

Using Sequence Diagrams to Document Web Elements' Unknown Behavior

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

Abstract

Software engineers depend on analysis and design methods that take into consideration user's requirements and computing resources on the processing site. With many new network applications being developed based on web browser user interfaces, software engineers have become more dependent on knowledge of network operations. Web caching is an existing mechanism that seeks to improve user's response time, but their existence and functionality are usually transparent to those not directly working with the network infrastructure. In this paper, a new method that uses UML sequence diagrams to perform requirements analysis on web applications is proposed, whereby the web application is designed in such a way to avoid the actions taken by hidden resources, such as web caches, becoming, then, independent of the uncertainty of the caching mechanisms for dynamic contents.

1. Introduction

Advances in data communication have increased the use of the Internet and Web based systems as basic components of new applications. This new communication environment has brought back old problems such as providing users with a short response time. Prolonged response times frustrate the user and require new network performance improvement mechanisms. One of these mechanisms involves Web caching, where intermediary systems can temporarily store information traveling through the network. Unfortunately, these new techniques may affect the behavior of some Web applications, introducing critical errors in their functionality.

Software engineers, working in this new environment, depend on methodologies that emphasize the satisfaction of the users' requirements and the availability of computing resources on the processing site [21]. New 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 [5]. However, such techniques focus on resources that are easily identifiable during the analysis phase of the software development process. Most of the network infrastructure is considered reliable and is not considered to influence the behavior of the application.

Most Web applications generate pages that are not supposed to be cached due to the volatile aspect of the information they contain. This prevents the application from benefiting from the performance improvement provided by Web caches [2]. Studies focusing on cache systems and algorithms to decide when to cache a dynamic page have been conducted and some new techniques have been implemented in commercial caches [6]. In this paper, a new methodology of documenting this problem is presented where the Web application is tagged as dependent on this hidden network environment, allowing the designer to consider the consequences of Web caching involvement in the design of the application.

Several solutions to the problem of caching dynamic pages have been proposed. A potential problem that must be considered at all levels of caching is the quality of the information of a stored Web page [1, 19]. If a document is potentially valid for several hours before it must be updated, then caching that document will improve the overall system performance. However, if the document has been dynamically created as a unique response to a specific query, such as an inquiry on a hotel room availability, then caching the responding page could imply the possibility of invalid responses to future user accesses to that information. Candan et al. describe a front-end dynamic-web-content cache, known as CachePortal [6]. This cache is kept current by a sniffer and an invalidator that interact with the web server, the application server, and the database system. The sniffer gathers information on user requests and database queries. The invalidator uses that

information to send messages to the cache to discard pages that were affected by changes in the database. However, these changes in the system require the developer to have control not only on the application system but also on the network implementation, which is not always possible if one considers the Internet as its communication media.

Web application designers depend on methodologies that focus on users' requirements and computing resources on the processing site. These methodologies are the subject of several research efforts. For example, Fernandez et al. describe a language to specify a site's content and its HTML representation [10]. Processing sites can also be modeled as distributed environments that allow the representation of communication networks and multiple systems. In this context, Bouguettaya et al. discuss how to describe data sharing [4]. Buhr proposes a simplification of system representation, introducing initial information on the network medium [5]. Karunanithy et al. present a scenario for mobile users with an assortment of end-user devices [12]. Mori, Paterno and Santoro describe a tool that represents tasks in detail and replicates their dynamic behavior [14]. However, such techniques are applicable to systems and resources that are easily identified either by the user or by the designer.

Other studies are oriented towards the application functionality and do not consider the possibility of Web caches impacting the behavior of the application. Pertriu, Shousha, and Jalnapurkar discuss the use of an UML description of a telecommunication product [17]. Petriu and Woodside report weaknesses in an analysis that includes lack of knowledge and other aspects using model building from software design products [18]. Bernardi and Santucci propose Web based hierarchical model libraries and reusable designs using an object-oriented architecture [3]. Knight and Dai also work towards an object-oriented Web applications approach [13]. Cloyd emphasizes human factor methods in use-driven processes [8]. Constantine and Lockwood focus on user interface design and usability issues applicable to Web centric systems [9]. Hendrickson and Fowler also concentrate their study on user interaction and hosts [11]. The results of this study affect the way that applications are written in the implementation phase of the software project. The proposed methodology has the benefit of alerting the designer to unpredictable behavior of the application when implemented in Internet based systems.

The next section introduces the concept of web caching and software engineering scenarios, followed by a discussion on problems already identified with the design and implementation of interactive web

applications. The subsequent sections propose a new approach to represent the application scenarios.

2. Background

The concept of Web caching is very similar to the traditional definition of cache memory. In a network environment, Web caches are placed close to the user. Figure 1 shows a representation of possible locations for a Web cache according to a hierarchical structure. In such a caching structure, the Web browser maintains a copy of the most recently visited Web pages in the local (user) system. When a new Web page request is initiated, the browser attempts to fulfill the request from the pages located within its cache. When the requested page is not found in the local cache, a request to the Web server is initiated.

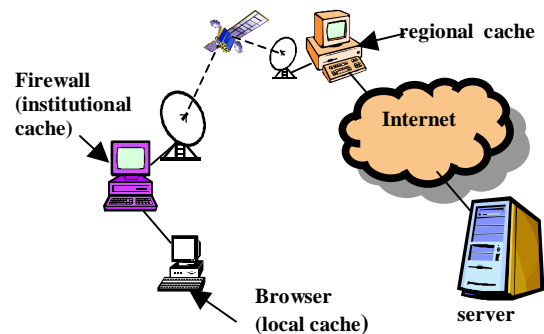


Figure 1. Possible placement of Web caches.

Web applications that require user interaction to generate new information, such as a flight reservation system or an online bookstore, are usually prototyped by using scripting languages. These prototypes may become operational products if considered efficient and able to provide the necessary functionality to the system. One of the most used scripting languages is PHP (PHP Hypertext Preprocessor) [20]. 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 [7, 22].

Web caching is usually transparent to the user and to the application designer. However, the application designer, usually, will not have enough information to establish if a web cache is involved in the system operation. In order to understand the consequences of ignoring the information on web caching, assume 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 the 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 on which she got the new reservation. A typical system should 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).

A simple solution exists to avoid the problem. Web caching systems store pages by their URL and should be able to identify a dynamic page when the URL includes any form of parameter passing. For example, *http://www.hotelslct.com/res/Inf?htl=34&pd=1* is a modified URL from an actual request for a hotel reservation system. The *htl* and *pd* parameters should provide the necessary information about the dynamic nature of the access. However, if the user is just requesting directory information, then even if the page is dynamically created, the use of a cached version does not invalidate its contents. Therefore, one must consider that some dynamic pages may be cached without any negative effect in the application.

3. Requirement Analysis

When developing applications involving network data communications, software engineers may try to cover all aspects of the system components by recording in their requirement analysis typical behavior of the network that might interfere with the functionality of the system being designed. While this may be the solution for multiple problems, it only works when the components are entirely developed as part of the system, i.e. no off-the-shelf browsers and no external network structure is involved. However, many systems are currently being developed on top of the Internet and commercial browsers in order to implement the so-called enterprise systems. These new systems depend on the standardization of those off-the-shelf components.

Web caching and firewalls are two examples of recent technology added to the networks in order to improve performance and security. Such systems have the characteristic of being configured by their administrators and those administrators usually are not associated with the enterprise system being designed. This situation allows changes in the system that could

affect the behavior of the application. A simple solution with respect to the web-caching problem, which would work according to the HTTP definition, is the use of HTTP header parameters to specify the 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. In any of these cases, the software engineering methodology used to document the requirement analysis must be able to indicate the possibility of such problems so the development team could follow those specifications in order to implement a reliable system.

A common technique used in describing the results of a requirement analysis is the use of scenarios or use cases. A sequence diagram can represent the behavior of a single use case [11]. In a sequence diagram there are boxes at the top of the diagram that represent objects. Each object has a lifeline, which is represented by a dashed vertical line underneath it. Between these objects there are actions taken (messages are passed), which displays their interactions. Each action is shown in order of their expected occurrence, from top to bottom and may have labels included for the name of the action. In this diagram an activation box may be included to show that the object is active. Take into consideration figure 2. A student who has been accepted to a university does not have the money for tuition. He or she goes to a local bank and applies for a loan. The student is granted the loan from the bank, and then he

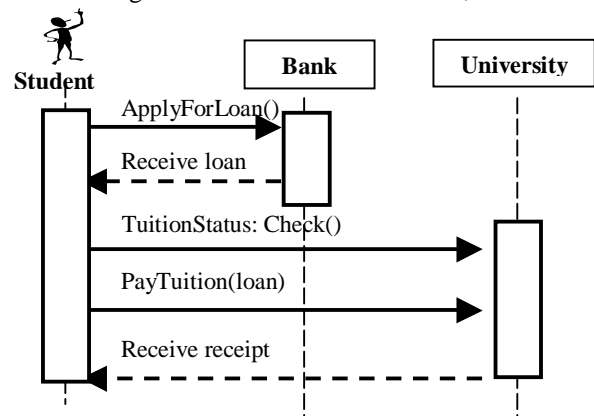


Figure 2. Sequence Diagram.

or she checks on the cost of the tuition. The student goes to the university to pay the tuition with the loan and in return receives a receipt.

An alternative to this procedure is the Use Case Maps (UCM) technique, proposed by Buhr [5]. A Use Case Map is a scenario-based notation that attempts to simplify and generalize how the structure of a complex system and the behavior of the system are intertwined. The focus is on the big picture. The Use Case Map lacks formality, does not pay attention to details, and is lightweight. UCMs are meant to help stimulate thinking, discussion, inference, and visualization about a system. The core notation of a UCM involves three fundamental elements:

- a. wiggly lines representing scenario paths
 - b. rectangular boxes representing runtime components
 - c. x's representing responsibility points
- Also included are start and end points, forks, and joins.

To use and read a UCM, it is suggested that a person place a token (either mentally or literally) on each start point and then move each token along its path. If paths cross intercomponent gaps, this infers intercomponent communication requirements. The UCM thus provides a framework for reasoning about such intercommunication without requiring commitment to details.

Figure 3 illustrates a network transaction. The component stacks (for client and server) indicate that many concurrent yet independent transactions may be in progress at once. There is a decision to be made concerning how transaction-path concurrency will be implemented. Yet this diagram does not concern itself with the details of that decision. It merely provides a context for thinking about the decision. The figure also illustrates the "behavior structure" for providing transaction completion in case of network failure (as indicated by the ground symbols). This UCM shows two routes from A to B. One is the result of a timeout and the other shows a normal transaction completion. In either route, the AND-fork mandates that a timer is set and the transaction must wait for cancellation of the waiting condition whether through normal transaction completion or a timeout.

Ordinary transaction completion prompts a cancel responsibility that cancels the waiting condition and the timer. This is a typical example of how to document the standard behavior of the network in such a way to have it documented for the software being developed.

4. Preventing web cache influences

In this study, in order to verify the possibility of errors, an online wireless testing application was

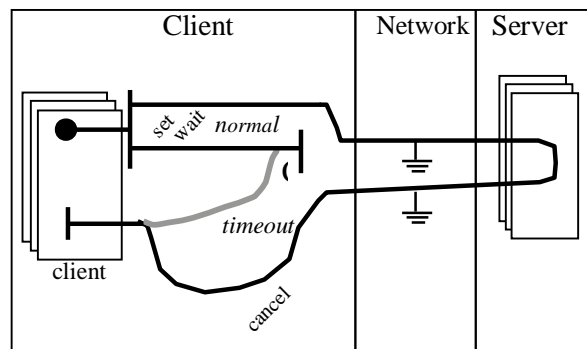


Figure 3. A generalized network transaction.

developed. A database was created containing all information about the tests including name, objectives, questions, answer choices, and feedback responses. A script language was used to connect the database to the user. A portion of the experimental output on a generic cell phone is shown in figure 4. This application allows a student to take multiple-choice tests. The questions are displayed one at a time, along with answer choices. Each time the student answered a question, the answer was saved and the next question along with answer choices was retrieved. The application continued to display questions until it was determined that all questions for a particular test had been presented. At that time, the application showed each question number, the student's response to the question, and if the answer was correct or not. If the student answered incorrectly, a brief feedback response was presented. After all answer choices and feedbacks were displayed, the student's test score was shown.

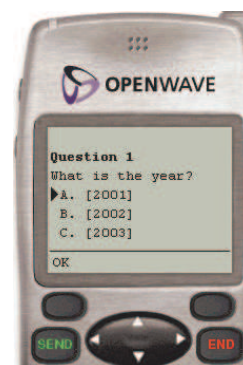


Figure 4. Display of question (Image courtesy Openwave Systems Inc. [15, 16].

The application was tested using cell phone and PDA simulators. Some of the browsers assumed that any page was cacheable and therefore, it retrieved from the cache dynamic pages that passed exactly the same parameters. After an initial run of a particular

test with 20 questions, 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 a second time. Before taking the test the second time, 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. Other simulators used seem not to cache any page.

Figure 5 shows a sequence diagram representing the most appropriate scenario for this testing application. It shows a non-initial run of the system in a multi-user environment, where questions could be cached and where answers were supposed to bypass the cache. While this scenario could be considered a reasonable tool in the development of the application, it does not work because the development team may not have access and control on how the cache operates. Therefore, the simple idea that there is a cache in the system is just a sign of trouble for the application. The correct idea to be documented in the requirement analysis diagram is the existence of an extraneous component, which cannot be controlled by the application.

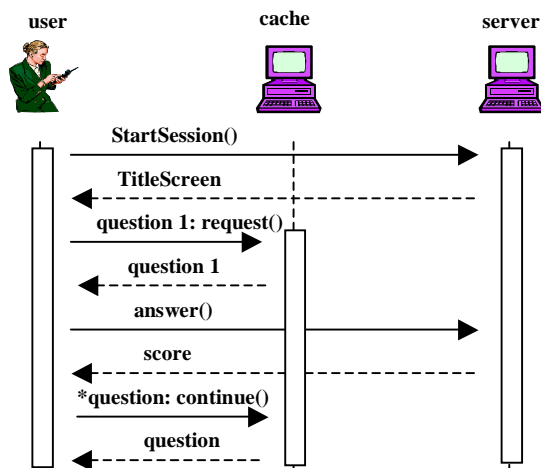


Figure 5. Sequence diagram showing intermediary systems

Figure 6 shows the proposed modification to the scenario diagrams where an interruption in the flow of the data indicates the intervention of elements external to the application system being developed. This interruption could be labeled or not, depending if all the elements are known, in such a way to allow future

project management decisions on replacing those elements by in-house developed components, which would guarantee a standard behavior, as seen in figure 6. This solution is similar to the *transactor* proposed by Buhr [5]. However, the *transactor* assumes that the designer has the knowledge on its functionality.

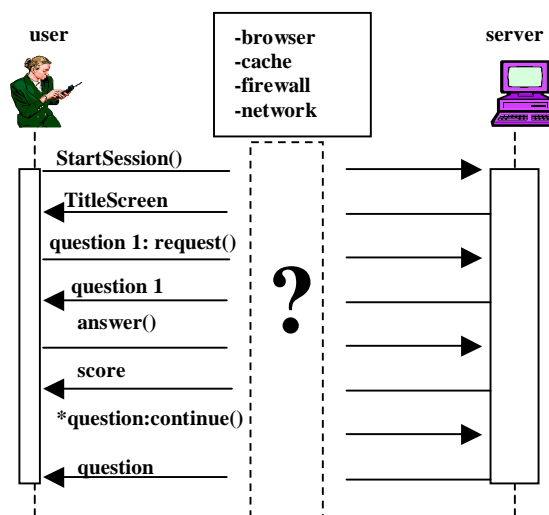


Figure 6. Scenario with hidden network resources

After receiving the diagram shown in figure 6, a software engineer working on an implementation team, would need to analyze all possible behaviors that could be expected from the unknown component interference. Assuming the designer has no network experience, he or she could still plan for the following situations:

- A. From the user's (client) side:
 1. messages do not arrive at the server
 2. messages arrive in duplicate
 3. erroneous messages are sent to the server by the unknown component
 4. messages reach the server with success
- B. From the server side:
 1. pages do not arrive to the user's system
 2. pages arrive in duplicate at the user's system
 3. pages are generated by the unknown component
 4. pages arrive successfully

With all these options planned, the system can be developed in such a way to provide the expected reliability. In this study experiment, several options were tested including the use of cache-time expiration parameters defined in HTTP, time tags generated in the access of dynamic pages, no action or disregarding the existence of the unknown element, etc. Exhaustive tests have shown that when the correct actions were

taken the system behaved according to the user initial requirements.

5. Conclusion

While Web caching seems to be fully transparent to the user, those systems may interfere with the behavior of an application and introduce invalid information in the flow between client and server machines. Simple mechanisms already available in the network protocols that allow the developer to guarantee the correctness, in any circumstance, of the behavior of the application with respect to web caching are costly in performance. The correct documentation of the system requirements becomes an essential tool for the developer to be able to formulate distinct solutions to the target problem. The use of sequence diagrams to represent scenarios is a common visual tool available to UML users. The introduction of a black box in the sequence representation will allow the development team to communicate the existence of system elements out-of-control of the project, requiring special handling in order to maintain the system reliability.

6. 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.

7. References

- [1] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich and T. Jin, "Evaluating Content Management Techniques for Web Proxy Caches," *ACM SIGMETRICS Performance Evaluation Review*, Vol. 27, No. 4, March 2000, pp. 3-11.
- [2] G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques," *IEEE Communications*, Vol. 38, No. 5, 2000, pp. 178-184.
- [3] F. Bernardi and J. F. Santucci, "Model Design Using Hierarchical Web-Based Libraries," *Proc. of the Design Automation Conference*, June 2002, pp. 14-17.
- [4] A. Bouguettaya, B. Benatallah, L. Hendra, M. Ouzzani and J. Beard, "Supporting Dynamic Interactions among Web-Based Information Sources," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No. 5, May 2000, pp. 779-801.
- [5] R. J. A. Buhr, "Use Case Maps as Architectural Entities for Complex Systems," *IEEE Transaction on Software Engineering*, Vol. 24, No. 12, 1998, pp. 1131-1155.
- [6] K. S. Candan, W.-S. Li, Q. Luo, W.-P. Hsiung, and D. Agrawal, "Enabling Dynamic Content Caching for Database-Driven Web Sites," *Proceedings of the ACM SIGMOD 2001*, Santa Barbara, CA, Vol. 30, No. 2, May, 2001, pp. 532 - 543.
- [7] E. Castro, *HTML for the World Wide Web*, 4th Ed., Peachpit Press, Berkeley, CA, 2000.
- [8] M. H. Cloyd, "Designing User-Centered Web Applications in Web Time," *IEEE Software*, Vol. 18, No.1, January/February 2001, pp. 62-69.
- [9] L. L. Constantine and L. A. D. Lockwood, "Usage-Centered Engineering for Web Applications," *IEEE Software*, Vol. 19, No. 2, 2002, pp. 42-50.
- [10] M. Fernandez, D. Florescu, A. Levy and D. Suciu, "Declarative Specification of Web Sites with Strudel," in the *Very Large Data Base Journal*, Vol. 9, No. 1, 2000, pp. 38-55.
- [11] E. Hendrickson and M. Fowler, "The Software Engineering of Internet Software," *IEEE Software*, Vol. 19, No.2, March/April 2002, pp.23-24.
- [12] 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 Conference on Research Challenges*, Buffalo, NY, April 2000, pp.127-132.
- [13] A. Knight and N. Dai, "Objects and the Web," *IEEE Software*, Vol. 19, No. 2, March/April 2002, pp.51-59.
- [14] G. Mori, F. Paterno and C. Santoro, "CTTE: Support for Developing and Analyzing Task Models for Interactive System Design," *IEEE Trans. on Software Engineering*, Vol. 28, No. 8, 2002, pp. 797-813.
- [15] *Openwave Mobile Access Gateway Edition*, Openwave Systems, Inc., Redwood City, CA, Feb. 2002, pub. DSMAG-R5.1-004.
- [16] *Openwave SDK WAP Edition 5.0*, Openwave Systems, Inc., Redwood City, CA, 2001, pub. DSSDK-R1-001.
- [17] D. Petriu, C. Shousha and A. Jalhapurkar, "Architecture Based Performance Analysis Applied to a Telecommunication System," *IEEE Transactions on Software Engineering*, Vol. 26, No. 11, November 2000, pp.1049-1065.
- [18] D. Petriu and M. Woodside, "Analyzing Software Requirements Specifications for Performance," *Proceedings of the Workshop on Software and Performance WOSP '02*, Rome, Italy, July 2002.
- [19] L. Rizzo and L. Vicisano, "Replacement Policies for a Proxy Cache" *IEEE/ACM Transactions on Networking*, Vol. 8, No. 2, April 2000, pp. 158-170.
- [20] A. Royappa, "The PHP Web Application Server," *The Journal of Computing in Small Colleges*, Vol. 17, No.4, March 2000, pp. 201-211.
- [21] M. Fowler and K. Scott, *UML Distilled*, 2nd ed., Boston, MA: Addison-Wesley Publ. Company, 2000.
- [22] *Wireless Application Protocol WAP 2.0. Technical White Paper*, WAP Forum, Jan. 2002.