Tunghai Science Vol. 2: 1–19 1
July, 2000

Automatic Testing for Link Correctness on the Internet

Wen-Kui Chang * Shing-Kai Hon *

Abstract

The user-friendly convenience of the WWW promotes its rapid application delivery, but the technical complexities of a website and variances in the browser make web testing and quality control much more difficult. This paper proposes an automatic framework for testing hyperlink correctness in the Internet environment. We first investigate the underlying features of Internet applications and then discuss quality issues and a performance index. We study how to employ the technique of statistical usage testing in such a domain and develop an integrated validating mechanism for web-based applications. All possible navigation paths are first formulated to represent a scenario model with the Markov chain property, which is then used to statistically generate a navigation-script file. After tracing these navigation scripts, we may then perform numerical computations of link validation based on Markov chain theory. It is noted that the proposed automatic mechanism is not only systematically effective on certifying hyperlinks, but also highly efficient for designing a test process.

Keywords: Internet, software performance testing, statistical usage testing, Markov chain, object-oriented approach

1. Introduction

As Internet applications have grown explosively in the last few years, more and more web sites employ a variety of new technologies including Common Gateway Interface (CGI), Java, Java script, Visual Basic (VB) script, Active Server Processor (ASP), etc. As a result, it is severely complex and even impossible to analyze the quality of web applications since there are so many factors such as networking architecture, communication speed, machine configurations, traffic load, etc., which is highly correlated with performance of a website.

In most web applications, it is unlikely that the application developers can enumerably validate all possible navigation paths. This paper proposes an automatic framework of quantitative certification of link validity to ensure that all linked paths provide consistent and reasonable information streams and appropriate contexts.

In this research, the rationale for statistical usage testing is investigated and the feasibility of its application in certifying hyperlinks of a website is studied. In principle, a navigation model is first established to characterize various browsing paths of a web site. Hence, if a browsing sample is drawn statistically as a test script from the navigation model, performance on this script sample may then be used as a basis for the evaluation of web quality.

With the illustration of a practical demonstration, it is shown that the proposed automatic framework is not only systematic in certifying effectively hyperlinks, but also highly efficient for complex web sites on the Internet. In contrast to the other methods of testing, the recommended approach has benefits including greatly reducing testing effort, providing quantitative testing coverage, etc.

In addition, since the combination of web site complexity and low quality is potentially fatal to company business, more research efforts must be devoted to performance of e-business applications. Hereafter, validating on e-business web site is the future direction of our research

2. Features of the Internet testing environment

2.1 Internet server architecture

As shown in Fig. 1, the server in a client/server architecture provides services to its client subsystems, which are responsible for receiving inputs from the user, validating query syntax, and initiating the required transactions. The server is then responsible for performing the transaction and guaranteeing the integrity of the results. Nowadays the request for a service is usually done via a remote procedure call mechanism or a common object broker, such as CORBA or Java RMI. Control flows in the clients and the servers are independent except for

* Department of Computer & Information Sciences, Tunghai University, Taichung 407, TAIWAN

synchronization to manage requests or to receive results.

Currently two approaches are frequently employed to resolve the parallelism dilemma in web server architecture. Specifically, one uses the traditional HTTP-based server and further enhances it by the parallel feature, and the other makes use of the object-oriented paradigm and implements it through the concept of distributed objects.

Fig. 1 depicts the traditional distributed Internet server. It consists of one or more front-end machines running HTTP daemons and one or more server machines running the distributed system. Naturally, a fast network facility connects the server machines. Alternatively, Fig. 2 illustrates a possible object oriented distributed Internet server. In this case, client objects connect to server objects by means of some underlying object framework like CORBA. Meanwhile, for performance reasons, multiple instances of the same server object may exist; and the runtime environment takes care of balancing incoming requests over the instances.

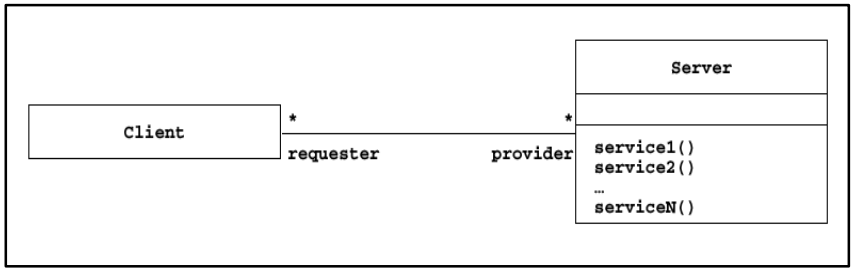

Figure 1. The client/server architecture in the UML class diagram

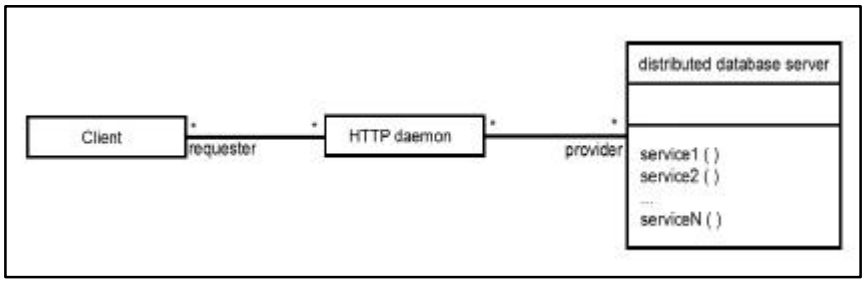

Figure 2. The distributed Internet server architecture in the UML class diagram

2.2 Hyperlink issues in Internet applications

Although the immediacy of the WWW creates immediate expectations of quality and rapid application delivery, the technical complexities of a website and variances in the browser make web testing and quality control much more difficult. Besides, more new technology such as Java script, ASP, ActiveX and CGI-bin scripts, etc., imply that it is more complicated to manage the quality of web sites. Consequently, website testing is still at the embryonic stage [1]. For instance, in e-commerce applications we are either building data up or retrieving data from a database. How do we know the retrieved data from a database is correct? If we input correctly, does the result satisfy what we expect?

In recent years, many studies have investigated the issue of quality analysis of web applications. In particular, reliability in web sites has attracted much more attention than before. Certain reliability indices are discussed in software quality assessment [2]. With more and more web sites, the emerging generation of critical Internet and web technologies is likely to require very high levels of reliability, and to explore its application for distributed computing technologies in the web environment.

2.3 Survey of current research

Theoretically, Yang and et al. [3] proposed an object-oriented architecture to construct web application testing environments by extending the well evaluated, traditional software testing architecture and applying some design patterns. The architecture contains six subsystems, and testing processes such as test case generation can be achieved with the cooperation of these subsystems. However, the prototype web-testing environment is still in its research stage, without a very high feasibility.

Also, there are not many web-testing tools to be found. Here, we present two popular web-testing tools. The first one is Astra, which is produced by Mercury Interactive Inc. and consists of three parts: LoadTest, QuickTest and SiteManager [4]. LoadTest is used to test scalability and performance of web applications, while SiteManager is a visual website management tool that automatically schedules and performs scans of an entire website and creates complete visual maps of the site. Finally QuickTest is designed to test websites by organizing interactive customizable tests. To employ Astra for web testing, a tester first clicks the hyperlinks or components of interest via a browser, and then QuickTest records those browsing paths automatically, which accordingly constitute the desired test cases to be executed.

Another testing tool is e-TESTER, developed by RSW Software, Inc [5]. With the aid of visual technology, a tester creates his visual script by simply interacting with the application in a browser. Afterward, e-TESTER automatically generates the actual navigation map of the tester's web application, which is then considered as a comprehensive test baseline for the entire application in order to validate the links, objects, images and text on the web pages.

In summary, since web applications have so many dependencies on other parts of the system, such as hyperlinks, databases, and network load, web-site testing becomes more complicated and difficult. In the following, we investigate the application of statistical usage testing to such a domain.

3. Background of software quality evaluation

3.1 Statistical usage testing

Recently statistical usage testing [6,7,8,9] has been justified and widely applied to software quality evaluation. Conceptually, the rationale of statistical usage testing lies in the fact that the failures which occur most frequently in practical use will be found early during the test cycle.

In essence, the main benefit of statistical usage testing is that it makes use of statistical inference techniques to compute probabilistic properties of the testing process, such as reliability, or mean time to failure (MTTF) for software quality evaluation.

3.2 Markovian usage model

Before performing statistical usage testing, a usage model must first be developed to represent all possible events in system use and their probabilities of occurrence. Essentially, a usage model briefly consists of possible stimuli, possible sequencing of stimuli, and expected responses. Interfaces and requirements are usually simplified or clarified when the model is reviewed systematically in the context of operational use. Actually, it works as an external view of the system specification that is readily understandable by system engineers, developers, clients, and end users.

Furthermore, we assume that an operational use is a skeleton for the intended use of the software in an intended environment. Thus, all possible operational uses of a software system

will constitute a population with a huge size. If a usage sample of test scenarios is drawn statistically from the usage population, performance on this sample may then be used as a basis for the evaluation of software quality. In addition, quantitative criteria about completion of the testing process may be obtained based on analyzing the usage model.

Whittaker suggests [10] that software testing is rather suitable to be treated as a stochastic process and, its usage can be modeled by a finite state, discrete parameter, time homogeneous, irreducible Markov chain.

In this paper, the Markov approach will be employed for scenario modeling [11]. Under the Internet environment, all possible navigation paths are first formulated to represent a scenarios model with the Markov chain property that is then analyzed and used to generate a test script file automatically.

3.3 Software reliability certification

In general, the two most common methods of system reliability evaluation are the combination modeling and Markov-state modeling approaches [12]. Combinational reliability models allow the reliability of a system to be calculated from the reliability of its component parts. Most techniques for the prediction of software reliability are based on an assessment of the techniques used to develop and test the software [13,14]. However, the combination modeling approach is frequently restricted to the analysis of random component failures.

Alternatively, the Markov modeling technique represents various operation states that appear in the history of software execution as either the discrete or the continuous Markov model [2], and analytic results derived from generic Markov chain theory are ready to analyze and certify software reliability indices.

Correspondingly, software reliability may be computed as the probability that an arbitrarily selected sequence of operations of the software will not cause a failure. Technically, it is the probability that an executing path beginning with the initial state and ending with the first occurrence of the terminating state will not contain a failure state [6]. In this paper, we employ the philosophy of the Markov modeling approach.

4. Automatic process of evaluation of link validity

4.1 Framework of automatic testing

For our system framework of automatic testing, we divide the complete function into five subsystems, which are represented by using the UML (Universal Modeling Language) class diagram as in Fig. 3.

In Fig.3, there is only one actor, i.e. tester, who negotiates the details of user requirements and specifications with the client. The responsibility of the tester is to ensure that the purpose of system requirements of a client has been both clearly described and completely satisfied. Thus, he must first build a scenario model by manually eliciting user requirements and specifications. Then, other testing procedures are followed in sequence. In our framework, they are designed and implemented as four subsystems that can execute automatically. These subsystems consist of navigation structure builder, navigation script generator, navigation script executor and evaluator, which are discussed briefly in the following.

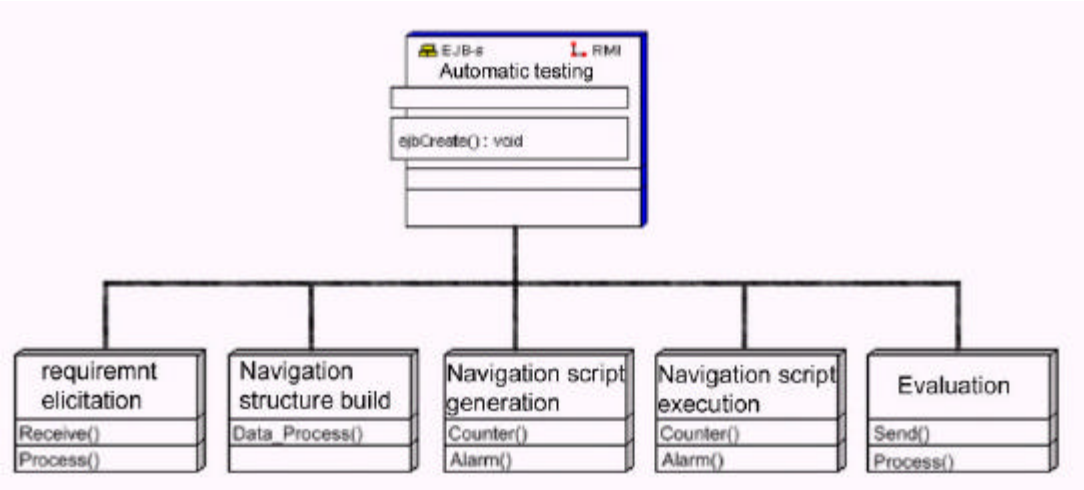

Figure 3. The framework of automatic testing in the UML (class diagram)

4.2 Evaluation process

4.2.1 Requirement elicitation

In general, requirement elicitation is an iterative process of communication among a tester, clients, and users for defining a new system. From the user's viewpoint, we depict user

requirements by definitely describing a system and its interaction with the surrounding environment such as the clients, their work process, and other systems, etc.

First, the client and tester identify a problem area and define a new system that addresses the problems encountered. Tester clarifies the system requirements and specifications with the client. Then user specifications behave as an agreement between the client and tester, and are continuously validated by the client and tester.

4.2.2 Navigation structure building

To begin statistical usage testing, a navigation structure must first be built to display all possible navigation paths for a web site. However, it is somewhat difficult to collect all complete browsing paths as the potential paths may diverge without termination. In this research, we design a module *Navigation viewer* to build the hierarchical navigation structure level by level, by scanning the source code of each homepage to catch all URL' s (Universal Resource Locator) of emergent hyperlinks from the root page.

4.2.3 Navigation scripts generation

Once the navigation structure is established, a browsing sample may then be drawn statistically to construct test scripts. That is, take a randomly selected path beginning from the initial state and ending with the terminating state.

Without loss of generality, the format of these generated test scripts with N nodes is assumed to be as shown in Table 1. Specifically, node A is the initial state, which is usually the homepage of the tested website. Node B is associated with node A (i.e., the homepage) and node C is further associated with node B, and so on. The last step--n^{th} step--means that this test script has reached the terminating node in the usage model. Naturally any test script may not transverse all nodes inside the usage model. Thus, the larger the number of test scripts generated and executed, the higher the test coverage of states or arcs of the model swept after their execution.

In this paper, test scripts are generated automatically by using the package ToolCertify [15].

Table 1. The format of navigation scripts

Navigation script: serial number	
Test step	Description
1	Software Not Invoked 1. click to node A node A
2	2. click to node B node B
3	3. click to node C node C
.	.
n	n. click to Software Terminated Software Terminated

4.2.4 Navigation Script execution

Implicitly, a generated test script constitutes one arbitrary usage path that consists of several browsing links starting from the initial state (i.e., homepage) of a website to its termination state (i.e., exit node). To execute one test script, each intermediate test step listed in Table 1 must be visited and checked to see if its function worked as intended for each test script. In this research, a subsystem *Script executor* is designed, by using the JAVA language, to automatically execute these generated navigation scripts as illustrated in Fig. 4.

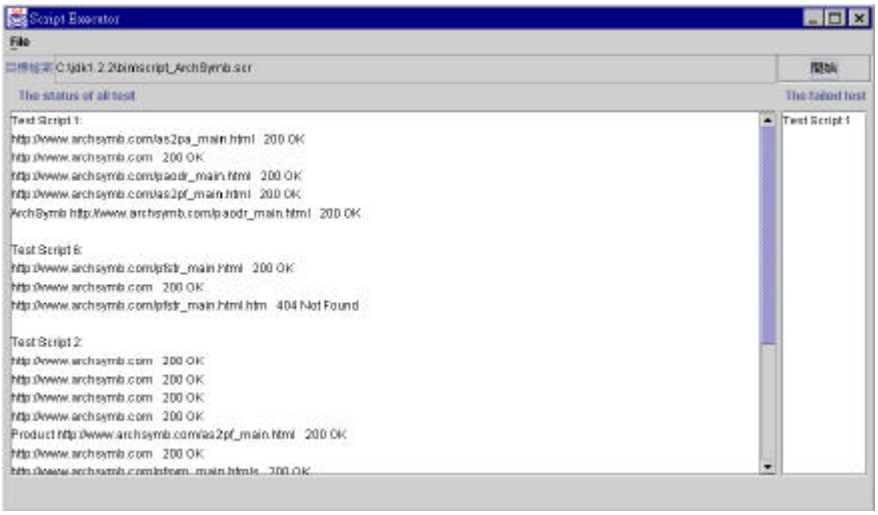

Figure 4. The *Script executor* executes navigation script automatically

4.2.5 Evaluation

After performing the test scripts execution, failure information like inter-failure data may be collected to evaluate the hyperlink validity of the web application in the following respects:

1. Failure analysis

From our observations, most failure information on hyperlink validity may be classified as: “HTTP 404 Not Found,” “HTTP Server at Compressed .com:8080 Replies: HTTP/1.0 500 Error from Proxy,” “HTTP 403 Forbidden/Access Denied, ” “Site Unavailable”, etc.

Normally the message “HTTP 404 Not Found” occurs when the browser cannot find the document that has been requested on the host computer. In other words, this failure occurs either because the request that is initiated by a client cannot be executed in the server, or because the required file is absent in that server.

On the other hand, as to the error message “Site Unavailable,” the most common reason we found is that there are too many people trying to visit that same website so that web communication is congested. In addition, the site could also be down for maintenance, or there is a network problem, or that site no longer exists. Meanwhile, browsing an incorrect web address could also cause such a message.

2. Evaluation result

As stated previously, since both the navigation structure and the generated test scenarios may be represented by a Markov chain, analytic results derived from generic Markov chain theory are ready to analyze and certify software reliability index.

As presented in Section 3, moreover, software reliability is computed as the probability that an executing path beginning from the initial state and ending with the first occurrence of the terminating state will not contain a failure state. However, under the web environment, software reliability is interpreted as the probability that a navigation path beginning from the homepage to the browsing end will not encounter a failure link.

In this research, reliability analysis is performed by the package ToolCertify with additional coverage information.

5. Applying the proposed method to an Internet example

5.1 General description

The web site ArchSymb [16], as shown in Fig. 5, is chosen to illustrate how to apply the proposed method. Since the ArchSymb web site uses many structural links, its navigation structure is hierarchical [17], which can also be observed from Fig. 6. The hierarchical structure possesses the advantage that it may retain the original structure of information contained in a hypermedia system. In addition, it may assist users in looking up quickly the desired nodes through browsing the structured navigation paths.

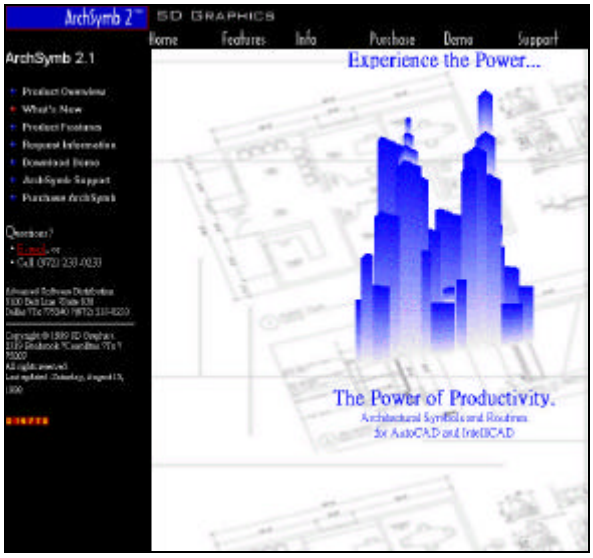

Figure 5. ArchSymb example homepage

5.2 Scenarios Model

As pointed out previously in this research, we make use of the subsystem *Navigation Viewer* to establish each level of the navigation map by assuming the root (homepage) is level 1, as shown in Fig. 6. The whole tree structure of ArchSymb contains five levels, and level two consists of 13 nodes. Consider every hypermedia link as a browsing state and the homepage as the starting state. While browsing ArchSymb, the user clicks every possible link from the starting node until he stops. Consequently, these possible operational patterns form a Markov chain.

On the other hand, although its navigation map is hierarchical, some links of ArchSymb

have no termination at all because they expand endlessly. To avoid cognitive overload and spaghetti links, we view some hypermedia links as termination nodes at some level if this is adequate. For instance, square nodes in Fig. 7 imply these nodes that are truly terminated, while triangle nodes indicate that they actually have more emergent links, but are considered as termination nodes for simplicity.

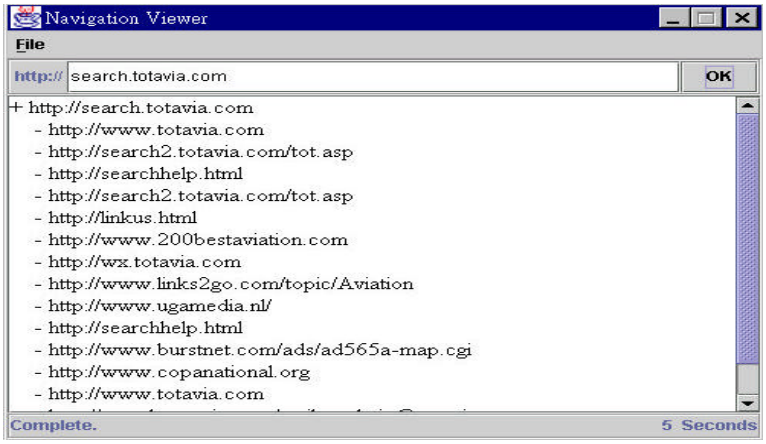

Figure 6. A navigation view of the ArchSymb example

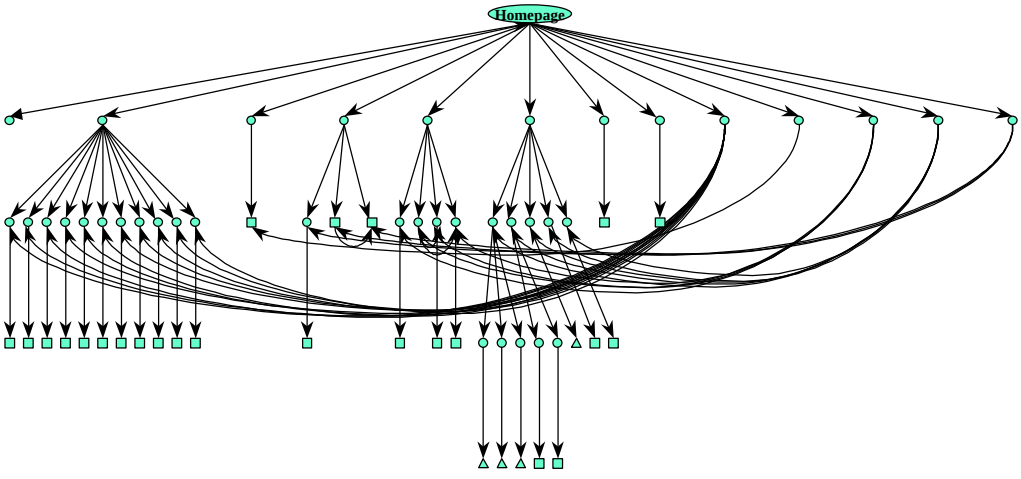

Figure 7. The scenarios model of the ArchSymb example

5.3 Model analysis

The usage model built for the ArchSymb web site includes 61 nodes in all, which has the fundamental impact of link evaluation on web applications. With the aid of the ToolCertify evaluation tool [15], we summarize the analyzed result as in Table 2.

In Table 2, the second row “Number of active hyperlinks” represents the numeric count of arcs in the navigation structure model, while the third row “Expected script length” indicates the expected number of hyperlinks in a typical test script. In other words, it provides the average script length among a large number of random scripts from the model. On the other hand, the fourth row “Least likely node-coverage expected at” shows the expected number of test scripts required to cover the minimal set of nodes. And the final row “Least likely hyperlink-coverage expected at” shows the expected number of test scripts required to cover the minimal set of hyperlinks.

Table 2. Analysis report of the ArchSymb navigation structure

Number of existing nodes	61
Number of active hyperlinks	114
Expected script length	5.5722
Least likely node-coverage expected at	89.9999
Least likely hyperlink-coverage expected at	132.0001

5.4 Evaluation analysis

1. Navigation Scripts

Based on Table 2, the minimal number of test scripts to test the web site of interest is initially estimated as 133 (132.0001). In practical testing, however, complete coverage of the browsing links was not reached until 335 non-failure executions of test scripts, as shown in Table 4. Unfortunately, after executing these 335 generated test scripts, 10 link errors were found. Thus, complete coverage of actual links was accomplished with 503 test scripts, as illustrated in the same table.

2. Failure analysis

In summary, the failure report of the 10 link errors is analyzed as in Table 3. Because only one failure was found, the values of probability of occurrence are all the same. In addition, the “Mean first passage” column, in terms of Markov chain theory, denotes the expected number of

hyperlinks for the first occurrence of a certain node from the homepage.

On observing the 10 distinct failure paths among the 503 testing browsing paths, one source of link errors was identified. In more detail, it is noted from the failure report that all 10 failures links were emerging from the node *Contact us* followed by the node *Support*. That is, any path emergent from the node *Support* will fail in this example. In fact, if we click this node directly, we see the error message like "HTTP 404 Not Found" on a browser.

Table 3. Failure analysis of the ArchSymb example

Failure ID	Mean first passage	Probability of occurrence
1	2796.999512	0.001988
2	2796.999512	0.001988
3	2796.999512	0.001988
4	2796.999512	0.001988
5	2796.999512	0.001988
6	2796.999512	0.001988
7	2796.999512	0.001988
8	2796.999512	0.001988
9	2796.999512	0.001988
10	2796.999268	0.001988

3. Evolution result

By the analysis result derived from the Markov usage model, partial certification results may be summarized as in Table 4, with 95% and 99% confidence intervals (MTTF denotes the Mean Time To Failure and R denotes the system reliability). Table 4 also illustrates how, as the number of test scripts increased to 503, the test coverage for both states and transitions reached the limit. Meanwhile, as seen from Fig. 8, the web reliability increases as the number of non-failure test scripts increases. Some corresponding test scripts are shown in Table 5.

Table 4. Evaluation result for the ArchSymb example

Script #	Result	MTTF	R	C=95%	C=99%	% State coverage	% Arc coverage
1	Pass	NA	1	0	0	9.836065	4.385965
28	Fail	28.00001	0.964286	0.012756	0.016765	72.13114	56.14035
31	Pass	30.99997	0.967742	0.011541	0.015168	73.77049	57.89474
125	Pass	24.99999	0.96	0.004824	0.00634	98.36066	92.10526
149	Fail	21.28573	0.95302	0.004325	0.005684	98.36066	92.10526
156	Pass	22.28571	0.955128	0.004132	0.005431	98.36066	92.10526
187	Pass	26.71429	0.962567	0.003452	0.004537	98.36066	92.10526
218	Fail	27.24999	0.963303	0.003021	0.003971	98.36066	92.98246
249	Pass	31.125	0.967871	0.002647	0.003479	98.36066	92.98246
337	Pass	33.70002	0.970326	0.002013	0.002646	98.36066	97.36842
347	Pass	34.70002	0.971182	0.001956	0.00257	100	99.1228
404	Pass	40.39996	0.975248	0.00168	0.002208	100	99.1228
466	Pass	46.60001	0.978541	0.001457	0.001915	100	99.1228
503	Pass	50.30001	0.980119	0.00135	0.001775	100	100

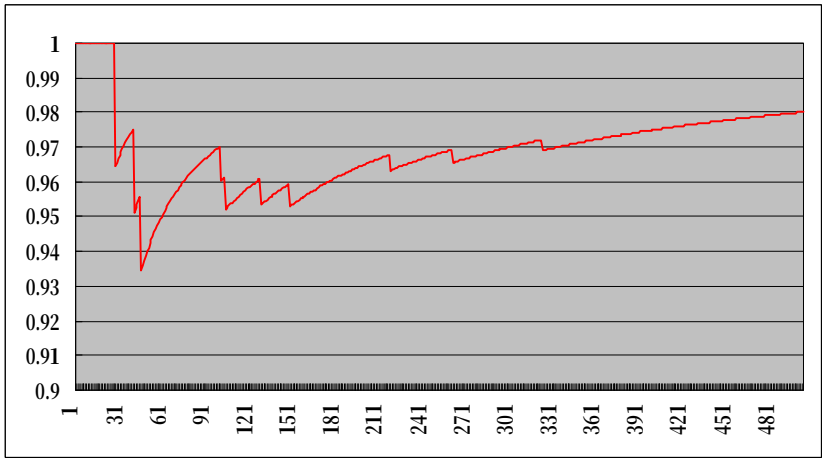

Figure 8. Reliability with 503 test scripts

Table 5. Test scripts of the ArchSymb example

Test Script: 28		Test Script: 41	
Step	Description	Step	Description
1	Software Not Invoked 1. to homepage homepage	1	Software Not Invoked 1. to homepage homepage
2	2. click to Support Support	2	2. click to Support Support
3	3. click to Contact us Contact us	3	3. click to Contact us Contact us
4	4. click to Software Terminated Software Terminated	4	4. click to Software Terminated Software Terminated
Test Script: 101		Test Script: 104	
Step	Description	Step	Description
1	Software Not Invoked 1. to homepage homepage	1	Software Not Invoked 1. to homepage homepage
2	2. click to Support Support	2	2. click to ArchSymb Support ArchSymb Support
3	3. click to Contact us Contact us	3	3. click to Contact us Contact us
4	4. click to Software Terminated Software Terminated	4	4. click to Software Terminated Software Terminated

6. Conclusions

In this paper, the issue of web site quality evaluation is investigated, and an automatic mechanism of testing hyperlinks is developed.

To evaluate all possible navigation links, the rationale of statistical usage testing is employed. Its philosophy essentially suggests organizing a complete and systemic method, rather than the general *ad hoc* approach. One of the drawbacks of an *ad hoc* approach is that it cannot cover all scenarios of navigation script. In contrast, the suggested approach based on statistical usage testing may provide both complete testing coverage and quantitative analysis.

The proposed automatic framework has been shown to be effective for certifying quickly link validity on the Internet. It has many benefits such as: helping in test planning, generating test scripts automatically, and navigating script execution automatically and evaluating hyperlinks automatically. Hence, the proposed testing framework is more practical and efficient than other methods of testing using an *ad hoc* approach.

Moreover, web testing must validate not only hyperlinks but also web technologies. In future research, other common techniques in e-business websites, such as ASP, forms, submit buttons, CGI-script, etc., will be investigated and evaluated. Hereafter, we plan to set up a web-based test environment for automatically validating e-business applications.

Acknowledgements

This research is supported by the National Science Council, ROC (Grant NSC89-2213-E-029-00).

References

[1] Miller, E. (1999) "The WebSite Quality Challenge" (http://www.soft.com/Products/Web/Technology/website.testing.html), Software Research, Inc..

[2] Tian, J. (1999) "Measurement and continuous improvement of software reliability throughout software life-cycle," *Journal of Systems and Software* **47**, pp. 189-195.

[3] Yang, J.T., Huang, J.L., Wang, F.J., and Chu, W.C. (1999) "An object-oriented architecture supporting web application testing," *23rd Annual International Computer Software and Applications Conference*, October, pp. 2-7.

[4] Mercury Interactive Inc. Website: http://www.merc-int.com/, 2000.

[5] RSW Software Inc. Website: http://www.rswsoftware.com/, 2000.

[6] Whittaker, J. A., and Thomason, M.G. (1994) "A Markov Chain Model for Statistical Software Testing," *IEEE Transactions on Software Engineering* **20**(10), October, pp. 812-824.

[7] Walton, G.H., Poore, J.H., and Trammell, C.J. (1995) "Statistical Testing of Software Based on a Usage Model," *Software Practice and Experience* **5**(1), January 2, pp. 97-108.

[8] Chang, Wen-Kui (1997) "A Quadratic Programming Approach to Usage Modeling for Software Reliability Certification," *Tunghai Journal* **38**, July, pp. 65-78.

[9] Chang, Wen-Kui, Twu, Stephen, and Teng, William (1999) "Ensuring Functional Test Coverage For Avionics Control Applications Through Statistical Usage Testing," *FESMA'99 - 2nd European Software Measurement Conference*, Amsterdam, The Netherlands, October 4-8, pp. 261-268.

[10] Whittaker, J. A., and Poore, J.H. (1993) "Markov Analysis of Software Specifications." ACM Transactions on Software Engineering and Methodology, vol. 2(1), pp. 93-106.

[11] Chang, Wen-Kui, Wang, Che-Po, and Fu, Ching-Chun (1999) "A Study on Usage

Modeling for Software Reliability Certification via Prototyping Simulation," *The 3rd Symposium on Reliability and Maintainability*, Taiwan, October, pp.279-285.

[12] Storey, N. (1996) *Safety-Critical Computer Systems*. Addison Wesley Longman, ISBN: 0-201-42787-7.

[13] Lyu, M. (editor) (1996) *Handbook Of Software Reliability Engineering*, IEEE Computer Society Press, McGraw-Hill, New York.

[14] Pham, H. (2000) *Software Reliability*, Springer-Verlag, ISBN: 981-3083-84-0.

[15] Q-Labs: ToolCertify User Guide, Version 4.0, 1999.

[16] ArchSymb Websites: http://www.archsymb.com/, 2000.

網際網路環境架構下超連結正確性驗證的自動測試

張文貴* 洪昕凱*

摘 要

由於全球資訊網的便利性，導致它的應用蓬勃發展；然而卻因網站技術的困難度與瀏覽器的差異性逐漸增加，網站的測試與品質管制也愈來愈困難。本論文將針對網際網路環境架構下的超連結正確性驗證，提出自動測試的系統架構；首先我們將扼要地探討網路應用系統的一些特質，以及討論網路品質的相關議題與其性能指標，接著研究如何運用軟體使用測試的技術在網路的領域方面，並發展一套整體的驗證機制。它主要的觀念，在將所有可能的瀏覽路徑，透過馬可夫鏈的特性，表示為一個案例模式，並據以用來自動產生測試網站系統的路徑案例檔。再追蹤這些路徑案例的超連結執行結果，我們即可應用馬可夫鏈理論的分析運算，處理超連結的驗證評估工作。這個自動測試的機制不僅能有效且系統性的驗證超連結，而且在協助規劃測試過程方面，更為有效。

關鍵詞：網際網路、軟體性能測試、軟體使用測試、馬可夫鏈、物件導向法。

* 東海大學資訊系