Reverse Engineering of Web Pages

Joseph Scoccimaro, Spencer Rugaber
College of Computing
Georgia Institute of Technology
jscoccimaro@gmail.com, spencer@cc.gatech.edu

Abstract

It is estimated that more than ten billion web pages exist on the World Wide Web. Hence, we argue that there exists a web-page maintenance problem analogous to the software maintenance problem. That is, there is a need to understand, correct, enhance, adapt and reuse web pages. Therefore, it is meaningful to ask, what does it mean to reverse engineer a web page? We examine this question and discuss the similarities and differences from software reverse engineering. We discuss the features and design of a tool, Page Monkey, that we have built to support web page reverse engineering and briefly describe user feedback we have received.

1. Introduction

Imagine while web surfing that you visit a web page and wonder how the author achieved a particular effect. You think that you would like to do something similar on a page you are composing. You then view the page source and are bewildered by the complexity of the information you see there. You find it difficult to locate the instructions responsible for the effect that impressed you, and, once found, you have to make several attempts before you can extract all that you need and nothing else.

Web pages are rich in their expressivity. Besides HTML they may contain scripting language and (references to) style sheets. There are a variety of structuring mechanisms used such as tables, frames and headers. Pages may be interactive, employing a variety of user interface devices and the accompanying logic. Many pages are automatically generated, leading to the same sort of comprehension problems found with generated software. In general, web pages compare in complexity to software programs of the same size.

Reverse engineering (of software) is the process of analyzing that software and constructing an abstract representation useful in supporting maintenance and reuse. Although web pages have some of the aspects of software, they are also different. Content and presentation concerns are much greater for web pages while, in most circumstances, logic and semantic precision are reduced. On the other hand, structure, interfaces, and abstraction are common to both.

When reverse engineering software, we can always fall back on execution semantics to situate the various insights we obtain. When reverse engineering a web page, however, while there may be execution aspects, they are often domain specific and provide little help understanding the page source. Instead, we have found that focusing on the "users" of a web page, its visitors, provides the same kind of support for analysis. Presentation, style, structuring, and functionality all contribute to enhancing the user experience.

This paper is organized as follows. In Section 2, we survey the different kinds of analyses that can contribute to understanding web pages. We then look at various technologies that might be used to help automate the reverse engineering of web pages. In Section 4, we discuss the role web page design guidelines can play in understanding web pages. Section 5 presents our tool, Page Monkey, and, in Section 6, we discuss the feedback we have received on it. We then look at other efforts undertaken to understand web pages. In Section 8, future directions for web-page reverse engineering tools are discussed, and we conclude with a discussion of the relationship of web page reverse engineering to software reverse engineering.

2. Characterizing Analyses

As with programs, there are many kinds of analyses that can be performed on web pages. We have surveyed different types of analyses by looking at the features of other related projects and at web page design guidelines, by studying the specification of HTML and by interviewing a web page design expert. The remainder of this section presents a collection of interesting analyses.

Readability Analysis. Readability Analysis predicts the reading skills required to appreciate a text. By obtaining this insight, an analyst can learn the difficulty of understanding and the age level of the intended audience. The most popular readability measurements are provided by the Flesch-Kincaid Reading Level and Flesch-Kincaid Reading Ease

tests[1]. Flesch-Kincaid Reading Level can be computed with the following formula, where *tw* is total words; *ts* is total sentences, and *ty* is total syllables:

*.39 * (tw/ts) + 11.8 * (ty/tw) - 15.59*

The Flesch-Kincaid Reading Level test maps the understandability of a piece of writing into the different US grade levels. Flesch-Kincaid Reading Ease can be calculated with this formula:

206.835 - 1.015 * (tw/ts) - 84.6 * (ty/tw)

The formula produces a score from zero to one-hundred. A lower number means a harder-to-read page, while a higher number means the page is easier to read.

Structural Analysis. HTML provides various means to hierarchically structure pages, including frames, tables and headers. Frame Analysis looks at the nesting of framesets and frame tags to determine their structure. A wire frame drawing tool can then convey a skeleton drawing of the frame layout of a page. Table Hierarchy Analysis examines the different layers of nested tables. A table hierarchy aids the user in learning how a page is organized and how information is nested within different tables. A table of contents presents an outline of the web page using all the header tags contained in the document. They are then presented based on their importance as determined by the level of each tag.

Link Analysis. Link Analysis examines the hyperlinks embedded in a web page. It looks at how many links are internal to the page, external to the page, external to the web site or are *mailto* links. The analysis can also determine which links are broken and characterize the targets of the active links. A graphical presentation of this analysis can show external links having lines leaving a page and internal links pointing to their targets on the page. Broken links can be drawn using a wavy line. A similar analysis examines the different image maps contained on the web page. These types the images embed links that can be examined in the same fashion as textual ones.

Form Analysis. Form Analysis captures the use of HTML forms on a given page. It looks at the different types of inputs and fields employed within a form. Form Analysis also looks at hidden fields. These fields contain important information about what is done to the data provided in the form when the user submits it.

Universality. Universal design of a web page means that it can accommodate users with disabilities. An example of testing for universal design is to see if a web page can support people with eyesight problems or hearing problems. The analysis is performed by evaluating the page with respect to a list of universal design guidelines.

Style Analysis. Style Analysis allows a person viewing a web page to see how the author organized the presentation of the page. The analysis examines the style sheets and HTML style options for relevant information. The World Wide Web Consortium (W3C) recommends that web page developers separate presentation from structure in the composition of web pages. *Presentation* is defined as the attributes of different tags such as color, font size and layout. According to the World Wide Web Consortium, one should put presentation elements into style sheets and avoid using tag attributes for that purpose [11]. Style Analysis looks at the extent to which the page developer has separated structure from presentation. Another type of presentation analysis determines how cluttered a page is based on the amount of white space, text density, image density, and link density occurring in it.

Content Analysis. There are many different types of analyses that can be done on the content of a web page. For example, Statistically Improbable Phrases (SIP) Analysis[2], which determines the “best” phrase that describes the web page for purposes of indexing. The converse operation is to try to characterize a page versus a set of known categories. Examples of categories include on-line store, news site, and course descriptions.

Augmentation Analysis. Augmentation Analysis looks for features of a web page that are not part of HTML. Such features include JavaScript, Java applets, and Flash content. For example, Augmentation Analysis can be used to determine if JavaScript is only employed as a mouse-over feature for links or provides more elaborate functionality.

Metadata Analysis. HTML metadata can include the title of the page, its description, suggested keywords for indexing and the date the page was last modified. Since the metadata can contain any type of

1. http://en.wikipedia.org/wiki/Flesch-Kincaid

2. http://en.wikipedia.org/wiki/Statistically_Improbable_Phrases

data, any extra information the author has included can also be extracted.

Multimedia Analysis. The final type of analysis examines the multimedia content of a web page. For example, the use of animated images and embedded audio and video can be determined.

Table 1 lists the analysis techniques, an estimation of the difficulty of implementing them, an indication of whether the necessary information is available in the page's Document Object Model[3] (DOM) or whether further lexical, syntactic, or semantic analysis is required, and the extent to which we have included the analysis in our Page Monkey tool, to be described in Section 5.

Table 1: Web Page Analyses

Analysis Name	Difficulty	Complexity	Status
Augmentation	Difficult	Semantic	Future
Broken links	Moderate	Lexical	Future
Clutter	Difficult	Syntactic	Future
Domain	Difficult	Semantic	Future
Form	Moderate	Semantic	Complete
Frame	Easy	DOM	Complete
Image map	Moderate	Syntactic	Partial
Link target	Difficult	Lexical	Future
Metadata	Easy	DOM	Complete
Multimedia	Easy	Syntactic	Future
Reading ease	Easy	Syntactic	Future
Reading level	Easy	Syntactic	Future
Statistically improbable phrases	Difficult	Lexical	Future
Style	Moderate	Syntactic	Partial
Table of contents	Moderate	Lexical	Complete
Table hierarchy	Easy	DOM	Complete
Universal design	Difficult	Syntactic	Future

3. Available Technologies

There are two fundamentally different approaches to building a tool for reverse engineering web pages: As a separate application or as a plug-in to a web browser. A separate application offers substantially less restrictions. It does however pose an inconvenience to users because they have to download the web page into the application or remember the URL of the page so that the application can download the content. On the other hand, building a plug-in enables the user to run the tool while looking at the web page. For this reason, we decided to implement our tool, Page Monkey, as a plug-in. Plug-ins, in turn, offer several different implementation approaches. The technologies we researched included Visual C++ extensions to Internet Explorer, the HotJava browser, the NetClue browser and the Greasemonkey add-on to Firefox.

Visual C++ technology provides the ability to make full-featured extensions to Internet Explorer using C++. By "full featured" we mean that it is not limited in its ability to perform tasks that a stand-alone application can perform. Visual C++ gives the programmer DOM level-two interactions with the browser. This means the programmer can read information from the DOM as well as make dynamic additions to the document. Visual C++ extensions contain ample documentation via the MSDN libraries, and examples of use are available on the internet. On the other hand it can be very difficult to find any support resources for this technology that are not provided by Microsoft. Visual C++ has a steep learning curve and is limited to the Internet Explorer browser.

HotJava is a browser that is written entirely in Java. It has the ability to support Java applets and JavaScript. A limitation of the browser is that it contains support for DOM level one interactions, which only allows a programmer to read the contents of the document. Also, the only documentation available for this technology is located on Sun Microsystem's website[4]. The browser does support all operating systems except MacOS. This feature is alluring, but HotJava is no longer actively supported by Sun. The absence of long-term availability, the inability to support MacOS and the lack of DOM level-two support have lead us to choose a different technology for our system.

NetClue is a browser written in Java that is capable of running on multiple devices such as computers, cell phones, and other non-PC devices. It supports JavaScript and Java applets. The browser also contains DOM level-one support. The only documentation available for this technology is on the creator's web site[5]. There are no outside examples of how to interface with this browser. The support for many different devices is impressive and the capabil-

3. See the Appendix for a short description of the DOM levels.

4. http://sunsite.nus.sg/hotjava/hotjava.html
5. http://www.netcluesoft.com/top.php3?cat=base&sub=home

ity to support JavaScript is appealing. But the limited user base and lack of documentation and examples limits its appeal.

Greasemonkey[6] is an add-on to the Mozilla Firefox browser that supports the incorporation of JavaScript into client-side browsers. JavaScript programs have access to the Document Object Model (DOM) representation of the page being visited, thus obviating the need to explicitly perform HTML parsing. Moreover, Greasemonkey supports DOM level 2, enabling the creation and display of new content, such as analysis results [7]. The fact that Greasemonkey is an add-on to the browser enables the development of scripts in a lightweight manner, since the programmer does not have to deal with the interaction between the browser and the script.

There are a many available examples and tutorials for JavaScript that ease the learning curve of this simple language. Also, JavaScript may at some point in the future be added to other browsers in a type of file called a *bookmarklet*, which allows one to activate a JavaScript program through a browser's bookmark. Using JavaScript, an analysis tool can be employed on any operating system and on most popular browsers. Installation of a script into Firefox is straightforward, facilitating testing. The Greasemonkey system also provides a short-cut key-press combination to activate a script. The mix of quick learning and DOM level two support have led us to choose Greasemonkey and JavaScript as the technologies to implement Page Monkey.

4. Design Guidelines

To inform our design, we interviewed an expert web-page designer. The expert has six years experience in designing web pages and web development. Our expert is experienced in layout design, graphic layout, HCI, HTML, and other web page related languages. We asked the expert to tell us about the principles of good page design and what practices lead to bad pages. The expert told us that the number one principle of good web page design is usability. Other design principles include easy navigation and a striking presentation. Good navigation enables users to find what they are looking for with a minimum of effort. A striking presentation is needed to draw attention to a site. Some situations to avoid are non-uniform navigation across pages and non-uniform presentation across pages. Non-uniform navigation and presentation bewilders the user and makes it difficult to navigate through a web site [12].

For the second portion of the interview, we had the expert look at four different web sites and answer questions. The questions concerned layout, information density and navigation. One of the common themes we found in the expert's responses was that a page should be designed to meet the user's expectations. For instance, CNN's web site is tailored for users who wish to view important news going on in the world. The expert noted on all the sites visited that links should be given descriptive titles so that the user can readily determine if they wish to follow that link. One key factor is that a page should provide easy access to important information or capabilities. We also had the expert visit Newegg.com, which is an online retail store, and the expert mentioned how important it was to have the search capability readily available. The expert found a few bad design choices in the sites that were examined. One bad design concerned clutter: Some of the pages contained too many third-party advertisements that took up valuable screen real estate better filled with user content.

The interview served two purposes. Not only did it guide us in the direction of making our tool usable, but it also provided a basis for the analyses we included. That is, pages that are highly rated by an analysis ought also to conform to expert guidelines.

5. Page Monkey

Page Monkey is a tool that a visitor can use to select different analyses to perform on a web page. Page Monkey can show properties of nested structures, such as frames and tables, style aspects, and active features such as links and scripts. Input is obtained from the DOM and accessed through various APIs. After getting access to the HTML structure, Page Monkey needs to infer higher level abstractions. The purpose of our research is to determine what it means to reverse engineer a web page: specifically which analyses are feasible and useful.

5.1 System Architecture

Page Monkey is implemented in JavaScript installed as a plug-in to Firefox via Greasemonkey. It is composed of three major components: the DOM, the analysis engine and the user interface. The DOM is important to the system because it holds all the document content for analysis. The analysis engine contains the code for all of the different analyses that Page Monkey can perform. The user interface enables the user to select the various analysis techniques and display the results in new pop-up windows. When the user selects an analysis to perform, the user interface queries the DOM for the specific

6. http://greasemonkey.mozdev.org/

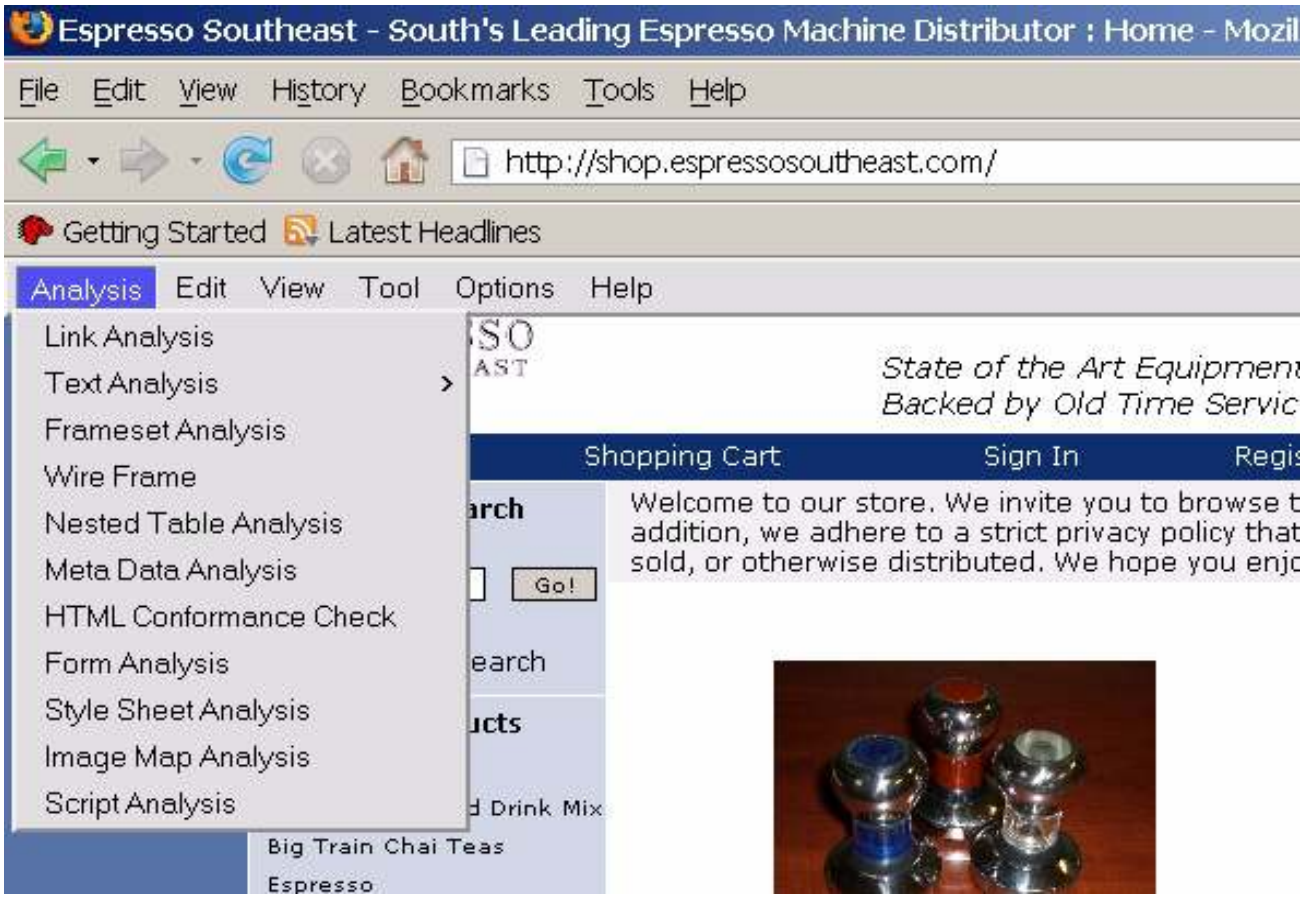


Figure 1. Picture of User Interface

information and then parses it accordingly to get the results, which are then written to the output window.

The system contains few platform dependencies. The fact that Page Monkey is written in JavaScript enables it to be run on any browser that supports JavaScript functionality. There are many different browsers across different operating systems and devices that contain support for JavaScript. The integration into the browser provides a major convenience for the user. The user can now analyze a web page without having to open a separate application and save the web page to it for study. Page Monkey also provides access to the W3C's HTML conformance checking tool to determine if a page is compliant with the HTML 4.01 standard.

Page Monkey's JavaScript code follows the recommended style for writing Greasemonkey scripts[7]. The script contains four distinct sections. The first section is the system initialization, which adds the Page Monkey menu to the DOM of the web page and sets the appearance attributes for the menu. The second section is responsible for menu items. The third section contains the analysis engine of Page Monkey. Each analysis function's signature is named to indicate the type of analysis it performs. Within each analysis function, a script element is created to be appended to the DOM of the web page. This script element contains the actual code that performs the analysis. The final section of the script contains the system activation code that tells Greasemonkey to initialize our system and begin accepting commands from the user.

To use Page Monkey, both Greasemonkey and Page Monkey must be installed. When visiting an interesting page, the user can then select an analysis technique from the menu that appears above the web page as shown in Figure 1. For example, suppose a user wants to perform Metadata Analysis on the page currently being viewed. In order to view its Metadata, the user displays the Analysis menu, and a list of analysis techniques appears. The user then selects the "Metadata Analysis" menu entry. The sys-

7. http://greasemonkey.mozdev.org/authoring.html

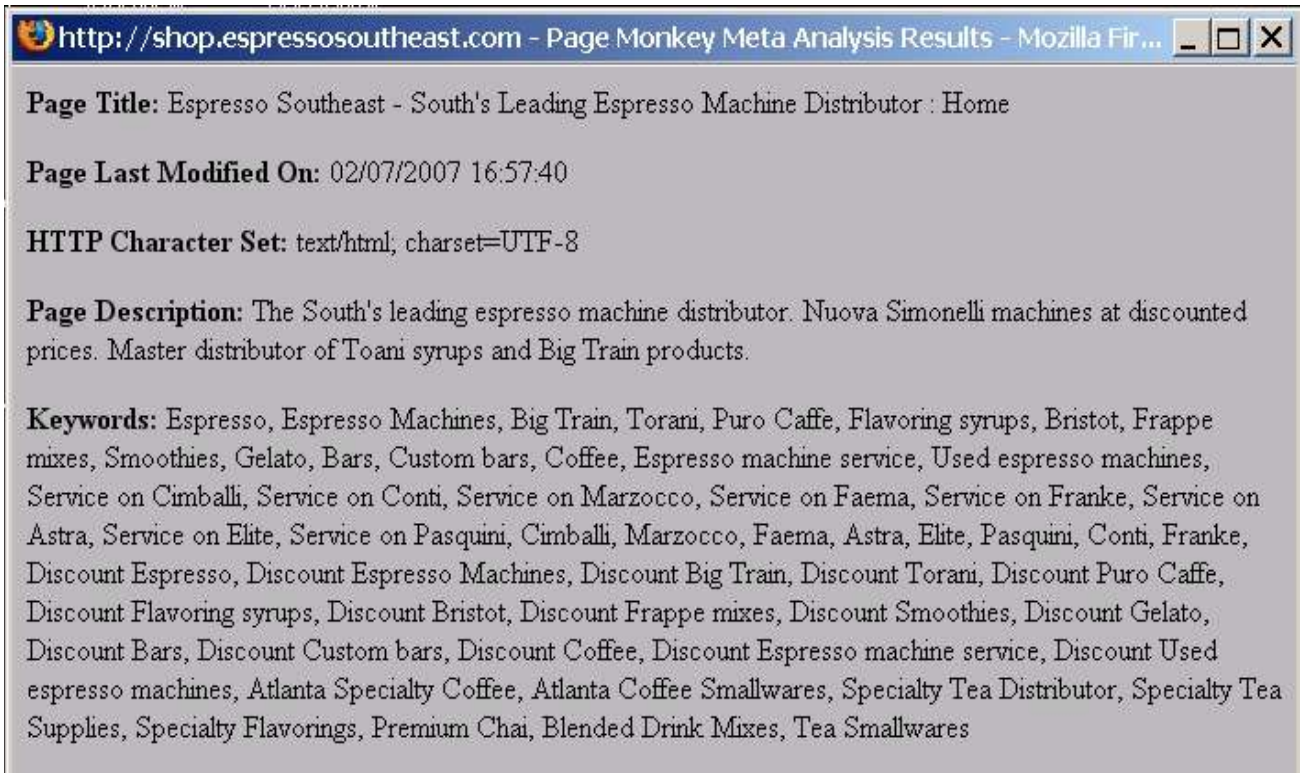


Figure 2. Picture of Metadata Analysis Result Screen

tem accesses the DOM and formats the results for display, as shown in Figure 2. Users of Page Monkey can close the results window when they are finished looking at the presented data, or they can copy its content into another document for purposes of reuse or documentation.

5.2 Implementation Limitations

JavaScript contains limitations that make it difficult to implement a fully featured system. For example, it only provides a simple drawing interface for creating images. We found only one library for drawing simple figures such as lines, rectangles, circles, and sine curves [14]. This limited capability makes it difficult to express the results of analyses diagrammatically, for example, in the form of a tree or a set of contained boxes.

Another major drawback we found is JavaScript's inability to perform file input and output operations. This hinders Page Monkey's capability to store data in files and load data from previously saved files. The lack of file I/O in JavaScript is due to the inherent security risk associated with allowing a script to access a user's hard drive. JavaScript also does not have the ability to create network sockets for communication with other machines. This prevents us from making additions to the system that can transfer data over the network.

6. User Feedback

To perform an informal user evaluation, we tested our system on some students from the Georgia Institute of Technology. First, we gave the subject a brief introduction to the topic of our research. We then guided them through an example use of the system to reverse engineer Delta Airline's home page. We allowed them to try any analysis they wished, with the exception that we specifically asked them to perform a form analysis of Delta's home page. After they were done interacting with the Page Monkey we asked them the following questions.

- "What aspects of the system did you find useful?"
- "What improvements can we make to Page Monkey?"
- "Are the results of the Form Analysis understandable to you and easier to understand than if you had to read the web page source code?"

Results of User Evaluation. Overall the evaluators felt that Page Monkey presented the analysis results in an understandable way. All of them agreed that Page Monkey provides better feedback than looking

at the source code of the web page. The evaluators provided us with many ideas for improving our system. They recommended that we make the results of an analysis more interactive for the user. For example, one evaluator suggested that we use a tree menu to present form components so that the user could expand the menu to see detailed information. For Link Analysis, one evaluator recommended that we display the email addresses associated with a mailto link and also present the destination of links for the user to examine.

7. Related Work

In the process of planning the functionality of Page Monkey, we investigated other approaches to the reverse engineering of web pages. During our research process, we were unable to find a system that provided all we had in mind. The systems presented below deal with specific aspects of reverse engineering of an online document or a whole web site. Nevertheless, some of the information provided by these systems is useful to us. In addition to other systems, we looked at web page design guidelines as a source of evaluation criteria.

Reverse Engineering Systems. Estievenart, Francois, Henrard, and Hainaut present a methodology to extract data from static HTML pages and store them in a database [4]. Their process includes the extraction of a schema and data from the page. The process consists of a classification, cleaning, semantic enrichment, extraction, and conceptualization. The overall goal of their proposed system is to enable the translation of static HTML pages into dynamic pages that access a database. Page Monkey differs in that our system reverse engineers a page to present higher level information to the user. Also, the authors' system deals with whole web sites and does not give detailed information about individual pages.

Vanderdonckt, Bouillon and colleagues developed a system called Vaquista to examine the presentation model of a web page [10][1]. They use this information to migrate the page's presentation model to another environment. Vaquista performs a static analysis on web pages when given a file or a URL. Page Monkey differs from Vaquista in that we perform an on-demand analysis while the user is looking at the web page in the browser. Our system also performs more types of analysis. Page Monkey perForms Analysis on presentation, style, structure, and content of a web page while Vaquista only examines the presentation model.

Carchiolo, Longheu, and Malgeri analyze the structure and semantics of pages and sites for the purpose of highlighting their organization [3]. They build a logical schema of the structural and semantic organization by reflecting the author's intentions and quality criteria used during development. Their system views the web page as a tree of nested HTML tags. They locate specific tags that are then labeled as primary tags if they describe structural or logical information. They organize their gathered data into tree structures for presentation. Page Monkey differs from this system in that its analysis is performed on-demand as compared with Carchiolo *et al*.'s static analysis approach. Page Monkey also deals with the Document Object Model of a page compared to actually parsing the page's HTML. This provides us with a higher level of abstraction than the raw tags of the page and gives us a predefined organizational structure. Furthermore, Page Monkey provides a wider variety of analysis techniques that this proposed system does not contain.

Web Design Guidelines. Hymes and Olson present a process for organizing the content of a web page for the user's needs [5]. They say the most difficult problem for the user is to understand how the content is organized within a site. Their techniques help a designer represent a model of the user's conception of the content. We believe this to be an important approach because it suggests that we analyze the content of a web page to see if it effectively organizes data to represent a user's mental model. This type of analysis will also allow our system to categorize a web site to determine the intended audience of its page.

Calongne [2] argues that one can use HCI usability techniques to help design a good web site. The goal is to determine the identified target audiences and partition it into classes of users. She also identifies what type of web site it will be and determines the content and any constraints or boundaries to the content. The author says one should postulate the content going into each page to be sure it provides a good reason for the page to exist. Page Monkey employs these arguments when analyzing the content of a web page. Furthermore, Page Monkey checks for HCI usability requirements such as Universal Design.

Zimmermann argues that information design principles can be effectively applied to the creation of web pages [13]. The author supports her thesis with Edward Tufte's general principles of information design [9]. She claims that most web page design advice is based on the page style that is popular at a certain time. This does not provide a consistent methodology for designing web pages. The author's

approach includes five different information design principles that can be applied to web page design. We analyzed her recommendations for incorporation into Page Monkey.

8. Status and Future Work

Page Monkey is a working tool that can be downloaded from our web site[8]. Future work will include the addition of more analysis techniques. Currently the system only provides the lowest level abstraction of the data. Along with these additions, the system will need to have the capability to move up and down between these layers of abstraction. Also, a HELP system needs to be added.

JavaScript has the ability to catch mouse and keyboard input from the user. We would like to use this ability to enable the user to highlight a certain area of a web page and display data about it. Also, shortcut keys need to be added for each of the analysis techniques so that experts can submit queries to Page Monkey without having to go through the menu.

We plan to augment several of the analysis techniques. The analyses that need to be enhanced are Image Map Analysis, Link Analysis, and Style Analysis. Additions to the Image Map Analysis include displaying which image the map is intended for. An idea we have for a different presentation of the image map is to section the areas on the image that correspond to the links in order to provide a visual presentation of the data. We plan to add destination information to links detected during Link Analysis and an indication of which links are broken. Furthermore, the presentation model we intend to make the results of the Link Analysis more interactive for the user. Currently it just prints all information to the results window. One possibility for an improvement was found during our informal user evaluation. The evaluator suggested that the system could represent the link information in a tree menu and allow the user to expand the results to see more detail. Currently, Style Analysis looks at the style sheets that are in external files. It needs to be updated to also handle the style tags in the <HEAD> section of the web page. Additionally, Style Analysis needs to evaluate the extent to which structure has been separated from presentation on the page.

Once the system contains these extensions, we intend to perform a more extensive usability evaluation, by having users perform pre-defined tasks and make note of what they find difficult. We should make sure our HELP documentation is understandable to the beginning user and that it provides tips to intermediate and expert users.

Our ultimate vision for Page Monkey is not just to support the reverse engineering of web pages but to facilitate their selective reuse. In particular, we want to enable the visitor to select a segment of a page and have the ability to save any aspect of the segment, such as its style, content or structure. Clearly, Page Monkey would have to determine exactly what *segment* means in a given circumstance and how much context information must be included with the selected element.

9. Discussion

Based on our literature review and our experience with Page Monkey, we can make the following comparisons of web page reverse engineering and software reverse engineering. In particular, we can distinguish the two on both intrinsic and extrinsic grounds. By *intrinsic* we mean the techniques used to reverse engineer in the two domains. An extrinsic comparison, on the other hand, considers the uses for which the analyses are performed.

Intrinsic Comparison. Software reverse engineering technology includes both static and dynamic analyses. Static analyses can, in turn, be broken down into those that are primarily structural—based on the programming language syntax or flow properties—and those that examine a program's semantic intent. Web page reverse engineering clearly involves structural analysis. Page Monkey includes analysis of headers, table nesting and frames. Likewise, Link Analysis is analogous to program flow analysis. Semantic analysis, such as that based on a program's application domain [8], is more interesting. With software, although analysts can mine comments and mnemonic variable names, they must mostly rely on documentation external to the software to gain understanding of why certain features are included in a program. For web pages, there is a higher ratio of content available on the page itself. Although analyzing text is difficult, machine learning technology will continue to make progress.

This paper has intentionally not examined the question of how to analyze a set of related web pages. For example, there is already much interest in better understanding the experience of a user when visiting a web site to make a purchase. A whole industry, web analytics, exists for this purpose. Roughly speaking, this sort of analysis corresponds to the dynamic analysis of software reverse engineering.

8. http://www.cc.gatech.edu/projects/PageSleuth/

One further intrinsic issue should be discussed: How the nature of complexity differs between software and web pages. HTML 4.01 uses 91 kinds of elements and 120 attributes (not counting multiple uses of the same attribute name in different elements). In comparison, Java 6 comprises 113 syntax productions and 50 keywords. In this sense, the language complexity of the two is similar. However, the semantics of the languages differ greatly. Web page processing was designed to be stateless, whereas languages like Java make heavy use of state. On the other hand, the presentation of programs is of much less importance than the presentation of web pages, and, as stated above, there is much more textual content to be analyzed semantically on a web page than there is in a typical program. Finally, it should be noted that web pages can include executable scripts, raising all of the problems of software reverse engineering in the web page context.

Extrinsic Comparison. Software is ubiquitous and has become an essential part of our economy. Software development costs are dominated by maintenance activities which include significant program understanding. Likewise, web pages are also everywhere. One important difference between the two is in the authors. Although there is increasing interest in end-user programming, most software development efforts are conducted by highly trained programmers. Many web pages, in contrast are written by amateurs. This is a double-edged sword. It means that a given page may have been written in a suboptimal way, making its comprehension all the more difficult. It also implies that there is a greater need by the amateurs to look at and comprehend examples.

There are a variety of reasons why software maintenance is undertaken: to add new features, to adapt to new technology, to fix problems and to detect and excise any malicious insertions. All of these reasons have analogs that can motivate web page reverse engineering. For example, web pages are one way that businesses and other organizations present themselves to the public. That is, they constitute a major part of how these organizations market themselves. Any change in marketing approach, whether the introduction of a product, the announcement of a sale or the implementation of a reorganization must all be reflected on the web. Hence, there is a constant need to change the organization's web presence.

Like companies, the web itself is always changing. New technologies are constantly being introduced, whether for use directly within web pages, such as a new W3C standard; because of new client-side technology, such as Greasemonkey, or when new computing technology becomes available, such as the introduction of a new operating system. Consequently, there is a never ending need to update web pages in order to adapt to or take advantage of such changes.

Like programs, web pages may contain bugs. Whereas some software bugs can render a program's output entirely worthless, many are, in fact, intentionally ignored because they are relatively unlikely to occur or because the cost to fix them is too high. Web pages, in contrast, are inexpensive to fix, but even trivial bugs like misspellings can undermine a company's image. Hence, there is a need for web page debugging similar to that for software.

In the past few years, the need for *software forensics,* the detection and elimination of malicious code, has become a significant driver for maintenance and reverse engineering. Analogous problems, such as web bugs and phishing attacks, hold with web pages. Hence, we see an increasing need for forensic-driven reverse engineering of web pages.

Programs are read not only by the people maintaining them but also by competitors trying to leverage knowledge of a program. For example, computer game designers have invested large amounts of effort to reverse engineer game engine software in order to produce games that compete with those provided by the original vendor of the engine. Similarly, companies have been known to examine competitors' web sites in order to learn about new products, reduced prices or marketing approaches. Web page reverse engineering can facilitate this kind of analysis.

Finally, one other comparison should be made between software and web pages. This concerns usability. As computers become more available, untrained users are more likely to encounter them, not only in developed countries, but throughout the world. We are still learning how to promote these interactions. To do so, we need to know much more about how technology features effect user experience. This implies instrumentation and analysis, both of which may involve reverse engineering. For example, some pages, such as Google's, feature a minimalistic presence, while others, such as CNN, are quite busy. Which approach is more effective? Or do the differences in the purpose of these two sites necessitate different page style?. Much of the literature insists that we should apply good usability principles to the designs of our web sites. We believe the checking of these principles by analysis tools could help establish the importance of these principles.

That is, analysis tools can provide normative pressure to improve pages. Consequently we see an important future role for the reverse engineering of web pages.

References

[1] Laurent Bouillon, Quentin Limbourg, Jean Vanderdonckt, and Benjamin Michotte. “Reverse Engineering of Web Pages Based on Derivations and Transformations.” 07 August 2005. 27 August 2005.

[2] Cynthia M. Calongne. “Designing for Web Site Usability.” *The Journal of Computing in Small Colleges,* 16(3):39-45, March 2001.

[3] V. Carchiolo, A. Longheu, and M. Malgeri. “Improving Web Usability by Categorizing Information.” *Proceedings of the IEEE/WIC International Conference on Web Intelligence,* October 13-17, 2003, pp. 146 - 152.

[4] Fabrice Estievenart, Aurore Francois, Jean Henrard, and Jean-Luc Hainaut. “A Tool-Supported Method to Extract Data and Schema from Web Sites.” Proceedings of the Fifth Workshop on Web Site Evolution, 00(3), 2003.

[5] Charles M. Hymes and Gary M. Olson. “Quick but not so Dirty Web Design: Applying Empirical Conceptual Clustering Techniques to Organise Hypertext Content.” *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques,* pp.159-162, August 18-20, 1997, Amsterdam, The Netherlands.

[6] Melody Y. Ivory, Rashmi R. Sinha, and Marti A. Hearst. “Empirically Validated Web Page Design metrics.” *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* pp.53-60, March 2001, Seattle, Washington, United States.

[7] Mozilla.org. “What Does Each DOM Level Bring?” July 10, 2005. http://www.mozilla.org/docs/dom/reference/levels.html.

[8] Spencer Rugaber. “The Use of Domain Knowledge in Program Understanding.” Annals of Software Engineering, 9:143-192, 2000.

[9] Edward R. Tufte. The Visual Display of Quantitative Information. Graphics Press, 1983.

[10] Jean Vanderdonckt, Laurent Bouillon, and Nathalie Souchon. “Flexible Reverse Engineering of Web Pages with VAQUISTA.” *Proceedings of the Eighth Working Conference on Reverse Engineering (WCRE'01),* IEEE Computer Society, Washington, DC, USA, 2001, pp. 241.

[11] World Wide Web Consortium. “HTML 4.01 Specification.” December 24, 1999. http://www.w3.org/TR/REC-html40/.

[12] World Wide Web Consortium. “Web Content Accessibility Guide.” November 23, 2005, http://www.w3.org/TR/WCAG20/.

[13] Beverly B. Zimmermann. “Applying Tufte's Principles of Information Design to Creating Effective Web Sites.” *Proceedings of the 15th Annual International Conference on Computer Documentation,* ACM Press New York, NY, USA, 1997, pp. 309-317.

[14] Walter Zorn. “DHTML: Draw Line, Ellipse, Oval, Circle, Polyline, Polygon, Triangle with JavaScript.” October 24, 2005, http://www.walterzorn.com/jsgraphics/jsgraphics_e.htm.

10. Appendix - DOM Levels

The W3C Document Object Model contains level specifications that provide different capabilities for manipulating a document [11]. Currently there are three levels of the DOM. Level one provides a low-level set of fundamental interfaces that can represent any HTML document. Level two provides extra enhancements not present in level one. Additions include the capability of programs to dynamically access and update the content of a document. getElementByID and getElementByTagName methods have been added to provide easier access to the document. A generic event system for programs, access to style sheet information, and the ability to dynamically access and update the content of HTML documents have all been provided. Level three adds even more options. Additions include the ability for a program to dynamically load and save document information. A validation component has been added to allow programs to update content and still be valid by HTML specification standards. Also, an extension of the event handler allows one to focus on keyboard input.