

Hidden Challenges on Teaching Web Software Engineering

Katia Passos, Eric Freeman, Cerise Wuthrich, Catherine Stringfellow
Department of Computer Science
Midwestern State University
Wichita Falls, TX 76308

ABSTRACT

The evolution of data communication technology has increased Internet traffic with negative effects on the user's response time. An existing mechanism to combat such effect is Web caching, in which intermediary systems are used to temporarily store information that may be retrieved by multiple users or by a same user multiple times. Software engineers, designing applications for this new environment, depend on techniques that emphasize the satisfaction of the users' requirements and the availability of computing resources on the processing site. However, software engineering classes may be overlooking some of these problems. In this paper, caching problems related to Web applications are presented. Knowing these problems exist is important in software engineering web-based applications.

INTRODUCTION

The evolution of data communication technology and graphical user interfaces has influenced software engineering course projects in recent years. While these courses still place emphasis on team development, applying analysis and design techniques, planning the project and testing, concepts of user interface design are often covered as well. Students working on their software engineering projects apply all these concept, and are also influenced by the commercially available software they use, much of which is networked and web-based. Some software projects may require developing software for hand-held devices like personal digital assistants (PDAs).

Network devices that are transparent to the users, such as web caches and firewalls, are usually ignored in the design process and not considered as resources that can improve or downgrade the performance of the application. Ignoring the existence of these devices during application design, however, may introduce critical errors in the behavior of the designed system. New software engineers should be prepared to identify these problems and document them in their requirement analysis.

Analysis of a set of interactive Web applications shows that designers prefer using scripts to assemble dynamic Web pages after evaluating a user's request. However, this design approach results in dynamic Web pages that are not usually cached, and consequently do not benefit from the performance improvement provided by the cache [2, 5]. Software engineers, designing applications for the new Web environment, depend on techniques that emphasize the satisfaction of the users' requirements and the availability of computing resources on the processing site. Other extensions to software engineering methods can model the processing site as a distributed environment that allows the designer to anticipate the use of communication networks and multiple systems. Bouguettaya et al. discuss a dynamic structure and system for describing data sharing [3]. Buhr explains how Use Case Maps (UCMs) can simplify complicated systems and introduce initial information on the network medium scale [4]. Karunanithy et al. discuss a

scenario where they provide personal information for mobile users with an assortment of end-user devices [10]. Mori, Paterno and Santoro discuss a tool that will help developers with different ways of presenting their tasks in a very detailed manner and to replicate their dynamic behavior [12]. However, such techniques focus on systems that are easily identified either by the user or by the designer. Other studies in software engineering focus on application development and do not consider the web caching influence on the behavior of the application [7, 8, 9, 11, 15].

In order to establish the basis for the new application development guidelines, the next section presents a brief description of web caching, followed by a discussion on problems identified with the implementation of an interactive web application. The paper ends with a summary of the concepts and observations presented in this study.

BACKGROUND

Web caching is commonly used to reduce the latency that a Web user perceives when requesting a page on the Internet. It is a mechanism in which intermediary systems are used to temporarily store information that may be retrieved by multiple users. One of the most common forms of Web caching has been to place a cache component in the user's system. Other choices are at institutional, regional and higher levels, as represented in figure 1. In a caching structure similar to the one in figure 1, the Web browser maintains in the host hardware a copy of the most recently visited Web pages. When a user requests information from a Web site, the browser attempts to fulfill the request from the pages located within its cache. When the request cannot be fulfilled from the local cache, a request to the remote host is initiated.

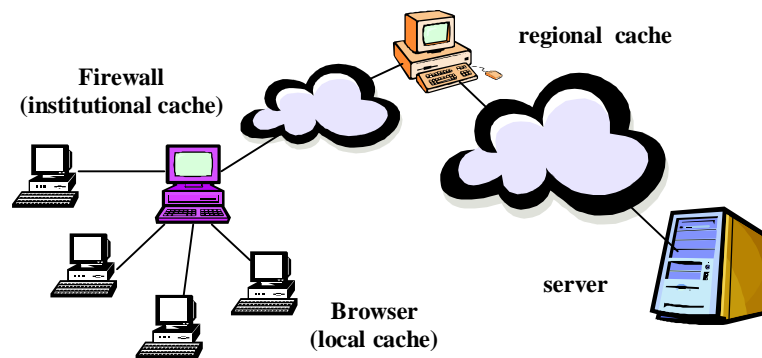


Figure 1. Diagram showing possible placement of Web caches.

To avoid having users from the same organization initiate duplicate requests for the same Web page, a cache is also often placed at the point where an organization's computer network connects to the Internet, usually a firewall system. This helps to reduce the number of outgoing requests, reducing the user's response time and the data traffic on the external network. A potential problem that must be considered at all levels of caching is the quality of the information of a stored Web page [1, 16]. If document X is potentially valid for several hours before it must be updated, then caching X will improve the overall system performance without having any negative consequence. However, if an object Y is more likely to have a short life span and be meaningless if not frequently updated, caching Y could send invalid responses to the user accessing that information. The hypertext transfer protocol (HTTP), the communication protocol used to access Web pages, defines cache management parameters that describe how to properly handle information located on a page, for example, if a page is 'cacheable' or not.

Currently, some of the most common and significant parameters allow the applications to specify if a page is cacheable, the length of time the information can be kept in a cache, if the page should be cached only at the user's local cache, or not cached at all.

Applications that interact with the user in a way that generates new information, such as a flight reservation system or an online bookstore, are usually implemented by using scripting languages and database access. PHP (PHP Hypertext Preprocessor) is considered to be the most popular Web application server scripting language [17]. PHP is used to define that access to the database and to build the web page to be sent as response to the user's request. The response web page is coded in HTML for the more traditional Wired Web and eventually in wireless markup language (WML) when communicating with wireless devices such as a cell phone [6].

WEB APPLICATION CHALLENGES

Web caching is usually transparent to the user and to the application designer, except for the possible improvement in response time. The application designer, when planning the development of a system, usually will not have enough information to judge if a web cache is involved. Also, if this developer is not knowledgeable in network protocols, he or she will focus on the application functionality, i.e., the interface between the scripting language and the database and the assembly of the pre-defined response pages. In order for students to understand the consequences of ignoring the information on web caching, the instructor may present the following scenarios exemplifying the use of interactive applications.

Case 1: A cell phone user arrives at the airport and finds out that her flight was cancelled. By accessing the Internet through the cell phone, such a user can immediately re-schedule her flight (as seen in some TV commercials). In order to do that, the user needs to find a different flight, check for seat availability and make a reservation. A second passenger, with the same problem, borrows her cell phone and without disconnecting tries to re-schedule his flight to the same flight that she got the new reservation. A typical system will provide an answer for the second user that shows at least one less seat in the availability chart of the new flight (the seat reserved by the first user). However, if the caching system resident in the cell phone decides to store dynamic web pages, the second user may end up getting exactly the same answers as the first user, which would imply that both reserved the same seat (also assumed here is that no special dynamic caching technique was used to invalidate the pages after they were stored).

Case 2: A student takes an online multiple-choice test and when verifying the score, finds out that one of the answers in the test key was wrong. Under these conditions, the student got a score of 85 points when it should really be 90. The student complains with the faculty member responsible for the test who immediately corrects the answer key in the database. A second student comes in and takes the same test in the same machine. By coincidence, this student answers exactly the same choices as the previous student. Even considering that the database with the answer key has been corrected, the student still gets an 85 point score. It is clear then that the machine had cached the previous responses from the server and was unable to distinguish between the two users.

When developing the application, the designer may not be able to obtain enough information to predict these situations. However, a simple solution exists to avoid such a problem. The web caching systems store and identify pages by their URL. The caching system, when receiving a URL, should be able to identify it as a dynamic page, if the URL includes any form of parameter passing. For example,

<http://www.hotelselect.com/res/html/HInfo?hotel=FL555&sd=7>

is a modified URL from an actual request for a hotel reservation system. If the browser is unable to recognize this URL as a dynamic request, then it will retrieve stored pages that were cached earlier with the same URL, introducing an error in the information generated by the system as described in the two case scenarios presented earlier. Users, as well as developers, may look at solutions that investigate all the current characteristics of the network structure in which the system will be used. Such a solution is not sufficient to eliminate the possibility of errors, since future changes in the network may introduce new variables in this process. Another possibility is to resize the local cache to zero in all machines using the system. However, this would still leave a chance of future changes in the cache size. In addition, it would not solve the problem, if another cache existed in the path between the local system and the application server. This last option would also downgrade the performance of the Web browser for other applications that do not depend on dynamic pages.

A simple solution, which would work according to the HTTP definition, is the use of HTTP header parameters to specify dynamic responses as non-cacheable. While this is probably the best approach, it still assumes that any cache system will be correctly configured, working under the expected HTTP standards. A more trivial solution would be to add to the list of parameters the time and date of the request, making each request unique with respect to the cache screening mechanism. The server side script would disregard the date/time information, if not necessary to the application, and proceed by preparing and sending a new answer. Unfortunately, this solution increases the amount of data being transmitted by the system. Other solutions would require a better software engineering methodology.

EXPERIMENT DEMONSTRATION

In order to verify the possibility of these catastrophic errors, an online wireless testing application that allows a student to take multiple-choice tests is developed. First, a series of WML decks are written introducing the user to a school's home page and then linking to an individual faculty member's pages. From the faculty Web page, the online testing application is accessed through a hyperlink. Next, the database containing all information about tests including name of the test, objectives covered, questions, answer choices, and feedback responses was created. The WML code can be written with a PHP script embedded into it, connecting it to the MySQL database to prompt the user to select which test to take. Figure 2 shows experimental output on a generic cell phone and on a PDA. This application allows a student to take multiple-choice tests. The questions are displayed one at a time, along with answer choices.



Figure 2. (a) Display of question with answer choices (Image courtesy Openwave Systems Inc.).
(b) Testing system implemented in a PDA.

Each time the student answers a question, the answer is saved and the next question along with answer choices is retrieved. The application continues to display questions until it was determined that all questions for a particular test had been presented. At that time, the application shows each question number and the student's response to the question. After all answer choices and feedbacks have been displayed, the student's test score is shown.

The suggested application was tested using the simulator embedded in Openwave™ SDK [13, 14]. This simulator assumes that any page is cacheable and therefore, it will retrieve from the cache dynamic pages that passed exactly the same parameters. During the experiment, the main concern was if and how the various pages would be cached. When the simulator was initially started and a particular test with 20 questions was selected and taken, there were 25 server hits. After taking the test once, the same test was taken again with exactly the same answers. During this instance of test taking, there were 0 server hits or 100% cache hits. To further examine the caching process with the simulator, the test was taken once again, supplying all correct answers (score of 100) and without disconnecting before taking the test. However, the database in the Web server was updated and the correct answer choice on question 1 was changed. Taking the test again with the same 20 answers as the first time should result in a score of 95. Actual results show it still scored 100, because it used a page in the cache left over from the previous time the test was taken. In order to improve the test, meta tags were added to the header portion of the WML code specifying a zero value for maximum age. This action was equivalent to assigning the page as non-cacheable. The experiment was conducted again and during the second test, all pages were retrieved from the server even though the answer list was identical. The experiments conducted with this simple application help to confirm the uncertain behavior of the caching systems. The current research in caching dynamic pages also suggests that future changes will be implemented in the systems and the developer will not be able to prepare for them. The use of unique parameters in the page requests seems to be the more stable and reliable solution at this point and worked well in experimental trials.

SUMMARY

Network solutions, such as firewalls, and web caching, are usually transparent to the user and to the software engineer designing the application. This paper shows that web-caching systems may interfere with the behavior of an application and introduce invalid information to the user. Software engineering instructors need to prepare students for these new challenges. Two simple mechanisms already available in the network protocols allow the developer to guarantee the correctness, in any circumstance, of the behavior of the application with respect to web caching. Unfortunately those solutions contribute to increase the network traffic load. Additional studies are required to allow the identification of new solutions to this problem and to evaluate the cost benefit of adopting those solutions.

ACKNOWLEDGEMENTS

This work was partially supported by the Texas Advanced Research Program under Grant No. 003656-0108b-2001.

Openwave and the Openwave logo are trademarks of Openwave Systems Inc. All rights reserved.

REFERENCES

1. M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich and T. Jin, "Evaluating Content Management Techniques for Web Proxy Caches," *ACM SIGMETRICS*, 27, (4), 2000, 3-11.
2. G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques," *IEEE Communications*, 38, (5), May 2000, 178-184.
3. A. Bouguettaya, B. Benatallah, L. Hendra, M. Ouzzani and J. Beard, "Supporting Dynamic Interactions among Web-Based Information Sources," *IEEE Trans on Knowledge and Data Engineering*, 12, (5), May 2000, 779-801.
4. R. J. A. Buhr, "Use Case Maps as Architectural Entities for Complex Systems," *IEEE Trans on Software Engineering*, 24, (12), December 1998, 1131-1155.
5. K. S. Candan, W.-S. Li, Q. Luo, W.-P. Hsiung, and D. Agrawal, "Enabling Dynamic Content Caching for Database-Driven Web Sites," *Proc of the ACM SIGMOD 2001*, Santa Barbara, CA, 30, (2), May, 2001, 532 - 543.
6. E. Castro, *HTML for the World Wide Web*, 4th Ed., Peachpit Press, Berkeley, CA, 2000.
7. M. H. Cloyd, "Designing User-Centered Web Applications in Web Time," *IEEE Software*, 18, (1), January/February 2001, 62-69.
8. L. L. Constantine and L. A. D. Lockwood, "Usage-Centered Engineering for Web Applications," *IEEE Software*, 19, (2), March/April 2002, 42-50.
9. E. Hendrickson and M. Fowler, "The Software Engineering of Internet Software," *IEEE Software*, 19, (2), March/April 2002, 23-24.
10. K. Karunanithi, K. Haneef, B. Cordioli, A. Umar and R. Jain, "Building Flexible Mobile Applications for Next Generation Enterprises," *Proc of the IEEE Academia/Industry Working Conf on Research Challenges*, Buffalo, NY, April 2000, 127-132.
11. A. Knight and N. Dai, "Objects and the Web," *IEEE Software*, V. 19, (2), 2002, 51-59.
12. G. Mori, F. Paterno and C. Santoro, "CTTE: Support for Developing and Analyzing Task Models for Interactive System Design," *IEEE Trans on Software Engineering*, 28, (8), 2002, 797-813.
13. *Openwave Mobile Access Gateway Edition*, Openwave Systems, Inc., Redwood City, CA, Feb. 2002, pub. DSMAG-R5.1-004.
14. *Openwave SDK WAP Edition 5.0*, Openwave Systems, Inc., Redwood City, CA, Sep, 2001, pub. DSSDK-R1-001.
15. D. Petriu and M. Woodside, "Analyzing Software Requirements Specifications for Performance," *Proc of the Workshop on Software and Performance WOSP '02*, Rome, Italy, July 2002.
16. L. Rizzo and L. Vicisano, "Replacement Policies for a Proxy Cache" *IEEE/ACM Trans on Networking*, 8, (2), April 2000, 158-170.
17. A. Royappa, "The PHP Web Application Server," *The Journal of Computing in Small Colleges*, 17, (4), March 2000, 201-211.