A practice on performance testing for web-based systems Hyperlink testing for web-based system

*Wen-Kui Chang , *Shing-Kai Hon
*Dept. of Computer & Information Sciences, Tunghai University, Taichung, Taiwan
Software Engineering Laboratory, P.O. Box 5-809 Tunghai University, Taichung, Taiwan 407
tel.: +886-4-3598915, fax.: +886-4-3591567, email: wkc@mail.thu.edu.tw

Abstract

This paper investigates the issue of performance testing on web browsing environments. Among the typical non-functional characteristics, index of link validity will be deeply explored. A framework to certify link correctness in web site is proposed. All possible navigation paths are first formulated to represent a usage model with the Markov chain property, which is then used to generate test script file statistically. With collecting any existing failure information followed by tracing these testing browsed paths, certification analysis may be performed by applying Markov chain theory. The certification result will yield some significant information such as: test coverage, reliability measure, confidence interval, etc. The proposed mechanism may provide not only completed but also systemic methodologies to find any linking errors and other web technologies errors. Besides, an actual practice of the proposed approach to a web-based system will be demonstrated quantitatively through a certification tool.

Key Words : web testing, statistical usage testing, usage model, Markov chain

1. Introduction

Web-based systems are increasingly developed in many application domains and most were implemented by the ad hoc approach nowadays. Nevertheless, various quality metrics become progressively more difficult to measure and mange as the developing application size and scope grows.

Among the typical non-functional characteristics, factors of link validity will be deeply investigated in this research.

Essentially link validity includes various characteristics such as correctness, relevance, completeness and integrity implicitly, which will deeply affect the effectiveness and efficiency in information retrieval and browsing. However, in most web applications, it is unlikely that application developers may validate all possible navigation paths. In this paper, a framework of quantitative certification on link validity will be proposed to ensure that all linked paths provide consistent and reasonable information streams and appropriate contexts.

In this research, the rational for statistical usage testing is examined and employed to certify links for a web site. Under the web-browsing environments, all possible navigation paths are first formulated to represent a usage model with the Markov chain property that is then analyzed and used to generate test script file statistically.

The proposed mechanism is not only systematic on efficient highly hyperlinks, but also efficient highly for those complex information structures. In addition, a real application of the proposed approach to a web application will also be demonstrated quantitatively through a certification tool ToolCertify.

ToolCertify is one piece of the technology involved in certifying software. The focus of the ToolCertify is to guide engineers in the performance of certification tasks and many tedious testing tasks.

The theory of ToolCertify based on the Usage Model. According Usage Model, it cans analysis the Usage Model, generate the report and certification Report. Using ToolCertify, we have many values of evolution.

2. Web-based systems

2.1 Features

As the growth of Internet and high-bandwidth networks continues to accelerate and many web sites contain multimedia information, more users want to create and browse links from which non-text information such as image, sound, and video usually exists. Projects such as Hyper-G[1], Microcosm[2] and AHM[3], etc. have explored the most effective ways to provide such support. For example, Microcosm is an open and extensible hypermedia architecture to support large-scale development through use of separate linkbase, which manages various links types.

Essentially, accessibility of web content is one of the most significant quality indexes for web-based system performance.

Furthermore, useful and accessible web-based systems will be enhanced to alternate client agents in the near future. They are web-aware technologies that allow web-based systems to be accessed in a variety of unusual ways, such as, a hand-held computer or intelligent agents. Agents are being implemented using proxy server and Java applets. Besides, agents can also be used to constantly monitor the information environment, to update links[4]. Web designers will begin today to anticipate and accommodate these alternate client technologies.

2.2 Quality issue

Certainly, WWW and web site pose unique software testing challenges. Although the immediacy of the WWW creates immediate expectations of quality and rapid application

delivery, the technical complexities of a web site and variances on the browser systems make testing and quality control much more difficult[5], such as JavaScript, ActiveX and CGI-Bin Scripts. More new technologies imply that it is harder to control quality of web site and quality process must be able to support these sites.

However, web site testing is usually carried out by the ad hoc approach. Nowadays, web site test is still at the embryonic stage, as methodology for validating web site application has not been well developed, even tools. In particular, confirming validity of what is tested is the key to assuring web site quality, which is the most difficult challenge of all[6].

Moreover, execution scenarios of web application are usually not repeatable and numerous factors may influence it. For instance, heavy load in a network would generate a long response-time and lots of throughput. Accordingly, those factors make web site test extremely complicated and difficult.

3. Software quality certification

3.1 Rationale

3.1.1 Statistical usage testing

Recently statistical usage testing[7,8,9,10] has been justified and widely applied to software quality certification. Conceptually, the rationale of statistical usage testing lies in the fact that the failures occurred 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 (MTBF) for software quality certification.

3.1.2 Software usage model

Fundamentally, a software usage model characterizes various operational uses of a software system. Suppose that an operational use is a skeleton for the intended use of the software in an intended environment. Then, all possible operational uses of a software system will constitute a population with a huge size. If a usage sample of test cases is drawn statistically from this usage population, performance on the sample may then be used as a basis for the evaluation of software quality.

Whittaker suggests[11] that software testing is rather suitable to be treated as a stochastic process and, its usage be modeled by a finite state, discrete parameter, time homogeneous, irreducible Markov chain. In this paper, Markov approach will be employed for usage modeling[12].

3.1.3 Markovian usage model

Represented by the notation of Markov

chain, a usage model consists of all usage states that are connected by links. These links indicate all possible stimuli and responses with a probability index. The probability represents the likelihood of choosing one link from a usage state to the other. Furthermore, test scenarios, generated randomly as a sample of the population, shows some possible usage paths that will traverse the usage model from the start state to the termination state.

Furthermore, the generated test scenarios are formulated as a Markov chain by the following facts[12]:

1) Occurrence of the current state depending on the previous state only.
2) All usage states are incompatibility.
3) All probabilities emerged from each state are summed to one.

3.2 Validation process

In principle, software quality certification by the usage testing approach may be performed in the following[8,13]:

Algorithm for validation

```
   */
   Use Case(n) imply number of Use Case
   Test Case(n) imply number of Test Case
   Build_usage_model imply building a usage model that defines all possible events and
   their transition distribution
   Generte_TestCase(time) imply cases statistically by the associated distribution
   Execute_TestCases(time) imply Executing the test cases.
   collect_tested_data(time) imply collecting performance information and inter-failure data.
   certify_by_evolution_model(time) imply certifying the software by the reliability evaluation model.
   R_accept imply users define value of reliability that can be accepted
   R_outcome imply real value of reliability
   Time imply number of iteration to execute
   */

public class WebTesting
{
      public build_usage_model (time):
      public Create UseCase (int n):
          For (int time = 1 : time < n : time++):
          {
              public void generte_TestCase (time)
              public void execute_TestCase (time)
              public void collect_tested_data (time)
              public void certify_by_evolution_model (time)
          }
          if (R_outcome < R_accept)
              public void WebTesting ():
          than
              break:
   }
```

1) Building a usage model that defines all possible events and their transition distribution.

2) Generating test cases statistically by the associated distribution.

3) Executing the test cases.

4) Collecting performance information and inter-failure data.

5) Certifying the software by the reliability evaluation model.

3.3 Algorithm for validation

In more detail, the above process can be represented by the following algorithm to validate the web site.

4. Demonstration on link validity for web applications

4.1 General description

A web site ArchSymb[14] is used to perform a web site test, as shown in Fig. 1. Usually information structures can be denoted in various ways, such as linear, hierarchical, network and matrix[1]. Any web application can have more than one kind of information structure. In this case, information structure for ArchSymb is hierarchical, which can be observed from the navigation structure as shown in Fig. 2. A hierarchical structure most often reflects structural links. The advantage of hierarchical structure lies that it may retain original structure of information contained in hypermedia system. Besides, since it is a commercial web site, links established by hierarchical structure may benefit users to browse the whole navigation paths to look up the desired products. Also, user can go this web site's index and select point within information space to read in the same way as is done in the field.

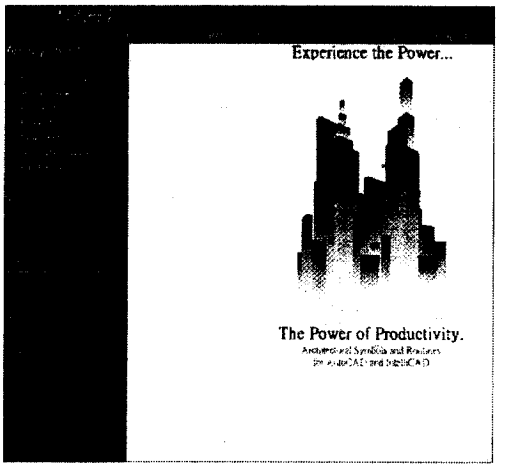

Fig. 1 The ArchSymb example homepage

4.2 Usage Model

To begin with statistical usage testing, a navigation structure must first be built to present all possible navigation paths for a web site. However, it is somewhat difficult to collect all complete browsing paths as the potential paths may go divergently without termination. Fortunately some tools, such as Microsoft FrontPage[15], etc., may be use to create all links, all buttons, and FORM-content tests passages. In addition, creation

of macro structure involves the integration of nodes into high cohesion. This integration happens through linking of nodes together.

In this research, a navigation map established by FrontPage is shown in Fig. 2.

Suppose that every hypermedia link as a browsing state and the homepage as the starting state. While using ArchSymb, the user clicks every possible hypermedia link from the starting node until he stops. As a result, these possible operational patterns formulate a Markov chain.

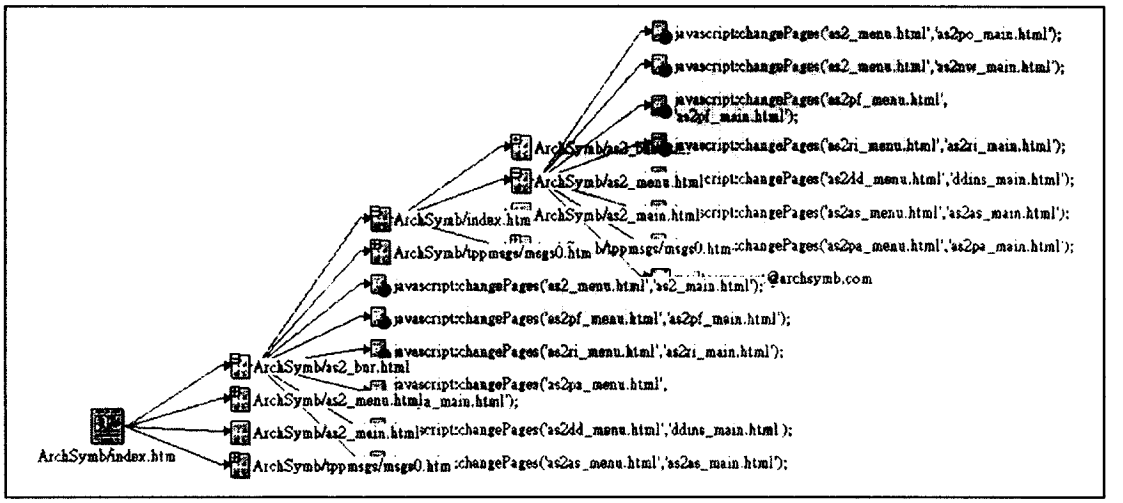

Fig. 2 The navigation map for ArchSymb example

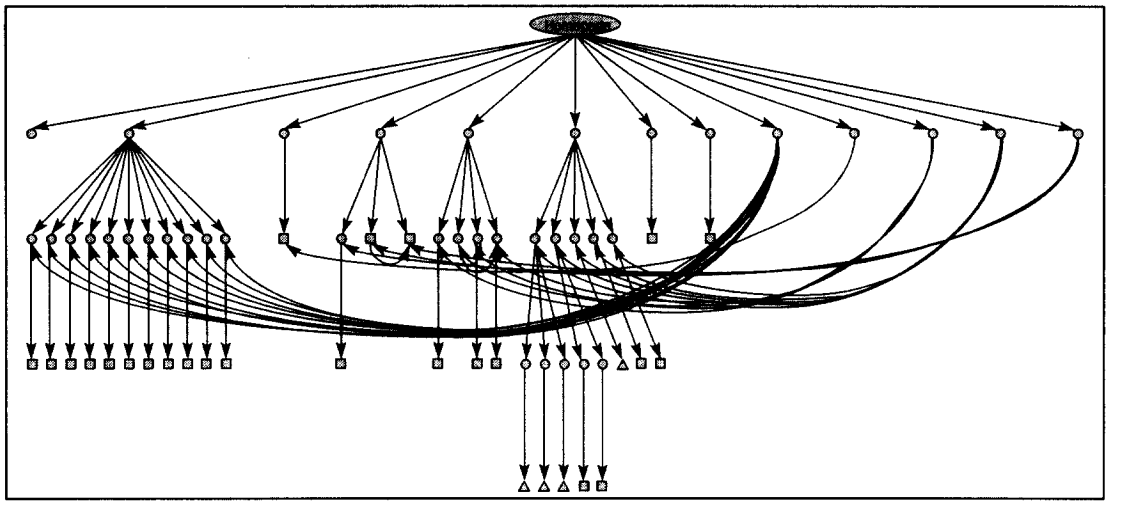

Fig. 3 The usage model of ArchSymb example

As for illustration, the navigation map can be further restructured as a usage model in Fig 3. However, some hypermedia links of ArchSymb have no termination, that is, the links expand endless. Such situation may usually happen to many other web sites. In solving it, the whole model may be considered as a tree structure, hypothesizing the root (Homepage) is level 1. The whole tree structure of ArchSymb contains 5 levels.

Besides, to avoid cognitive overload and spaghetti links, we view some hypermedia links as termination node at some level. In figure 3, square nodes imply true termination node. Triangle nodes imply it still have out coming linking, but view it as termination.

4.3 Model analysis

The built usage model for ArchSymb web site includes 61 states in all, which plays a fundamental framework for link certification on web applications. With the aid of ToolCertify certification tool[16] the analyzed report may be summarized as the Table 1.

In Table 1, Number of Active Arcs is the numeric count of arcs in the model and the value is 114. Expected Script Length is the expected number of state transitions in a typical random test script and the value is 5.5722. The expected number of test scripts with the complete state coverage (100%) is 90 (89.9999), while that for the complete transition coverage is 133(132.0001).

Table 1 Analysis report for the ArchSymb navigation structure

Number Of Nondeleted States	61
Number Of Active Arcs	114
Expected Script Length	5.5722
Least Likely State Coverage Expected At	89.9999
Least Likely Transition Coverage Expected At	132.0001

4.4 Certification

1) Test scripts

Based on Table 1, the required number of test scripts is theoretically estimated as 133 (132.0001). However, once executing these generated test scripts, a kind of link errors was found as listed in the Table 2. Thus, actual transitions for the complete coverage

Table 2 Failure analysis for the ArchSymb example

Failure number	Mean First Passage	Long Run Probability	Probability of Occurrence
1	2796.999512	0.000357	0.001988
2	2796.999512	0.000357	0.001988
3	2796.999512	0.000357	0.001988
4	2796.999512	0.000357	0.001988
5	2796.999512	0.000357	0.001988
6	2796.999512	0.000357	0.001988
7	2796.999512	0.000357	0.001988
8	2796.999512	0.000357	0.001988
9	2796.999512	0.000357	0.001988
10	2796.999268	0.000357	0.001988

are increased to 503 test scripts.

2) Failure analysis

After performing link executions manually for 503 test scripts, one source of link errors was found. On observing the failure information, it is noted that all failure links came from the node [Contact us]. The failure path is on class [Support]. In class [Support], there is a failure hyperlinks to [Contact us] that will cause error message, such "HTTP 404 Not Found" in browser.

In this case this failure was the principal causes that made 10 distinct failures among the 503 testing browsing paths. In following Table 2, because just one failure is found, the value of probability of occurrence is all the same. Mean First Passage implies the expected number of state transitions until the first appearance of the corresponding state starting from the initial state in the Markov chain.

3) Certification result

By the analysis result derived from the Markov usage model, partial certification may be summarized as in the Table 3, with 99% confidence interval. From table 3, it appeared when the number of test scripts reached 503, the coverage for transitions is 100%. In addition, as the number of test scripts increase, the reliability is increasing accordingly. Some corresponding test scripts are shown in Table 4.

We explain each of the columns in Table 4. Value of Confidence (99%), it presented confidence intervals with 99%. Arcs Certified imply number of Navigation Script that can reach 100% coverage of all relationship in Usage Model. MTTF is the expected number of uses until the user is expected to encounter the first failure. Reliability is the

Table 3 Certification result for the ArchSymb example

Serial Script	Confidence (99%)	Arcs Certified (%)	MTTF	Reliability	Outcome
28	0.016765	56.14035	28.00001	0.964286	Fail
41	0.014813	62.2807	20.49999	0.951219	Fail
46	0.015314	65.78947	15.33334	0.934783	Fail
101	0.00746	91.22807	25.25001	0.960396	Fail
104	0.007606	91.22807	20.8	0.951923	Fail
129	0.006369	92.10526	21.49999	0.953488	Fail
149	0.005684	92.10526	21.28573	0.95302	Fail
218	0.003971	92.98246	27.24999	0.963303	Fail
261	0.003371	93.85965	28.99999	0.965517	Fail
323	0.00276	96.49123	32.30003	0.96904	Fail
346	0.002577	97.36842	34.59998	0.971098	Pass
347	0.00257	99.1228	34.70002	0.971182	Pass
502	0.001778	99.1228	50.20007	0.98008	Pass
503	0.001775	100	50.30001	0.980119	Pass

Table 4 Test scripts for 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: 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 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
Test Script: 218		Test Script: 261	
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

estimated reliability, which is defined as the probability that a single execution of the software will be failure free. As number of test script increase, the reliability is increase also. In value of Outcome, it has two types-Pass and Fail. It indicates the script passed without failure, a failure was observed and the script is unresolved.

4.5 Discussion

1) Challenges

Theoretically, as the number of states in a usage model is innumerable, the number of test scripts will become quite huge. In this research, we executed 503 test scripts manually and it took about 3.5 hours.

According to Table 1, every test script must happen 6(5.5722) steps on average.

Naturally, more test scripts will imply more execution time. To avoid this situation, automated testing of web site will be an opportunity and a challenge.

Another challenge for web testing comes from the web-based system itself. In fact, different browsers are sometimes endowed with different features and support different techniques. Consequently, complexity on web testing is greatly increased. Furthermore, performance of a web browser is highly correlated with various factors such as: operating systems, platforms, machine configurations, etc. For instance, testing on the different configuration combinations can quickly become a huge and insurmountable task.

2) Research in future

In general, web testing must validate not only hyperlinks but also web technologies, such as JAVA and Active X. In particular, e-business applications propagate with extreme rapidity in recent years. It is more important to control quality of e-business web site. E-business web site with poor quality such as: broken pages, faulty images, CGI-Bin error messages, etc. can always overload the user. Using CAPBAK/WEB tools[17] not only can validate hyperlinks, but also can validate form that E-business always uses. Validating E-business Web site is also our purpose in future.

5. Conclusions

For software quality certification, a framework of statistical usage testing is investigated in this paper, and a mechanism for certifying all possible navigation links is developed.

The rationale of statistical usage testing is a complete and systemic method, which is rather than the general ad hoc approaches. It may provide complete testing coverage and quantitative analysis as well. Using ad hoc approach, which cannot assurance that all navigation paths and components have been tested. It also cannot provide accurate data to validate a web-based system.

The proposed mechanism has been shown to be effective for certifying quickly link validity in web site applications. It has many benefits such as helping in testing plan and allocating testing resource, generating test scripts automatically, and reaching the maximal testing coverage. Hence, the proposed testing framework will be more practical and efficient than an exhaustive testing.

References

1. Maurer H, Hyper-G now Hyperwaves: The Next Generation Web Solution, Addison-Wesley, 1996.
2. Hall W, Davis H and Hutchings G, Rethinking Hypermedia: The Microcosm Approach Kluwer,

1996.

3. Hardman L, Bulterman D and van Rossum G: The Amsterdam Hypermedia Model: extending hypertext to support real multimedia Hypermedia Journal 5(1), 1993, pp. 47-69.
4. Lowe, David, Wendy Hall: "Hypermedia and the Web-An Engineering Approach," John Wiley and Sons, ISBN: 0471983128, 1999, pp. 60-65.
5. Miller, Edward : "WebSite Testing" (http://www.soft.com/Products/Web/Technology/website.quality.html), Software Research, Inc, 1999.
6. Miller, Edward : "The WebSite Quality Challenge" (http://www.soft.com/Products/Web/Technology/website.testing.html), Software Research, Inc, 1999.
7. Whittaker, J. A. and M. G. Thomason: "A Markov Chain Model for Statistical Software Testing." IEEE Transactions on Software Engineering, October 1994, 20 (10), pp. 812-824.
8. Walton, G. H., J. H. Poore and C. J. Trammell: "Statistical Testing of Software Based on a Usage Model." Software Practice and Experience, vol. January 1995, 5(1), pp. 97-108.
9. Chang, Wen-Kui: "A Quadratic Programming Approach to Usage Modeling for Software Reliability Certification," Tunghai Journal, July 1997, Vol. 38, pp. 65-78.
10. Chang, Wen-Kui, Stephen Twu and William Teng: "Ensuring Functional Test Coverage For Avionics Control Applications Through Statistical Usage Testing," FESMA '99 - 2nd European Software Measurement Conference 4-8 Amsterdam, The Netherlands, October 1999, pp. 261-268.
11. Whittaker, J. A. and J.H. Poore: "Markov Analysis of Software Specifications." ACM Transactions on Software Engineering and Methodology, vol. 2(1), 1993, pp. 93-106.
12. Chang, Wen-Kui, Che-Po Wang, and Ching-Chun Fu: "An Investigation on the Markovian Property of Usage Model for Software Quality Certification", Proc. of the Chinese Institute of Industrial Engineers National Conference, Taiwan, December 1999, pp. 129-135.
13. Chang, Wen-Kui, Che-Po Wang, and Ching-Chun Fu: "A Study on Usage Modeling for Software Reliability Certification via Prototyping Simulation," The 3rd Symposium on Reliability and Maintainability, Taiwan, October 1999, pp.279-285.
14. ArchSymb Web site: http://www. archsymb.com/, 2000.
15. Microsoft(R) FrontPage(R) 2000, Version4.0.2, 1999.
16. Q-Labs: ToolCertify User Guide, Version 4.0, 1999.
17. Software Research, Inc.: CAPBAK/Web [IE] Ver. 1.5 User Manual, San Francisco USA, 1999.