70 students were enrolled in these three classes. I created a sample annotated portfolio, which can be viewed in Appendix B. I also created a test to determine whether the students were familiar with various Web terms and features. Each class first was given the examination as a pre-test in order to see

what knowledge they possessed before viewing the annotated portfolio. Then they were shown an online version of the portfolio which allowed them actually to follow the links in order to see what the features looked like. They were allowed to view the portfolio with no other instructions. They were then given the examination again as a post-test in order to determine what knowledge they gained by the viewing the portfolio.

Other Techniques

In an effort to obtain some feedback on the methodology as a whole, I created a Web site (http://cscilinux.northeaststate.edu/research/) that explained all of the methods described in this methodology. There were sections on creating an annotated portfolio, creating and using WebCards, tracking the creation and delivery of content items, determining navigational design, and creating a WebSRS. There was also a feedback form that asked the viewer for input on each of the sections individually, and on the methodology as a whole.

Announcements of the availability of this Web page and feedback form were posted on several mailing lists that cater to Web developers. In addition, I contacted several Web developers to whom I had spoken to prior to initiating this project, to get their feedback.

CHAPTER 5

RESULTS

Annotated Portfolio Experiment

To test the effectiveness of the Annotated Portfolio method, I created an examination to test the students' prior knowledge of various Web design features. A copy of this examination can be found in Appendix D. I also created a sample annotated portfolio that discussed the material found on the quiz, and pointed to some examples of Web pages that show the given feature in use. This annotated portfolio is available online at http://northeaststate.edu/research/portfolio.html. On April 7 and 8, 2004, 42 students in an introductory Computer Concept class at Northeast State Technical Community College were given the examination. The students were informed that the examination was optional, and would in no way affect their grade in the course. The students then were instructed to go through the annotated portfolio and follow the provided links. After viewing the portfolio, students were then instructed to take the examination again. The before and after quizzes for each student were marked with either a B for the pre-test or an A for the post-test and then stapled together.

The average score for the quizzes taken before viewing the portfolio was 14%, with a high score of 67% and a low score of 0%. Many students were able correctly to identify the difference between an Intranet and a Public site. The average score after viewing the portfolio increased to 86%, with a high score of 100% and a low score of 33%. Every student who took the examination had an increased score after viewing the portfolio. The average increase was 71%.

Methodology Review

To test the rest of the methodology as a whole, I created a Web site (http://cscilinux.northeaststate.edu/research) that explains all of the various parts of the methodology. The site includes a feedback form on which viewers of the site can give their opinions of the different tools in the methodology. The results from the feedback form are stored in a database for review. I then advertised the Web site on several mailing lists that deal with Web development issues.

Twenty people took the survey from June 1, 2004 to September 1, 2004. Experienced designers were targeted. The average amount of experience in Web design reported by the participants was 4.1 years, with a low response of one year and the most common response of 5 years. The next section of the survey asked for information about the users' background. Sixty-five percent of the respondents reported they had a programming background, 35% a graphics background, 25% a technical writing background, and 20% an education background. Twenty percent of the respondents selected only the Other category for background.

The next section of the feedback form asked if the respondents were currently using a formal requirements gathering system. Only 35% responded that they were using a formal system. Some comments indicated that many respondents have developed their own informal systems to gather requirements.

The respondents were then asked if they had any intentions of trying the methodology. Eighty percent of the respondents reported that they were planning to implement some of the methods described.

The final section of the feedback form allowed the users to give impressions of the usefulness of the overall methodology, as well as of each individual method. The choices for

each were Poor, which was coded as a 1; Fair, coded as a 2; Useful, coded as a 3; Very Helpful, coded as a 4, and Excellent, coded as a 5.

Response on the usefulness of the overall methodology ranged from one 2 (Fair) to three 5's (Excellent). The average score for this question was 3.6, a little more than half way between Useful and Very Helpful. Of the individual methods, the content management system and the WebSRS were the most popular, each averaging a 3.8. The WebCard method followed closely at a 3.65 average. The annotated portfolio was least popular with the respondents, but still averaged a respectable 3.55.

There was also a section on the feedback form for comments. Six of the twenty respondents added comments. Four of the comments were in general agreement with the methodology. One comment suggested that the methodology was most useful for small static sites, quite the opposite of the proposed use of the methods. For larger projects, the responder suggested simply using UML. The last comment questioned the business logic of my methodology. In this long comment, the responder explained that the business model that s/he is working from involved a prepackaged content management and e-commerce system. In the responder's opinion, the client should know what they want, and be responsible for providing that information to the developer.

CHAPTER 6

CONCLUSION

Overall, my Web design methodology was successful with most of the developers who looked at it. I was surprised by the interest expressed by some smaller Web developers, as my intent was to develop a methodology for large Web site. I was also surprised that there are some large Web design firms that sell content management solutions. These are prepackaged software applications that allow clients to modify their own sites. Much less requirement gathering is needed with this types of site, although it is less flexible in terms of what can be done with the sites.

The results from the annotated portfolio experiments were conclusive, with the average user experiencing a 71% increase in their scores after viewing a sample annotated portfolio. It was somewhat surprising, then, that the annotated portfolio was least popular among the Web developers completing the feedback form. It could be that the perceived difficulty of preparing such a document for each client was a limiting factor.

The methods introduced in this paper need to be tested more thoroughly. Plans are in process to use this methodology on some actual Web projects to see how they perform in the field. There are also plans to contact some of the developers who took the survey to see if they have actually implemented any of the methods.

Further development is also needed to produce an automatic prototype of the Web site to be generated by the content management system. Time and resources were insufficient to add this to the current project.

REFERENCES

Cockburn, Alistair (n.d.). *Using CRC cards*. Retrieved November 1, 2004, from http://alistair.cockburn.us/crystal/articles/ucrcc/usingcrccards.html.

Conallen, Jim (n.d.) *Modeling Web application design with UML.* Retrieved October 25, 2002, from http://www.rational.com/products/whitepapers/100462a.jsp.

Friedlein, Ashley (2001). *Web project management: delivering successful commercial Web sites*. San Mateo, CA: Morgan Kaufmann.

Fuccella, Jeanette (1997). *Using user-centered design methods to create and design usable Web sites.* Proceedings of the 15th Annual International Conference on Computer Documentation.

Ginige, A. and Murugesan, S. (2001). Web engineering: an introduction. *IEEE Multimedia: Special Issue on Web Engineering,* 8(1), 14-18.

Grady, Helen M. (2000). Web site design: a case study in usability testing using paper prototypes. *IEEE Annual Conference on Systems Documentation*.

Lazar, Jonathan (2001). *User-centered Web development*. Boston: Jones and Bartlett.

Powell, Thomas A. (1998). *Web site engineering*. Indianapolis: Prentice Hall PTR.

Powell, Thomas A. (2002). *Web design: the complete reference*. Emeryville, CA: Osborne McGraw-Hill.

Pressman, Roger S. (2001). *Software engineering: a practitioner's approach*. Fifth Edition. Columbus, OH: McGraw-Hill Higher Education.

Rosenfeld, L. & Morville, P. (2002). *Information architecture for the World Wide Web.* Sebastopol, CA: O'Reilly.

APPENDICES

Appendix A

Seven Categories of Web Applications

Category	**Examples**
Informational	Online newspapers, product catalogs
Interactive	Registration forms, online games
Transactional	Electronic shopping, online banking
Workflow	Online planning, inventory systems
Collaborative Work Environments	Distributed authoring systems
Online Communities, Marketplaces	Chat groups, online auctions
Web Portals	

(Ginige and Murugesan, 2002)

Appendix B

Research Web Pages Posted for Comment

Web Requirements Research Page

This page contains a requirement elicitation methodology for Web Development. Please select a link below to find out more information about the various tools in the methodology.
Please be sure to fill out the feedback form. Your experiences with the techniques will allow me to make needed improvements.

User Education

- Annotated Portfolio

Developer Education

- Gaining Domain Knowledge

Content Gathering

- Creating Webcards

Navigation Design

- Sorting the Cards
- Creating a Prototype

Content Management

- Tracking content acquisition

Requirements Specification

- Creating a WebSRS

Feedback
(cscilinux.northeaststate.edu/research/index.html)

Annotated Portfolio

Educating the Client

One way to educate a client about the Web is to give them an annotated portfolio. This portfolio is a Web page that contains descriptive links to different Web site features that show examples from different categories of sites.

The client should be shown features from static brochure-ware sites and from complex e-commerce applications. Typically, the portfolio will contain links to both sites created by the developer and by others. The designer should keep an updated list of good sites, sorted by feature. The designer should also search for organizations that are similar in scope to the client's.

The first step in developing the annotated portfolio is to determine which features the client may need on their Web site. This features list should range outlide any preconceived ideas that the developer has about what the client may need.

The only features to be excluded are those that, in the opinion of the developer, would be inappropriate for the client's site. Once the list of features is set, the designer needs to find good examples of each. The portfolio is more than a list of links. It should first explain what the feature is and how it can be used. Further instruction should be given on what the client should look for while viewing the example page.

The portfolio should then open the linked page in a new, smaller window, so that the client does not lose track of what they are supposed to be doing. Each link should describe only the page it is linking to. If the designer wants to point to more than one feature of a given Web site, it should be done through separately explained links.

Some features to show the client could include an informational page about the client organization, a product catalog, some detailed product or service information, and a shopping cart. If appropriate for the site, the portfolio may also include chat or other interactive features such as a blog.

The designer should never show a client a poorly designed feature or site, as the client may not see it as bad and actually want to implement the feature. Flashy, overly animated sites often appeal to clients, but not to designers.

The finished portfolio should be shown to anyone at the client organization who has an interest in the project.

(cscilinux.northeaststate.edu/research/annotated.html)

Sample Annotated Portfolio

The purpose of this document is to acquaint you with the different Web features available when designing a Web site. Each section contains a brief description of the feature, and many have a Example to a Web page that implements the feature. The Web pages will open in a new window.

Types of Web Pages

- **Public** - Available to anyone.
- **Intranet** - Run on a private network. Available to members of certain companies or groups only. Used for communications between business partners, or between companies and their suppliers.
- **Extranet** - Users must have a username and password to access the page.

Navigational Structure

There are several navigation structures that a Web site can have.

- Narrow Tree - Only a few choices on each page. Easier for the user to navigate, but requires more keystrokes to get to the content.
  Example
- Wide Tree - Many options per page. Can be confusing to the user, but gets them to the content in the fewest number of clicks.
  Example
- Full Mesh - Every page is linked to every other page. This only works on relatively small sites.

Access Options

On some Web pages, a user can directly access any page on the site. This is known as a **porous** site. This makes it easy for the user to bookmark a favorite page.

Another option that allows for more control is to limit the pages that a user can enter on. Then the owners of the site can be assured that the user has some needed information before proceeding. If only one entry page is possible, the site is said to be **solid**. If more than one entry point is possible, the page is said to be **semi-porous**.

Form

A Form can be used to obtain feedback from the user. The information entered on this form can either be mailed to you, or it can be used to update a database.

Example

Product Catalog

If your business has products, you can put them in an online catalog. An online catalog lists your products by category. Under each category, there is a list of products. Each product name can be clicked on to provide more information to the user.

Example An Introduction to Web Features

The purpose of this document is to acquaint you with the different Web features available when designing a Web site. Each section contains a brief description of the feature, and many have a Example to a Web page that implements the feature. The Web pages will open in a new window.

Types of Web Pages

- **Public** - Available to anyone.
- **Intranet** - Run on a private network. Available to members of certain companies or groups only. Used for communications between business partners, or between companies and their suppliers.
- **Extranet** - Users must have a username and password to access the page.

Navigational Structure

There are several navigation structures that a Web site can have.

- Narrow Tree - Only a few choices on each page. Easier for the user to navigate, but requires more keystrokes to get to the content.
  Example
- Wide Tree - Many options per page. Can be confusing to the user, but gets them to the content in the fewest number of clicks.
  Example
- Full Mesh - Every page is linked to every other page. This only works on relatively small sites.

Access Options

On some Web pages, a user can directly access any page on the site. This is known as a **porous** site. This makes it easy for the user to bookmark a favorite page.

Another option that allows for more control is to limit the pages that a user can enter on. Then the owners of the site can be assured that the user has some needed information before proceeding. If only one entry page is possible, the site is said to be **solid**. If more than one entry point is possible, the page is said to be **semi-porous**.

Form

A Form can be used to obtain feedback from the user. The information entered on this form can either be mailed to you, or it can be used to update a database.

Example

Product Catalog

If your business has products, you can put them in an online catalog. An online catalog lists your products by category. Under each category, there is a list of products. Each product name can be clicked on to provide more information to the user.

Example

Shopping Cart

A shopping cart is an essential feature if you want users to be able to buy products online. A shopping cart should allow users to add items to their cart as they move through your product listing. Basic features are the ability to add and remove items, change the quantity, and proceed to the checkout phase when finshed shopping.

Shopping carts come in several styles. Look at the examples below to see which style might suite your type of business.

- Example 1
- Example 2

Blogs

Blogs are a way to allow people to create an online journal. In a commercial setting, a blog could be used to highlight the uses of products. For example, if you have a outdoor equipment online shop, you may want to have a blog where employees tell of their outdoor adventures using the products.

Example

User Supplied Comments

User supplied comments can come in two flavors. The first is to allow user reviews of your products. Scroll to the bottom of the page after following this Example to see some customer reviews.

Example

The other variety lets the user make comments at the end of each page. Scroll to the bottom of the page after following this Example to see the comments.

Example

Shopping Cart
A shopping cart is an essential feature if you want users to be able to buy products online. A shopping cart should allow users to add items to their cart as they move through your product listing. Basic features are the ability to add and remove items, change the quantity, and proceed to the checkout phase when finshed shopping.
Shopping carts come in several styles. Look at the examples below to see which style might suite your type of business.

- Example 1
- Example 2

Blogs
Blogs are a way to allow people to create an online journal. In a commercial setting, a blog could be used to highlight the uses of products. For example, if you have a outdoor equipment online shop, you may want to have a blog where employees tell of their outdoor adventures using the products.
Example
User Supplied Comments
User supplied comments can come in two flavors. The first is to allow user reviews of your products. Scroll to the bottom of the page after following this Example to see some customer reviews.
Example
The other variety lets the user make comments at the end of each page. Scroll to the bottom of the page after following this Example to see the comments.
Example

(cscilinux.northeaststate.edu/research/portfolio.html)

Gaining Domain Knowledge

In designing the annotated portfolio, the developer has taken the first step in gaining some domain knowledge about the client organization. When looking at sites of similar organizations, he or she will gain some understanding of how other businesses in the client's field do their work.

While a good start, this is not sufficient. In order to create an effective Web site, the developer must know the business context in which the site will operate.In the traditional methods of Web development, the designer is often advised to gather from the client all brochures and ads they have produced, along with any forms used in the day to day business.

To more fully understand the client organization, the developer needs to spend some time at the organization. See how the employees interact with customers. Note the work flow. The time spent is this phase will help eliminate communication problems between client and developer.

(cscilinux.northeaststate.edu/research/domain.html)

WebCards

Purpose: The purpose of the WebCards Method is to determine what content to include on a Web page. The method also helps determine an effective navigational structure.

Gathering the Content

All content is to be placed on 3x5 cards. Each card should have separate sections for the title, content, source of the content, who maintains the conent, whether the page will change based on user input, and connections to other pages.

- Title - What the page is to be called
- Content - What information the page will hold
- Source - Who provides the information
- Maintained by - Who owns the info and has the responsibility to update

Every type of information to be found on the page should be on a separate card. For example, if you wanted to include a profile of each employee on the Web site, you would create one Employee profile card. The card would contain all of the information that you want on the page. Be careful here. These cards should contain indivisible units of info. Don't clump too much together. The information on one card should be the content of one Web page.
Some methods to use to generate content

- Brainstorming
- Viewing existing Web site
- Talking to potential users of the site

Several people can generate cards. When all of the cards are completed, the development team needs to go though each card to find duplicates, and generally pare down the number of cards. Some cards may be moved to a future features pile. The end result of this meeting is a set of cards that shows all of the types of pages that are going to be on the site. These cards are known as the Site Content Deck (SCD).

(cscilinux.northeaststate.edu/research/webcard.html)

Sorting the Site Content Deck

Shuffling the Deck
The next step in the process is to divide the SCD into separate piles. Each team member should have a turn at this. The goal is to divide the SCD into 5 to 9 piles by placing related cards together. Once the user is happy with the piles, a new card should be added to the top of the stack that describes the whole pile. This is a navigation card. The navigation level of the card is 1. If possible, navigation cards should be a different color than content cards.
If there are more than 9 cards in a pile, repeat the above process, incrementing the navigation level for each new navigation card.
When you are happy with your piles, write the titles of all sub-piles on each navigation card. Pull out the navigation cards. Shuffle the deck and pass it on to someone else for them to do the same.
Finalizing the Navigation
When everyone has had a chance to create a navigation scheme, have another meeting. Everyone should bring their navigation cards to see how they compare. Go through the navigation to which scheme or combination of schemes works best. Create one set of navigation cards.
The navigation cards function as a paper prototype, and can be used to complete use cases on the site.
Expanding the WebCards method
If you are working on a more complex site, it may be useful to add programming information on the cards, such as what program, object or bean provides information for the page, or what database it is coming from.

(cscilinux.northeaststate.edu/research/sorting.html)Content Tracking

One of the biggest hurdles in creating a Web site is keeping track of the content. Exactly what content will be needed? Who will provide the content, the developer or the client? When will the content be delivered? What format will the content be in? Who will provide updates to the content?
To handle content, you need to create a content matrix. This can be as simple as a spreadsheet where you can keep track of all content on a project. Here is an example:

Name	Description	Provided By	Owned By	Format	Date Expected	Date Received
Company Logo	Main Company Logo	Julie Smith	Julie Smith	GIF	10/5/03	11/5/03

From this brief listing, you can easily see that the Logo was delivered a month later than expected. By using a content matrix, you can see easily what content you have, and what content is still to be delivered.
Using a content matrix has the additional benefit of making clear who is responsible for developing content. I have been on more than one job where I was expecting the client to provide some important content, while they were expecting me to produce it. The content matrix should be given to the client along with a proposal just to make sure that everyone understands the project.

If possible, the content matrix should be posted online somewhere, so that all concerned parties can see it as it is updated. It is possible to create a database application to track content for you.
Example of a Content Matrix Program
There are several advantages to an online database content management system. If designed well, the system will be easy to update. Also, everyone involved will be able to see the current content status. Features can be added to the system relatively easily.

(cscilinux.northeaststate.edu/research/tracking.html)

Creating the WebSRS

Why have a requirements document?
Web projects are often complex. To insure the success of the project the Designer should take pains to ensure that they are creating the Web site that the client expects. One way to ensure that everyone is on the same page is for the designer to create a formal requirements document that details exactly the site to be produced. The client should then review the requirements document, and sign off on it before work proceeds.
Requirements document could be very simple, if the Web site to be produced is not overly complex. It could be nothing more that an agreed upon list of what features the site will contain. The requirement document could also be much more complex. If the site is large, or if it includes advanced features such as a shopping cart, the requirement document has to be through enough to cover the complexity of the site.
Traditional computer programming jobs have long relied on the Software Requirements Specification (SRS) format. The IEEE maintain the standard template for the SRS (830-1993). Web site design differs from traditional software engineering, so I am suggesting the developement of a WebSRS, that modifies the traditional SRS to make it more useful for Web Design.
Sections of the WebSRS

1. Introduction
2. Overall Description
3. Specific Requirements

(cscilinux.northeaststate.edu/research/websrs.html)

WebSRS Section 1 - Introduction

The first section of the WebSRS provides a general overview of the project. What is the purpose of the Web site? Is the Web site being designer to sell products, provide information, create goodwill?
This short, usually informal section is very important for the success of the project. If the client and developer have disagreements on the purpose of the site, a successful development project is impossible.
This section can be divided into 4 subparts:

1.1 Purpose
What is the goal of the Web site

1.2 Definitions
How are certain terms defined in the document. Terms could include "client" and "developer" along with any technical terms.

1.3 Overview
A broad view of the Web site and what if offers

1.4 Copyright
A statement as to who owns the resulting code.
(cscilinux.northeaststate.edu/research/srsIntro.html)

WebSRS Section 2 - Overall Description

This section provides an overall description of the Web site. It is divided into 4 sections.

2.1 Server Specifications

- Who will provide the Web hosting
- What Web server will be run
- What operating system will the server use
- What type of access will the developer have and for how long.

2.2 Major Web Site Functions

This section will have sub-points (2.2.1, 2.2.2, etc.) that give an overview of the major functions of the Web site.

2.3 User Characteristics

This section will describe the different groups of users that will visit the site, along with the type of knowledge of the client and the Internet that the members of each group can be expected to have.

2.4 System Constraints

This section will describe what markup, scripting and programming languages can be used on the site.

2.5 Security

What security measures will need to be taken on this Web site? What areas of the site should require authentication?

(cscilinux.northeaststate.edu/research/srsDesc.html)

WebSRS Section 3 - Specific Requirements

This section provides more details about the Web Site. It is broken up into 4 Subsections

3.1 Function Detail

Each function is described in detail

3.2 Database Requirements

Any database needs are described.

3.3 Maintenance

- Who maintains the site?
- How much maintenance will the developer provide once the job is done?
- Is a cost for maintenance part of this agreement?

(cscilinux.northeaststate.edu/research/srsReq.html)

Appendix C

Sample WebSRS

WebSRS for Ye Olde Bookstore

Section 1 – Introduction

1.1 Purpose

Ye Olde Bookstore specializes in antique and hard to find books. The purpose of the Web site for Ye Olde Bookstore is to provide customers an easy way to see what the store has to offer. Currently, you have to travel to the bookstore or call. The Web site should allow those not within driving distance of the store to be able to order books online.

1.2 Definitions

Store – refers to Ye Olde Bookstore
Developer – refers to Fly-By-Night Web Developers

1.3 Overview

The Ye Olde Bookstore Web site will be a place for book lovers to look for out-of-print and hard to find books. The Web site will allow the customer to search for books or display books by category. If a visitor finds a book they like, they will be able to order it online. If the book is not in stock, the client can asked to be informed if the book becomes available later. The customer should also be able to sign-up for a weekly newsletter of new books received.

1.4 Copyright

The developer assumes that Ye Olde Bookstore has a clear copyright to all information provided to them. The resulting Web site and all associated Web pages, programming and graphics will become the property of Ye Olde Bookstore upon payment to the developer of all fees owed.

Section 2 – Overall Description

2.1 Server Specifications

The site will be hosted by RackSpace. The developer is not responsible for the setup or maintenance of the server. The server will be running an Apache Web server under the Red Hat Linux operating system. The developer will have ftp access to the server during the development phase of the project.

2.2 Major Web Site Functions

2.2.1 Browse for books
The Web site should allow the customer to search for book title, author or keyword. The customer should also be able to search by category.

2.2.2 Shopping Cart
The Web site should have a shopping cart feature to allow the customer to place an order.

1. Newsletter signup

The customer should have the option to sign up for a weekly newsletter about new arrivals.

2. Notification Service

The customer should be able to request Ye Olde Bookstore to contact them if a desired book becomes available.

2. User Characteristics

Customers – Members of the general public. No prior knowledge of the bookstore industry or of the Ye Olde Bookstore can be assumed. Customers are only expected to have a cursory knowledge of the Internet.

Employees – Employees are assumed to be familiar with the store and its policies. Employees are further expected to be familiar with the Web site. Some training for new employees will be necessary, but will not be provided by the developer. Documentation and training materials will be provided by the developer.

Management – Management of the store can be assumed to be knowledgeable about the store and the Web site. Managers are assumed to have an intermediate knowledge of the Internet. Initial training for managers, along with documentation, is to be provided by the developer.

2.4 System Constraints

The Web site has to operate on the Web server rented from RackSpace. The Web site can use HTML, XML, PHP, and the MySQL database program.

3.0 Specific Requirements

3.1 Function Details
3.2 Database Requirements

3.2.1 Book Catalog – A database of books with keywords
3.2.2 Customers – A database of customer information
3.2.3 Employees – A database of employee information
3.3 Maintenance
The developer is responsible for producing, testing and uploading a Web site that meets the requirements specified in this document. Any further changes to the system are outside of the scope of this document.

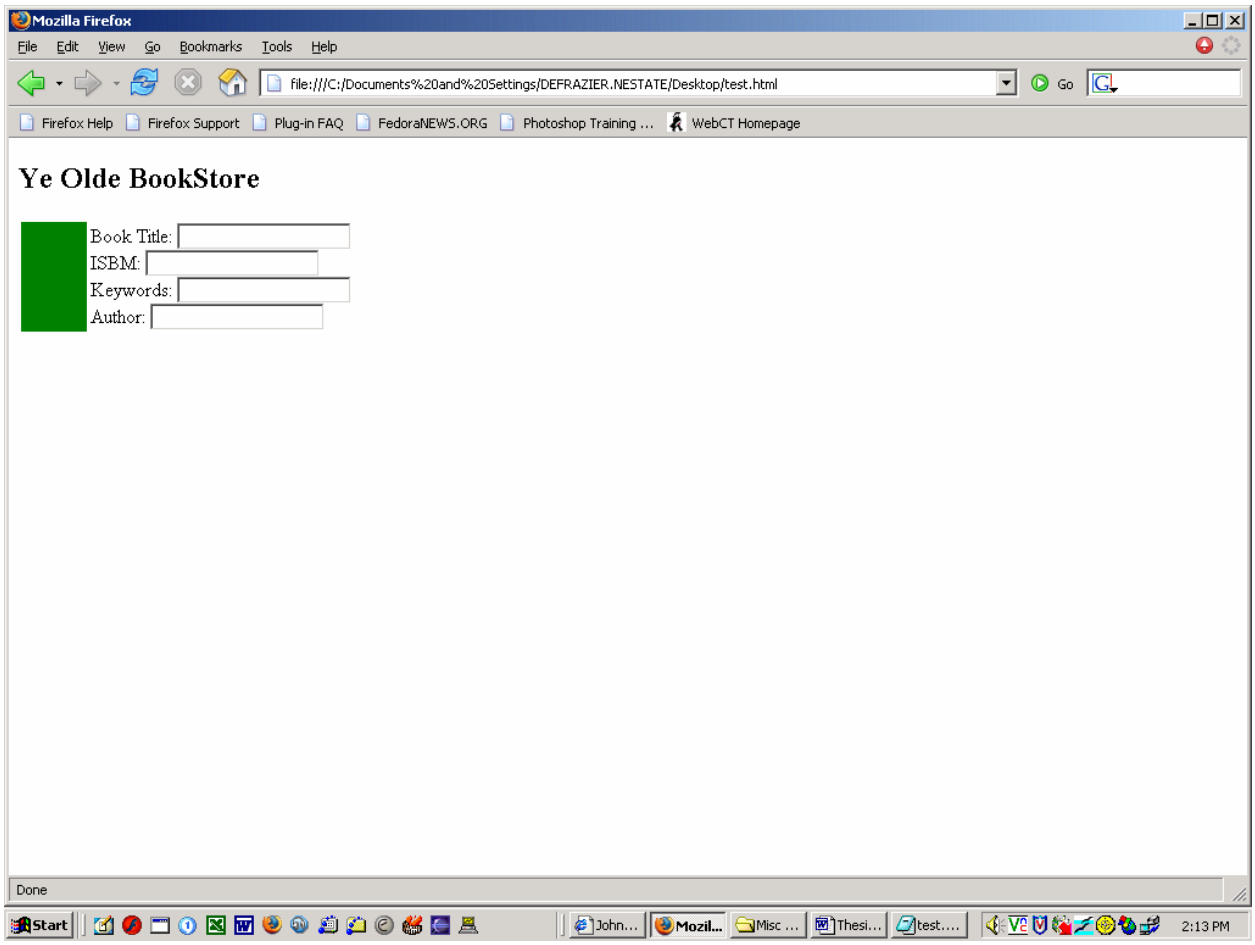
3.4 Security
The database needs to be secure against unauthorized changes. The shopping cart needs to protect the private information of the customer, including Credit Card information.

Attachment A – Content Matrix

Name	Description	Provide By	Owned By	Format	Date Expected
Logo	Ye Olde Bookstore Logo	Fred Murtz	Bill Smith	GIF	08/23/04
Photos	3 or 4 shots of the store	Bonnie Jones	Bill Smith	JPEG	07/05/04
Books	List of all books in catalog	Mindy Michaels	Roger Exum	Comma Delimited	08/06/04
Users	List of employees	Kay Shay	Bill Smith	Text	08/17/04

If information is not received by the developer by the date expected, the project completion date could be delayed.

Attachment B – Screen Shot of Search Box



Appendix D

Annotated Portfolio Test

An Introduction to Web Features

- Explain the difference between a public Web site and an Intranet. When might you use one over the other?

- What is a full mesh, and when can it be used?

- What is a solid Web site? Why would you make a Web site solid?

- What Web feature allows you to get feedback from the user?

- What is a blog?

What features would you need if you want to allow users to purchase items from your site?

GLOSSARY

E-commerce -- The conducting of business communication and transactions over networks and through computers. As most restrictively defined, electronic commerce is the buying and selling of goods and services, and the transfer of funds, through digital communications.

IEEE-- Institute of Electrical and Electronics Engineers. The world's largest technical professional society, based in the USA. The IEEE sponsors technical conferences, symposia and local meetings worldwide, publishes nearly 25% of the world's technical papers in electrical, electronics and computer engineering and computer science, provides educational programs for its members and promotes standardization.

Prototyping -- The creation of a model and the simulation of all aspects of a product. Some prototypes offer the end-user the ability to review all aspects of the user interface and the structure of documentation and reports before code is generated.

Software Engineering -- A systematic approach to the analysis, design, implementation and maintenance of software.

UML -- Unified Modeling Language. A non-proprietary, third generation modeling language. The Unified Modeling Language is an open method used to specify, visualize, construct and document the artifacts of an object-oriented software-intensive system under development.

Web application -- An application program that is designed to be accessed from a Web page.

Web Engineering -- Applying the techniques of software engineering to Web development projects.

Web project -- A software design project that will be accessed through the World Wide Web.

Web site -- All of the Web pages and associated files that make up the Web presence of one organization or entity.

VITA

DAVID E. FRAZIER

Personal Data:	Date of Birth: March 12, 1963 Place of Birth: Ashland, Kentucky Marital Status: Married
Education:	University of Louisville, Louisville, Kentucky Mathematics, B.A., 1986 East Tennessee State University, Johnson City, Tennessee Computer Science, M.S., expected December 2004
Professional Experience:	Instructor of Computer Science, Northeast State Community College; Blountville, Tennessee, 2002-Present Graduate Assistant, East Tennessee State University, Student Affairs Division, 2001-2002 University Web Editor, Morehead State University; Morehead, Kentucky, 2000-2001
Presentations:	"Web Project Requirements Engineering". MidSoutheast Association For Computing Machinery Conference. Gatlinburg, Tennessee, November, 2003. "Talking Heads and Intuition: Artificial Intelligence and Gödel's Incompleteness Theorem in William Gibson's Neuromancer". Twentieth Century Literature Conference, University of Louisville, Louisville, Kentucky, February 2002 (with Jill LeRoy-Frazier, Ph.D.) "User Involvement in Web Design: Two Perspectives"—WebDev Share, University of Indiana, Bloomington, Indiana, September 1998.
Honors and Awards:	Phi Kappa Phi Upsilon Pi Epsilon