Development and Maintenance of aWeb Site for a Bachelor Program

Dr J Rajaram Professor Ph.D ----CSE
E Ravi Assistant Professor M.Tech—CSE
D Navya,Assistant Professor
Donda Prashanth,Assistant Professor
Nagole institute of engineering and technology

Abstract

This paper describes our experiences with the development and maintenance of the BSENG Web site, which providesinformation about a new Bachelor program for softwareengineering at the University of Victoria. The site'srequirements lead to a design that emphasizes simplicity toease maintenance. We used Macromedia Dreamweaver todevelop the site. During development, we identified twoimportant maintenance tasks—detection of dead links andpreservation of the site's navigational structure—that arenot well supported by Dreamweaver. We discuss how weuse the Rigi reverse engineering environment to aid the Website developer with these maintenance tasks.

1. Motivation

Nowadays, many Web sites are highly complex softwaresystems [17]. At the same time, Web sites are oftenimportant assets for an organization. In response tothis need, many professional development tools that supportthe building of sophisticated Web sites have emerged(e.g., Macromedia's Dreamweaver, IBM's WebSphere, andAdobe's GoLive). However, despite tool support, the currentpractice of Web site development is still immature anddoes not adhere to software engineering principles [18]. Forexample, often there is no design or system documentationavailable [22].The first release of a Web site is only the beginning ofits lifetime. A Web site has to constantly evolve in orderto remain useful for its users. Because Web site maintenanceis costly and continuous, Web site developers have toemploy tools and techniques that also assist them in theirmaintenance tasks.The Web site's source code (typical artifacts are HTMLpages, client-side and server-side scripts, and configurationfiles) is the most reliable and detailed information availavailableto the developer. But often higher-level abstractionsare equally important to facilitate the understanding of theWeb site for development and maintenance activities; reverseengineering is concerned with the generation of suchdocumentation. Development tools often focus on the forwardengineering aspect of software development, but neglectthe reverse engineering aspect.In this paper, we illustrate with a case study how we usedreverse engineering technology in addition to a commercialWeb development tool in order to improve the developmentand maintenance of a Web site. Our case study is BSENG,a Web site for a new Bachelor program for software engineeringat the University of Victoria. We decided to buildBSENG with Macromedia Dreamweaver since it is a mature,commercial tool that fits well into our environment.However, during the development of the BSENG Web site,we identified two important maintenance tasks—detectionof dead links and preservation of the site's navigationalstructure—that are not well supported by Dreamweaver. Wecustomized the Rigi reverse engineering environment [25][24] to visualize dead links as well as BSENG's (navigational)structure. The documentation produced with Rigi iscomplementary to the information that Dreamweaver providesand gives the BSENG developer valuable, additionalinformation for recurring maintenance tasks.The paper is organized as follows: The next section identifiesthree different views of a Web site and their relationships.Section 3 gives more background on our case study,BSENG. Section 4 discusses the problem of dead links andhow we customized Rigi to visualize them. Section 5 showshow Rigi visualizes Web site structure and Section 6 closeswith some observations and future research directions

.2.Web Site Views

For both development and maintenance phases, it is importantto distinguish between the different views of a Website. We discern the following views:**client view:** The view of the Web site that a client (typicallyusing a Web browser) sees.**deployment/server view:** The view of the Web site that aWeb server (accessing the local file system) sees.**developer view:** The view of the Web site that a developer(using a Web development tool such as Dreamweaver)sees.Reverse engineering tools can operate on any of theabove views. The reverse engineering process starts withfact extraction from one (or several) views. Extracted factsare typically stored in a repository, which is then queriedby analyses. Analysis results are then visualized to assist indevelopment and maintenance activities.Depending on the view, fact extraction uses different extractionstrategies and extracts different artifacts. In the following,we discuss each view in more detail and give examplesof reverse engineering tools and extractors. Traditionalstand-alone extractors for Web sites work either onthe client view or on the deployment view

.2.1. Client View

Client-side extraction does not require direct access to aWeb site's sources, but the Web server has to be treated asa black box; only its output (i.e., served Web pages) can beobserved.Examples of client-view extractors are essentially Webcrawlers or spiders. These extractors request pages viaURLs, communicating with the Web server via HTTP. Forexample, the strategy of SiteSeer—a tool to collect Webmetrics—is typical for client-view extraction [29]:"Beginning at a known URL, SiteSeer spawns achild process to parse the html document whichwill be found there. Each link found within thehtml text will be added to a links file producedby the child, and ultimately returned to the parent.The parent will spawn a new child for each ofthese new links, so that the whole hyperdocumentis eventually examined. . . . This search, whichis initially exponentially increasing, is limited bythe 'edge' of the website; that is, html documentsoutside the initial website are not parsed, althoughthe links are followed to ensure the URLs aregood."SiteSeer's parser is implemented with Lex and Yacc. Similarly,Perlbot is a web crawler written in Perl that gathersmetrics for a Web site to help understand its evolution [12].Ricca and Tonella have developed the ReWeb tool,which consists of an extractor, analyzer, and viewer forstatic Web sites [19] [20]. The extractor, written in Java,downloads all pages of a certain Web site starting from agiven URL. Links in pages that point outside the Web siteare ignored. The extracted Web site is represented as atyped, directed graph. The ReWeb tool has several analysesthat operate upon the graph structure. Most of these areinspired by traditional compiler (flow) analyses and mappedto the Web site domain (e.g., dominators of a page, shortestpath from the start page to a page, and strongly connectedcomponents). Results of analyses are visualized with Dotty[16] (a customizable graph editor).**2.2. Deployment View**Most reverse engineering research has targeted the clientview, but more deployment-view extractors are emerging.A deployment-view extractor has access to the Web site'ssources (such as HTML pages, CGI scripts, Java SeverPages, and configuration files).Hassan has developed coarse-grained deployment-viewextractors for HTML, JavaScript, VBScript, SQL databaseaccess, and Windows binaries [9]. During the extractionprocess, each file in the directory tree that contains the Website is traversed and depending on the file type the correspondingextractor is invoked. Extracted facts are representedin the Tuple Attribute (TA) format [10]. All extractoroutput is consolidated into a single file and visualized withthe Portable Bookshelf system [7] for the purpose of Website architecture recovery.Di Lucca et al. describe a reverse engineering processforWeb sites that employs both static and dynamic analysis[3] [4] [5]. For the static analysis, facts are extracted fromHTML pages, client-side scripting languages (JavaScriptand VBScript), server-side scripting languages (ASP andPHP), and the directory structure. These facts are representedin a proprietary, XML-based format called IRF,which is translated to a relational database (comprising aschema with 74 tables). Analyses use SQL on the relationaldatabase to retrieve facts. Several graph drawing tools(Rigi, VCG, and Dotty) are used for visualization

.2.3. Developer View

To our knowledge, neither extractors nor analyses for thedeveloper view have been developed by the reverse engineeringcommunity. Development tools expose the developerview to the Web site developer, but they are geared towardsforward engineering as opposed to reverse engineeringfunctionality needed for maintenance activities.A study of three commercial tools (FrontPage 2000,Dreamweaver UltraDev 4, and SmartSite 3) conducted byTilley and Huang in 2001 came to the conclusion that alltools had limited capabilities with regard to supporting reverseengineering activities [23]. Maintenance activitiesthat are supported by these tools are, for example, validationProceedings

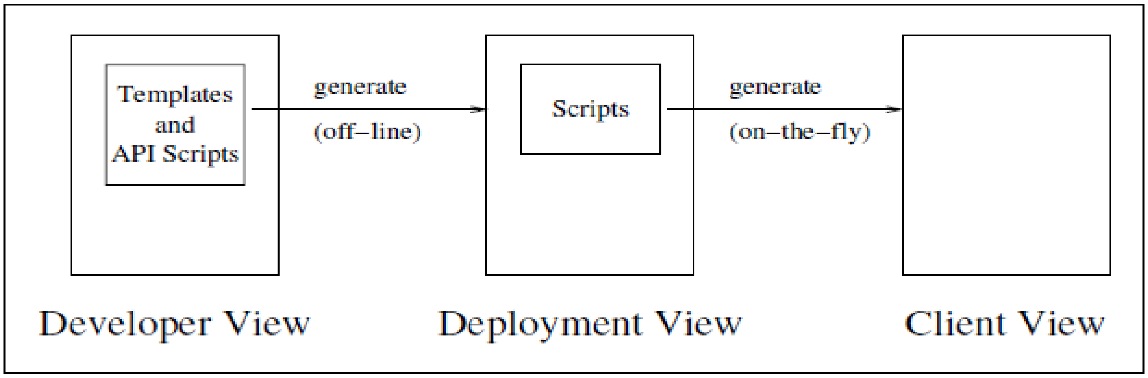

Figure 1. Generative aspect of Web site views

of HTML and XML documents, reports of usage-violationsof ALT and META tags, link checking, metrics that summarizecharacteristics of Web pages, and page download-timeestimates.Fortunately, Web site development tools such asDreamweaver and GoLive have now scripting capabilitiesand expose tool functionalities with a programmable API.These *API scripts* can be used to automate recurring developmentactivities, but also make it possible to implementdedicated reverse engineering functionality

.2.4. Relationship of Views

All of the three views introduced above are of potentialinterest to the Web site maintainer. For example, the developerview shows the high-level Web design such as informationabout templates; the deployment view is the one theWeb server uses and thus important for server maintenance;finally, the client view is the one that the user sees and thusis important to assess navigability and structure of the siteas well as to detect dead links. The maintainers have to takeinto account the characteristics of the view that they workwith.Analyses that rely on facts extracted from the client ordeployment view can give misleading information to themaintainer of a Web site. An example is clone detectionof HTML and embedded scripting [11]. If Web pages aregenerated from templates (which is supported, for example,by GoLive, Dreamweaver, and certain Wikis), a clonedetection analysis (operating on the deployment or clientview) will report a large amount of cloned material.[1] Theseclones, however, are of no concern to the maintainers, becausethey are working in the developer view, which exposesthe templates to them.[1]Dreamweaver can export HTML comments that function as meta-datato identify templates in the generated HTML page. A dedicated clonedetection algorithm could make use of this information to suppress clonescaused by templates. However, the exporting of meta-data information istypically didisabled to optimize bandwidth in the deployed version of theWeb site.Many of the differences in the views are caused by generativetechniques that connect the views. Mechanisms suchas templates at the developer view drive the generation oftarget code (often off-line) for the deployment view. Dynamic,server-side technology (such as JSP and servlets)in turn generates on-the-fly target code for the client view.Another example of differences between deployment andclient view are the potentially complex mappings fromURLs to file-system paths that are dynamically performedby the Web server (governed by the server's configurationfiles). These relationships between the views are depictedin Figure 1.Little attention has been paid so far to the particular reverseengineering problem caused by systems that have agenerative component; this is the case for Web sites as wellas traditional software systems. It is now common to findWeb sites that employ generative techniques. Traditionalsoftware also uses generators such as compiler compilers(e.g., Yacc), embedded languages (e.g., ESQL in C), ordomain-specific languages [27]. To give an example, Deanand Chen describe the design recovery for a traditional softwaresystem consisting of a textual domain-specific language(S/SL) with a generator that produces PT Pascal code[2].In order to analyze such systems, the reverse engineeringtool must be targeted to a specific generator. Ideally,an analysis should identify mappings from pre-generationartifacts to post-generation ones and vice versa.**3. BSENGWeb Site**The Bachelor of Software Engineering (BSENG) is anew interdisciplinary program offered by the Faculty of Engineeringat the University of Victoria (UVic). The programis scheduled to start in autumn 2003 with an initial enrollmentcapacity of 75 students per term. One of the authorsof this paper was in charge of the initial development of theBSENG site.The BSENG Web site is located at the domain `http://www.cs.uvic.ca` (see Figure 2).[2] The BSENG Website is a recruitment tool and information source for potentialapplicants to the program. While populating the sitewith material, it became clear that the site is also an importantresource for members of the faculty and committeesinvolved in the program. The site currently provides informationabout the curriculum, course structure, programrequirements, application procedures, contacts, as well aspromotional and program-related download material (e.g.,class room slides, presentations, and program proposals).As the BSENG program grows and matures, theWeb site isexpected to evolve accordingly.

Figure 2. Screenshot of BSENG home page

home page The BSENG site has high textual information content, which will change rather infrequently once the BSENG programis established. However, new information will beadded to the Web site continuously. In order to simplifythe process of adding new information, we decided on alist structure for each page with a table of contents on topfrom which the user can jump to the detailed information onthe page. Specifically, with about 50 courses and the limitedtypes of changes, we decided that there is no need fordatabase-driven dynamic Web pages.2This paper describes the first version of the BSENG site. Since theacceptance of the paper, a new version has been published on June 19,2003.We further decided on an easy-to-navigate and fastloadingdesign that is accessible for a wide variety ofbrowsers. The latter is of special importance in academicenvironments, which typically exhibit a heterogeneous infrastructure.Another important aspect of keeping the design simplerefers to maintenance and costs. In research and educationcommunities turnaround times of Web site administratorsare short. We also took the following considerations intoaccount:• On the user side, Nielsen observes that "users haveless patience for bleeding-edge technology these daysas the Web gets dominated by later adopters and theupgrade speeds for new browsers and plug-ins slowdown" [15].• On the developer side, Schmeiser advises to subjectany Web technology to a rigorous screening beforemaking a decision to adopt [21]. Questions to ask are,for example, "Will this solution save me resources?"and "Will this technology be time-intensive to extractfrom my system if I decide to replace it?"As a consequence, an overriding principle for the BSENGWeb design was the KISS principle: Keep It Simple, Stupid.Despite the simple design outlined above, we aimed for aprofessional, colorful design that might appeal to students,who have come to expect Web sites that exhibit a high professionalismand quality

.**3.1. Site Development**There is a staggering variety of technologies, standards,and tools to choose from when developing aWeb site. OftenWeb sites try to employ the latest cutting-edge techniqueswithout regards to the site's clientele as well as developmentand future maintenance implications. According toour considerations for maintenance outlined in Section 3.2,we consciously tried to avoid advanced features of the toolin favor of simpler, proven technologies.The BSENG site is static, which means that pages arecomputed at application definition time and remain immutableduring application usage [8]. This is in contrastto dynamic sites where pages are computed on-the-fly andcan change during application usage. Our site is Level 0(i.e., "only HTML pages without frames") according to theclassification introduced by Ricca and Tonella [19]. The siteuses plain HTML 4.01 without frames. The page layout iscontrolled with nested tables.We exploit cascading style sheets (CSS) for accessibilitypurposes. They are mainly used to set font sizes and colorswithout affecting the navigation of the Web site. In the firstversion, we did not use client-side scripts, but we will usethese for future versions, for example, to adapt the layoutdepending on the detected browser type.The BSENG site has been developed with MacromediaDreamweaverMX , an integrated development environmentfor Web site development. Dreamweaver is available forboth Windows and Mac, which is an important considerationin our university environmenz where the site is beingmaintained.

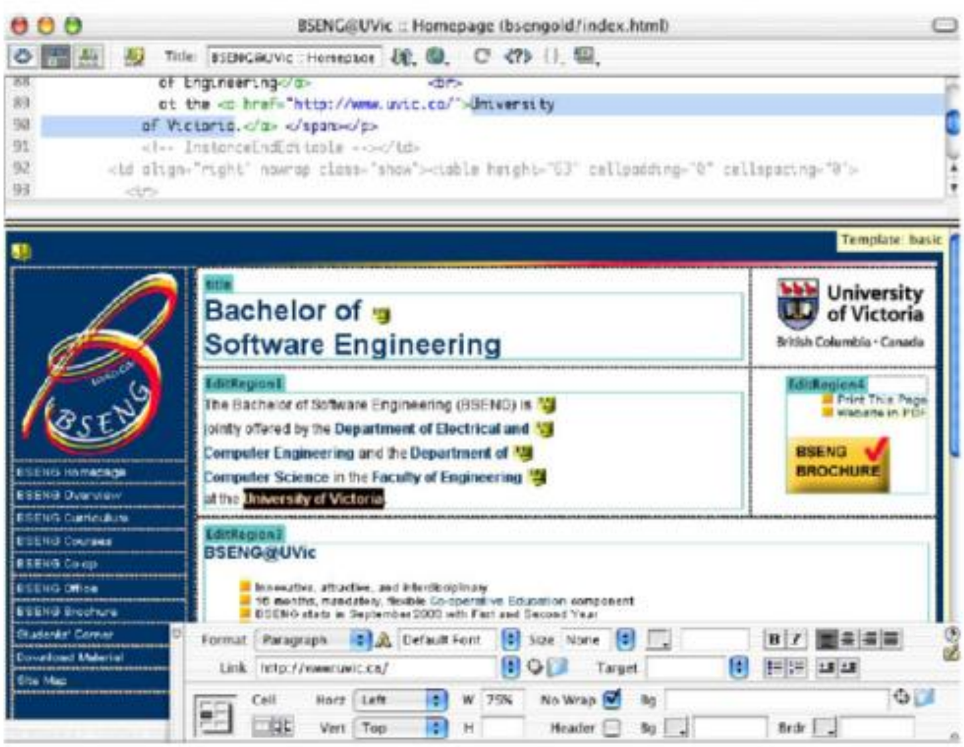

Figure 3. Developer views of the BSENG site

in Macromedia Dreamweave

Dreamweaver supports multiple editing views for a page.The WYSIWYG view (cf. large window of Figure 3) allowsediting of the page without an in-depth knowledge ofHTML. If a page is based on a template, only certain partsof the page can be edited. The code view (cf. top windowof Figure 3) shows the deployment view of the page (optionallypretty-printed). Similar to the WYSIWYG view,code that defines the template cannot be modified. Eitherview can be edited during development and both views aresynchronized.An advanced feature of Dreamweaver that we decidedDreamweaver supports multiple editing views for a page.The WYSIWYG view (cf. large window of Figure 3) allowsediting of the page without an in-depth knowledge ofHTML. If a page is based on a template, only certain partsof the page can be edited. The code view (cf. top windowof Figure 3) shows the deployment view of the page (optionallypretty-printed). Similar to the WYSIWYG view,code that defines the template cannot be modified. Eitherview can be edited during development and both views aresynchronized.An advanced feature of Dreamweaver that we decidedoff-line with an API script (cf. Figure 1) that is invoked bythe developer before deploying the site.

total files	98
HTML files	28
image files	9
all links	1968
external links	345
broken links	294
orphaned links	29

Figure 4. Statistics of the BSENG

siteFigure 4 shows some statistical information for theBSENG site taken from Dreamweaver's developer view.The site has only 28 pages (half of which are pages for theprint view), but a rather large number of links

3.2. Site Maintenance

The maintenance aspect of aWeb site must be taken intoaccount right from the start of theWeb site development effort.In fact, the maintenance aspect of Web sites is oftenneglected. Nielsen lists "Forgetting to Budget for Maintenance"among his *Top Ten Mistakes of Web Management*[14]. The choice of technologies and tools has a significantimpact on the future maintenance effort.For the BSENG site we chose to simplify maintenanceby employing a small set of simple, well-established technologies.We consciously avoided approaches that have anegative impact on maintenance; specifically:• No advanced HTML features such as frames andclient-side image maps.These are often hard to understand for maintainers wholack a Web development background and can causeproblems when migrating to anotherWeb developmenttool.• No JavaScript for important functionality such as linknavigation.If a JavaScript cannot be properly executed (e.g., becauseit is broken or has been disabled by the client) itshould not degrade the usefulness of the site (e.g., interms of navigability).• No sever-side scripting (such as JSP, PHP, or ColdFusion). This simplifies configuration and updating of the Websever as well as eases migration to anotherWeb server.It also simplifies maintenance of the Web site itself.For these reasons we decided to keep the BSENG sitestatic by generating the print view of the pages off-line.• Limited use of technology and features that are specificto the development tool.This mitigates vendor lock-in in case there is the desireto change the development tool. Worse, sometimes aneeded update of the development tool forces the developerto migrate code—the consequences and costsof such a migration are hard to predict.As an example, Toeter describes the migration of severalWebsites at the University of Amsterdam to a newversion of the ColdFusion Markup Language (CFML)[26]. 12000 lines of code had to be migrated fromCFML 1.0 to 4.0 because the ColdFusion developmenttool ceased to support the old CFML version.The migration had to deal with technological changes,changes in the tool, and non-backwards compatiblechanges in the CFML. We hope to avoid such migrationsscenarios for the BSENG site.During the development of the BSENG site, we noticed twoimportant tasks that we expect will remain critical duringthe maintenance and evolution of the site:1. Dead links both within the site and as the result of linkrot.2. Ensuring the integrity of the site's navigational structureUnfortunately, both of these tasks are not well supported byDreamweaver. As a response to this gap in tool support, wecustomized the Rigi reverse engineering environment. Wediscuss the tool support that Rigi provides for the maintainerin more detail in Sections 4 and 5.During development and maintenance activities,Dreamweaver and Rigi are used alternately. Typically,the Web developer first modifies the Web site withDreamweaver and then deploys the new site on the server.Once deployed, the Web site is crawled and its structurevisualized in Rigi. The visualization allows the maintainerto identify problems such as dead links, which leads to asubsequent maintenance activity with Dreamweaver.**4. Dead Links**Maintenance is constantly necessary even if theWeb siteitself does not change—the reason being that the environmentin which the Web site operates changes. This oftencauses links to pages that point outside of the Web site tobecome unreachable. (This phenomenon has been dubbedlink rot [6].) In a study conducted by Linos et al., four out ofnine Web sites had more than one percent of broken links;one site had as many as 7.5% dead links [12]. Another studyconducted in 1999 found that 5.7% of links on the Web arebroken.33Available at http://www.pantos.org/atw/35654.html.**Figure 5. Dreamweaver's link checker**Dreamweaver has a link checker that reports brokenlinks. However, the checker operates in the developer viewonly. This means that the report is accurate for broken intrapagelinks (i.e., anchors), but can give false positives forlinks between pages. False positives occur if mappings betweenthe deployment view and the client view are not properlyresolved. The mapping of a link between the two viewsis governed by the Web server, which is not taken into accountby Dreamweaver.Figure 5 shows part of the broken links reported for theBSENG site. In this run of the checker, broken intra-pageand inter-page links are detected. During development,broken links often act as to-do items indicating unfinishedwork.While Dreamweaver's ability to detect broken intra-sitelinks is useful for the Web site developer, it is too limitedbecause dead external links are missed. Dreamweaver listsall external links, but does not check them. Thus, the maintainerneeds an additional tool that checks the client-viewfor dead links.Web2Rsf, which has been previously developed by oneof the authors of this paper, extracts the link structure (clientview) of a Web site for subsequent analysis [13]. The resultof the extraction is visualized with the Rigi graph editor[25]. Rigi uses directed, typed graphs to process and displaythe data to be analyzed. Rigi can be retargeted to differentdomains by defining a suitable *schema* that expressesthe types and attributes of the arcs and nodes that constitutethe domain. In the Web schema that we defined, URLsare represented by nodes and links between URLs by arcs.Different node and arc types are represented with differentcolors (cf. Figure 7).There are different node types corresponding to differentURLs: HTML pages (`HTML`, blue), image files (`Image`,

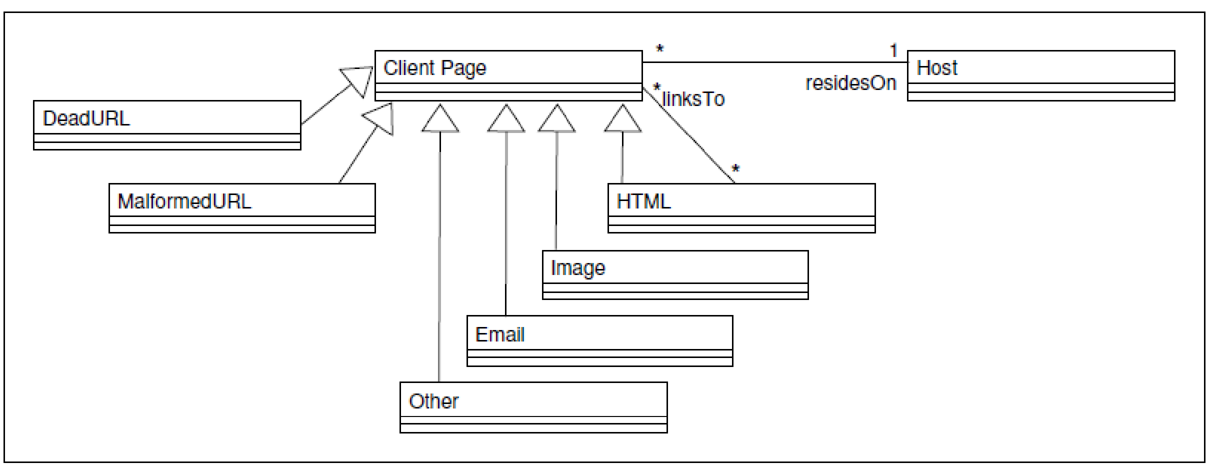

Figure 6. Rigi's Web schema for BSENGbrown),

`mailto:` (`Email`, green) etc. Files for downloading(e.g, PDF documents, ZIP archives, and Power-Point presentations) are grouped in a single type (`Other`,purple). There are also two distinct node types to indicatedead links (`DeadURL`, red) and syntactically incorrectURLs (`MalformedURL`, pink). Web2Rsf is writtenin Java and reports a dead link if an input/output exceptionis thrown while connecting.Figure 6 shows the Web schema that is sufficient tomodel the BSENG site. The schema is described with UMLand is similar to the one developed by Conallen [1] and DiLucca et al. [5]. Web pages are modeled as classes; an associationis used to represent hyperlinks. The complete RigiWeb schema is discussed in detail in a previous WSE paper[13].The Rigi views of the Web site allow Web site authorsto readily locate internal and external dead links. Web2Rsfdoes not check anchors, but these are reliably checked withDreamweaver. Figure 7 shows a rendering of a previousversion of the BSENG site. This Rigi view shows 54`DeadURL` nodes, which are rendered in red. No syntacticallyincorrect URLs were found. The site maintainer cannow filter out all the working URLs to focus on the deadones. When inspecting the dead links, we found externalones (referring to UVic's online calendar) as well as internalones. Many broken internal links were caused by problemswith wrong relative links in the print view. Furthermore, notevery page had a correct URL to its print view.

5. Navigational Structure

A well designedWeb site has an appropriate navigationalstructure that is apparent to its users and thus helps them toeffectively find desired information.Ricca and Tonella discuss several recurring navigationalstructures ofWeb sites [20]. A tree structure is acyclic andeach node has exactly one parent. A user navigates thetree structure from top to bottom, making a choice at eachlevel. A fully-connected structure means that each pagecan be reached by following a single link. An indexed sequencemeans that pages are arranged into a single/doublelinkedlist. Pages within the list allow navigation to theirprevious/next page. Optionally, a table-of-content page—from which all pages in the list are directly accessible—cansuper-impose a flat tree structure on the list. The BSENGsite's navigational structure is essentially a flat tree.Web sites can be composed of regions, each one havinga different navigational structure. For example, a future versionof the BSENG site might introduce a virtual tour of theUVic campus. This could be accomplished by adding anindexed sequence that is anchored with a table-of-contentpage to the current tree structure.A well-engineered Web site should be designed with acertain navigational structure and the structure should beclearly documented. Changes to a page due to maintenanceactivities and evolution of the site should not (unintentionally)change or violate the navigational structure.The navigational structure is determined by the linkageof client-view Web elements (e.g., HTML pages and`mailto:`). The Rigi Web schema describes these elements,and they are visualized in the Rigi graph editor asnodes and arcs. Thus, Rigi graphs can help the maintainerto assess the site's navigational structure. Rigi graphs canbe layed out with a spring algorithm to expose clusters inthe site's structure (cf. Figure 7). This allows the developerto assess the usability of theWeb site by evaluating thereachability of the most important Web pages.Data gathered from Web server log files can be used toinvestigate usage patterns of Web sites, i.e. which of theavailable links users actually followed and which of severalpossible paths users took to locate theWeb pages they werelooking for. This allows the Web site authors to optimizethe structure of their Web site to increase user satisfaction

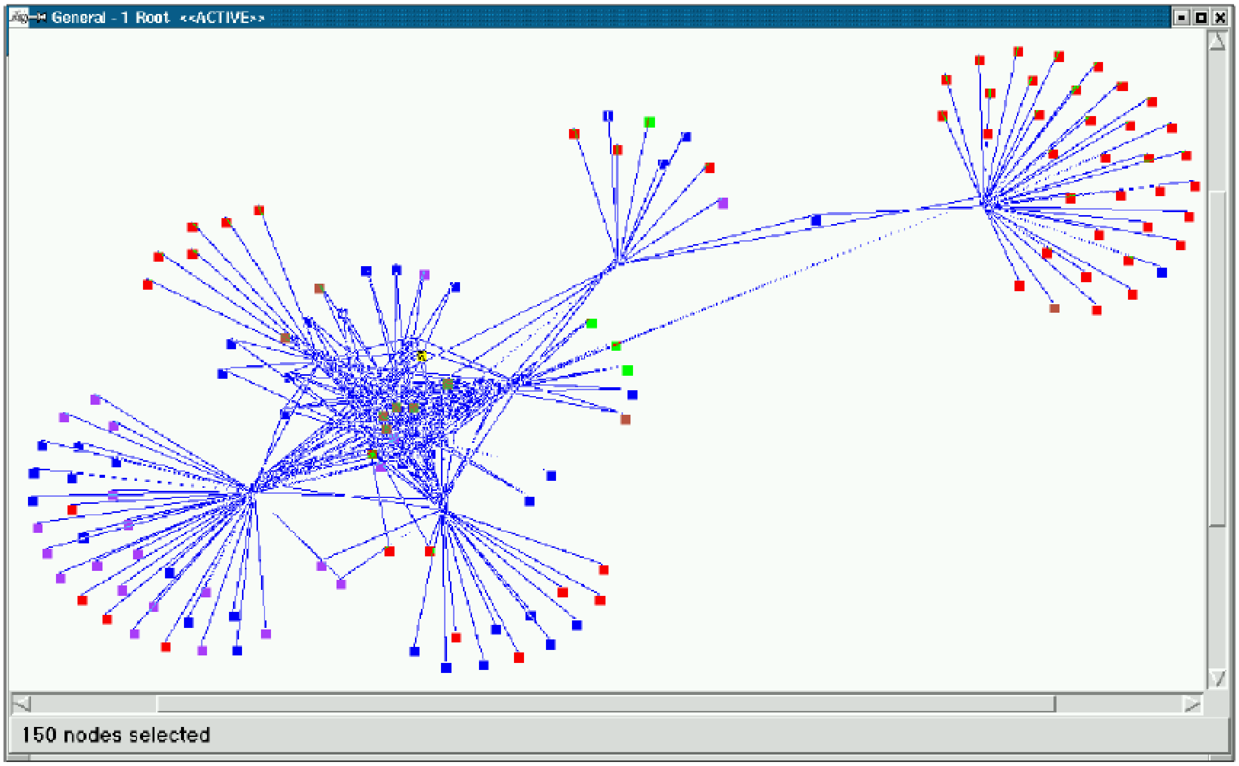

Figure 7. Visualization of dead links in Rigiand thus the potential success of their Web site.Since the BSENG site was still quite young and relativelyunknown at the time of our study, we were not ableto detect browsing patterns of Web site users. The dataWeb2Rsf gathered from the log files still revealed some interestinginformation when viewed in the Rigi graph editor.Web2Rsf records the number of accesses for every successfuland unsuccessful page request. In the log file we analyzed,there were about 300 requests for the BSENG startpage and about 130 requests for a file called `robots.txt`.By convention, this file is used to inform web spiders andsearch engines, whether they are allowed to index a website. The numbers suggest that about 40 percent of trafficon the Web site is caused by spiders. The main style sheetof the BSENG site was only requested about 250 times,suggesting that almost 20 percent of clients disregard stylesheets. Though these results can only be regarded as preliminarybecause of the small time span the log file covered,they give some useful hints about the users of the Web sitethat should be considered during future site maintenance.In Rigi, groups of related nodes can be collapsedinto a single “super” node (`Collapse`, cyan). Collapsingof nodes can be recursive, leading to a hierarchy of`Collapse` nodes. Such groupings are useful to documentthe site’s organization. When exploring a Rigi graph,a `Collapse` node can be expanded to reveal the elementsthat itcontains. The maintainer can do groupingsmanually in the Rigi editor. Alternatively, Rigi’sscripting capabilities make it possible to develop a scriptthat (semi-)automatically performs the grouping. Figure 8shows BSENG’s top-level graph after running the script thatwe developed for the BSENG site.4After a new version of the BSENG site has been deployed,the maintainer first runsWeb2Rsf and then executesthe Rigi script to visualize and assess the site’s new organization

.6. Conclusions and FutureWork

In this paper, we described our experiences with the developmentand maintenance of the BSENGWeb site, whichis an important asset for UVic’s BSENG program. We identifiedthe different views (development, deployment, andclient) that a Web developer has to work with. Differenttools and technologies operate on different views. It is importantfor the developer to understand the differences betweenthese views in order to interpret the information ofthe views correctly.Our experiences with the BSENG site show that MacromediaDreamweaver is an effective tool for site developmentbut lacks in support for important maintenance tasks.We discussed how the Web2Rsf extractor and the Rigireverse engineering environment can be used to generate4The size of the script is 220 lines of Tcl code.client-view documentation for assessing dead links and sitenavigability.Current reverse engineering approaches focus on the deploymentand/or client view, but neglect the developmentview. In the future, we plan to investigate reverse engineeringsupport for the developer view by extending existingcommercial Web development tools such as Dreamweaver,GoLive, andWebSphere with reverse engineering functionality.Reverse engineering analyses often generate documentationthat represents information at a higher level of abstraction.An important criterion for documentation is thatmappings between the different abstraction levels must bepreserved [28]. Mappings between abstraction levels are*vertical mappings*. The views that we introduced show thatdocumentation needs to be available for all three views andthat *horizontal mappings* between these documents are ofequal importance. Horizontal mappings can be quite complexand unintuitive for the Web site developer. We plan toinvestigate tool support that will allowsWeb site developersto effectively navigate such horizontal mappings.

Acknowledgments

Thanks to Keith Edwards and Crina-Alexandra Vasiliufor proofreading.This work has been supported by the Natural Sciencesand Engineering Research Council of Canada (NSERC),the Consortium for Software Engineering (CSER), and theCenter for Advanced Studies (CAS), IBM Canada Ltd

.References[1]

J. Conallen. Modeling Web application architectures withUML. *Communications of the ACM*, 42(10):63–70, Oct.1999.[2] T. Dean and Y. Chen. Design recovery of a two level system.*11th International Worshop on Program Comprehension(IWPC 2003)*, pages 23–32, May 2003.[3] G. A. Di Lucca, M. Di Penta, G. Antoniol, and G. Casazza.An approach for reverse engineering of web-based applications.*Eighth Working Conference on Reverse Engineering(WCRE '01)*, pages 231–240, Oct. 2001.4] G. A. Di Lucca, A. R. Fasolino, F. Pace, P. Tramontana, andU. De Carlini. WARE: a tool for the reverse engineering ofweb applications. *Sixth European Conference on SoftwareMaintenance and Reengineering (CSMR '02)*, 2002.[5] G. A. Di Lucca, A. R. Fasolino, and P. Tramontana. Towardsa better comprehensibility of web applications: Lessonslearned from reverse engineering experiments. *4th InternationalWorkshop on Web Site Evolution (WSE 2002)*, pages33–42, Oct. 2002.[6] D. Eichmann. Evolving an engineered web. *1st InternationalWorkshop on Web Site Evolution (WSE '99)*, Oct.1999.[7] P. J. Finnigan, R. C. Holt, I. Kalas, S. Kerr, K. Kontogiannis,H. A.M¨uller, J. Mylopolous, S. G. Perlegut, M. Stanley, andK. Wong. The software bookshelf. *IBM Systems Journal*,36(4), 1997.[8] F. Fraternali. Tools and approaches for developing dataintensiveweb applications: A survey. *ACM Computing Surveys*,31(3):227–263, March 1999.[9] A. E. Hassan and R. C. Holt. Towards a better understandingof web applications. *3rd InternationalWorkshop onWeb SiteEvolution (WSE 2001)*, pages 112–116, Nov. 2001.[10] R. Holt. TA: The tuple-attribute language.`http://plg2.math.uwaterloo.ca/˜holt/papers/ta-intro.html`, 1997.[11] F. Lanubile and T. Mallardo. Finding function clones inweb applications. *Seventh Conference on Software Maintenanceand Reengineering (CSMR 2003)*, pages 379–386,Mar. 2003.[12] P. K. Linos, E. T. Ososanya, and H. Natarajan. Maintenancesupport for web sites: A case study. *3rd International Workshopon Web Site Evolution (WSE 2001)*, pages 70–76, Nov.2001.[13] J. Martin and L. Martin. Web site maintenance withsoftware-engineering tools. *3rd International Workshop onWeb Site Evolution (WSE 2001)*, pages 126–131, Nov. 2001.[14] J. Nielsen. Alertbox for June 15: Top ten mistakes of webmanagement. *useit.com*, 1997. `http://www.useit.com/alertbox/9706b.html`.[15] J. Nielsen. Alertbox for May 2: "top ten mistakes" revisitedthree years later. *useit.com*, 1999. `http://www.useit.com/alertbox/990502.html`.[16] S. C. North and E. Koutsofios. Applications of graph visualization.*Graphics Interface '94*, pages 235–245, 1994.[17] J. Offutt. Quality attributes of web software applications.*IEEE Software*, 19(2):25–32, Mar./Apr. 2002.[18] R. S. Pressman. What a tangled web we weave. *IEEE Software*,17(1):18–21, Jan./Feb. 2000.[19] F. Ricca and P. Tonella.Web site analysis: Structure and evolution.*International Conference on Software Maintenance(ICSM '00)*, pages 76–86, Oct. 2000.[20] F. Ricca and P. Tonella. Understanding and restructuringweb sites with ReWeb. *IEEE MultiMedia*, 8(2):40–51, Apr.–June 2001.[21] L. Schmeiser.Web site evolution: Design and developing forthe future. *1st InternationalWorkshop onWeb Site Evolution(WSE '99)*, Oct. 1999.22] M. Taylor, J. McWilliam, J. Sheehan, and A. Mulhaney.Maintenance issues in the Web site development process.*Journal of Software Maintenance and Evolution: Researchand Practice*, 14(2):109–122, Mar./Apr. 2002.[23] S. Tilley and S. Huang. Evaluating the reverse engineeringcapabilities of web tools for understanding site contentand structure: A case study. *23rd International Conferenceon Software Engineering (ICSE 2001)*, pages 514–523, May2001.[24] S. R. Tilley. Domain-retargetable reverse engineering II:Personalized user interfaces. *1994 International Conferenceon Software Maintenance (ICSM '94)*, pages 336–342, Sept.1994.[25] S. R. Tilley, H. A. M¨uller, M. J. Whitney, and K. Wong.Domain-retargetable reverse engineering. *Conference onSoftware Maintenance (CSM '93)*, pages 142–151, Sept.1993.[26] B. Toeter. Lexical scanners for 4GL-source maintenance ofa corporate web site. *2nd International Workshop on WebSite Evolution (WSE 2000)*, pages 51–56, Mar. 2000.[27] A. van Deursen, P. Klint, and J. Visser. Domain-specificlanguages: An annotated bibliography. *ACM SIGPLAN Notices*,35(6):26–36, June 2000.[28] A. van Deursen and T. Kuipers. Building documentationgenerators.

International Conference on Software Maintenance(ICSM '99), pages 40–49, Aug. 1999.[29] P. Warren, C. Boldyreff, and M. Munro. The evolution ofwebsites. *7th International Workshop on Program Comprehension(IWPC '99)*, pages 178–185, 1999.