Volume 4, Issue 5, May 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Interactive Data flow Analysis using Web Application Testing on User Sessions

Shanthakumar.M[*]
PhD Scholar,
Department of Computer Science
Erode Arts & Science College
India

Janarthanam.S
Assistant Professor
Department of Computer Science
Gobi Arts & Science College
India

Dr.Sukumaran.S
Associate Professor
Department of Computer Science
Erode Arts & Science College
India

*Abstract—****The continuous use of the web for daily operations by businesses, consumers, and government has created a great demand for reliable web applications. One of the challenges of testing web applications derives from their dynamic content and structure. As we test a website, we may discover more about its structure and behaviour. This paper proposes a framework for collection of testability measures during the automated testing process (termed 'in-testing' measure collection). The measures gathered in this way can take account of dynamic and content driven aspects of web applications, such as form structure, client-side scripting and server-side code. We have completely automated the process from user session collection and reduction through replay. In order to control the maintenance cost and to enhance maintainability, quantitative metrics for predicting web applications maintainability must be used. Software applications in industry, novel testing approaches will be necessary to evaluate the quality of future (and more complex) web applications. In this paper, we investigate the testing challenges of future web applications and propose a testing methodology that addresses these challenges by the integration of search based testing, model-based testing, oracle learning, concurrency testing, combinatorial testing, regression testing, and coverage analysis. This paper also presents a testing meta model that states testing concepts and their relationships, which are used as the theoretical basis of the proposed testing methodology. it is not a systematic approach whose goal is precise control and scope. Rather, it is a technique that builds on the strengths of unscripted testing - speed, flexibility and range and by allowing it to be controlled, enables it to become a powerful part of an overall test strategy. Session-based testing mirrors the activities of experienced testers, but is not the subject that describes one situation in which session-based testing was successfully implemented.***

Keywords— Boundary Coverage Criterion, Session-based Testing, Software Metrics, Nonlinear Stream, Web Applications

I. INTRODUCTION

The product to be tested was an application delivered over the internet, and had been commercially live for just under a year. The application had a few hundred active users at a few dozen firms, and dealt with a large amount of incoming data submitted by many thousands of internet users, This application had been developed in-house by a medium size team (30-40 people total). The team continued to develop the application, and released a new version of the application every two weeks or so[1]. Work was driven by a semi-formal change request process. Web application developing has become a significant area, which has a great progress among other fields of software engineering. Web application has essential characteristics which have made it different from other ones. Web applications consist of some modules and interactive components in which various technologies could be used independently. It is important to note that in addition to the growth of web applications, the techniques used in web application developing have also great growth. So this fact makes web application testing more difficult.

The existing test process was immature, and the five-member team had little experience. None of the team had experience of a well-run test process. The precise nature of the problems cannot be detailed in this paper, but the process exhibited the following common characteristics.

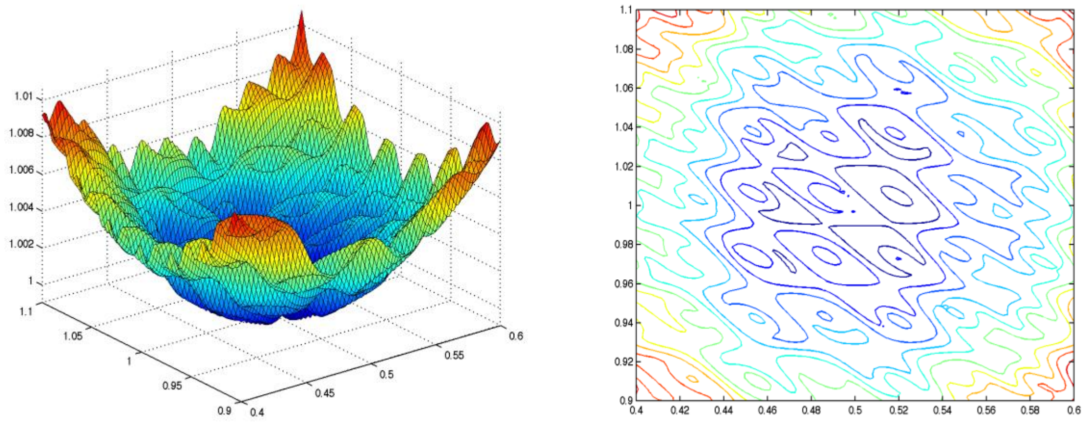
- Reactive - and therefore uncontrolled, and not necessarily focussed on important areas
- Could miss important bugs which had an immediate effect on customers
- Could not produce reliable information about the readiness of a release, and was not trusted.

Existing testing found good bugs, but in a bad way. Some of the most significant bugs would only be found by ad-hoc tests - particularly those caused by data problems and often characterised by intermittent symptoms - and the team were reluctant to move away from a proven approach. However, as the existing process was inefficient, time savings were not hard to find[12].

II. SESSION BASED TESTING

Session-based testing parallels the way that many experienced testers approach ad-hoc testing. While not a new technique, it has not been formalised - and there are no hard and fast rules to its execution. These limits are less well defined, but they are defined before the start of the test. By introducing these limits, session-based testing seeks to focus tester attention, allowing control, increasing the reliability of metrics and the repeatability of tests, and limiting the cost of poor exploration[4]. Furthermore, introducing concepts such as concerns, aspects in application design, and the structural properties may change their relationship with software quality factors. For example, software components relationships requires a more specific analysis, e.g., in aspect oriented design (AOD) is introduced an implicit coupling between the aspects and the modules in the principal decomposition, in that the latter may be unaware of the presence of aspects that intercept their execution and/or modify their structure On the other hand, some new metrics systems defined to study AOD software ([6],[3]) are also not adequate, because they study the aspect oriented programming, and they measure aspects properties such as advices, point-cuts, and so on.

A. *Problem of requirements :* This paper describes a metrics suite to help the user define quantitative system to measure existing Web software and to analyze its quality factors via structural properties. Nowadays the existing metrics systems for Web applications measure several structural properties, but often, they measure specific Web assets, such as navigation paths length, pages click-stream distances, and so on [10] for Web metrics roadmap. In this paper, we focus not only on Web specific measures, but, more generally, This imperfect specification keeps the tester way up to deliver time so that the whole context of the characteristic is known. When the entire context is understood, tests can be created that will confirm remaining scenarios. The secondary obstacle with textual description is that they are vague i.e. in the case that wrong integer is fed to the system, proper decision should be made [9].The first problem in the way of providing tests is to decide on the test target. Although, this issue may seem trivial, it is almost the primary stage that mistakes take place. A description of the system or software to be tested is essential. The shape of the description is widely different from a collection of call flow diagrams for a Voice Mail System, VO Mail, to the Graphical User Interface, GUI, of the E-Bank.



(a) (b)

Fig.1.Noise Quadratic (a) Data flow Surface Plots and (b) Level sets

A clarified set of characteristic and/or behaviours of a system are required to declare the boundary of the development and test work space. The classical ways of defining the valid system behaviour is with natural language prose in the style of *Requirement Specification* or *Functional Specification* [2,5]. For instance, the programmer might assume that the system should let the user to re-enter the digit, rather than the tester assumption that could be aborting the action.

III. TEST CASE

In the case of developing an algorithm that randomly (or deliberately) traverses along the machine through nodes and edges. [2, 7]. The order of edge names trough the created tracks are tests case e.g. In orders of edges from start node meet the minimum requirements of being a test case of the symbolized system. There is a diversity of constraints on what comprise a path to satisfy the requirements for test cases. Examples contain having the path start and end in the beginning state, limiting the number of loops or cycles in a path, and limiting the states that a path can meet.

A. *Test suite generation:* In such test suite generation, we generate a covering array in which all t-tuples, with the exception of constraints, are covered at least once. However, in our work on test suite prioritization, we noticed that the user-session-based test suites did not contain all possible t-tuples [6]. These fixes often involve small patches or revisions, but still, developers and testers need to perform regression testing on their products to detect whether these changes have introduced new faults. For instance, we examined three web applications and found that the associated user-session-based test suites did not contain an exhaustive collection of all t-tuples of parameter-values between windows.

Web applications change and are upgraded frequently due to security attacks, feature updates, or user preference changes.This makes sense because our web applications had many fields in which different users could manually enter personal data such as user ids, passwords, mailing addresses, and other textual data. This observation led us to design an algorithm that does not identify all parameter-values in a system nor enumerate the possible t-tuples, but rather only stores the t-tuples that appear in the test suite. We next construct a graph representing the Web application by using the information extracted from source code[11]. In our approach, Web application is recovered from source code and is modeled as a directed graph, the so-called Web navigation graph, where vertexes in the graph represent Web pages (either static HTML pages or dynamic generated pages), directed edges of this graph represent transitions among pages. These directed edges may be annotated with input parameters and path condition information associated with the transition.

B. Algorithm

```
(1) add the initial page identifier of the PFD into FIRST;
(2) if FIRST is empty, then go to (6);
(3) select the first page identifier denoted by pid from FIRST. If pid is within SECOND, then go to (5).
    Otherwise, add it into the end of SECOND;
(4) if pid is linking to other pages, then
    if some of the other page identifiers are within FIRST or SECOND, then generate their copies;
  • retain the links between pid and the other pages (or their copies) of the PFD,
  • add the other page identifiers (or their copies) into the end of FIRST;
(5) delete pid from FIRST and then go to (2);
(6) Output the derived PTT, which is the PFD with only the retained links.
```

We use five prioritization criteria. First, we use two-way and three-way inter-window parameter-value interaction coverage in order to evaluate whether three-way offers improvement over two-way in fault finding effectiveness. Second, we use two length-based criteria, namely, number of GET/POST requests in a test case and the number of parameter-values in a test case. We choose these length-based criteria as they performed well in our previous work [2,8]. Random ordering is used as a control.

Testing approaches for non-WAs have to be extended to handle these features before they are used in WA testing. This paper presents an agent based approach to perform data-flow testing of WAs. More precisely, the data-flow testing will be performed by autonomous test agents at the method level, object level, and object cluster level, from low abstraction level to high abstraction level. In the process of the recommended data-flow testing, an agent-based WA testing system (WAT) will automatically generate and coordinate test agents to decompose the task of testing an entire WA into a set of subtasks that can be accomplished by test agents. The test agents, rooted in the Belief–Desire–Intention (BDI) model, cooperate with each other to complete the testing of a WA. However, current automated testing techniques may produce false positives (or false negatives) even in a perfectly working system because the outcome of a test case depends on the state of the database which changes over time as data is inserted and deleted.

IV. TEST SUITE REDUCTION

An evolving program can require augmenting an existing test suite with new test cases that test the program's new functionality. The additional test cases can lead to redundant test cases, which waste valuable testing resources, in the testing process. The goal of test suite reduction for a given test requirement (e.g., statement or all-uses coverage)is to produce a test suite that is smaller than the original suite's size yet still satisfies the original suite's test requirement. Advantages of test-suite reduction techniques include reducing the cost of executing, validating, and managing test suites as the application evolves. Our study focuses on requirements-based reduction techniques and a concept analysis-based approach. Reduction techniques have also been developed based on bicriteria models, which derive a reduced test suite that maximizes coverage and improves the error-detection rate of the suite [3]. Harder et al. [10] applied an operational difference technique to generate and minimize the number of test cases. Their technique dynamically generates operational abstractions from test suite executions by adding test cases until the operational abstractions do not change.

Presented a dynamic navigation testing tool for Web applications. This tool can explore sequences of links in Web applications by exploring action sequences starting from a given URL and create data for form fields by choosing from a set of name-value pairs provided by the tester. Basically, the tool can automate the testing process of Web application. In this paper, we generate test path with path navigation diagram to model the behaviours of Web application. The noticeable difference lies in how the navigation graph is built and how to obtain the data for form fields.

However, the technique requires executing the test cases to determine their impact on the operational abstraction. We use the example test suite in Figure 1(a) to illustrate our study's reduction techniques. The rows show a subset of user sessions from a bookstore web application. The corresponding URLs of the application label the columns. Each table entry represents a user session's URL request.

This testing technique is relevant since it represents a first tentative step to extend to the Weld of Web applications the data flow testing approaches applicable to traditional software. However, to make it actually usable further investigations are required. Indeed, the effectiveness of the technique has not been validated by any experiment involving more than one example Web application: to carry out these experiments, an automated environment for testing execution including code analysers, data flow analysers, and code instrumentation tools would have been necessary. Moreover, indication about how this data flow testing approach may be integrated in a testing process would also be needed: as an example, the various testing perspectives and levels proposed by the approach might be considered in different phases of a testing process to carry out unit test, as well as integration or system test. However, also in this case an experimental validation and tuning would be required [8].

This approach was based on a test model named the *Navigational model* that focuses on HTML pages and navigational links of the application. Later, the same authors presented an additional lower layer model, the *Control flow model* representing the internal structure of Web pages in terms of the execution flow.

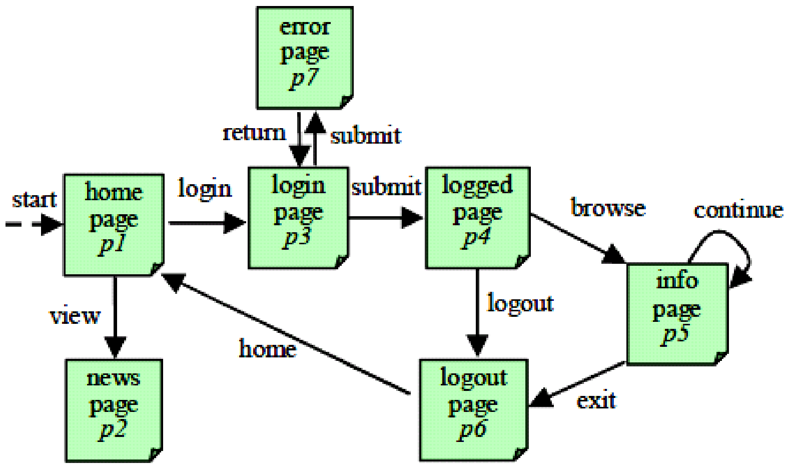


Fig.2.Page Via Control Data flow

Some conclusive considerations about the testing levels supported by white box techniques can be made. Some approaches will be applicable at the unit level, while other ones can be considered at the integration and system levels. For instance, the first approach proposed by Liu et al. [1, 9] is applicable at various testing levels, ranging from unit level to integration level. As an example, the intra-object perspective can be used to obtain various types of units to be tested, while inter-object and inter-application perspectives can be considered for establishing the items to be tested at the integration level. As a consequence, the choice of a testing technique to be applied in a testing process will also depend on the scope of the test to run.

V. CONCLUSIONS

The main conclusion is that all the testing aspects that are directly dependent on the implementation technologies have to be deeply adapted to the heterogeneous and 'dynamic' nature of the Web applications, while other aspects may be reused with a reduced adaptation effort. This finding also remarks that further research efforts should be spent to define and assess the effectiveness of testing models, methods, techniques and tools that combine traditional testing approaches with new and specific ones.

Moreover, as additional future trends, we expect that new relevant issues will arise in the Weld of Web services testing, as well as a new challenge will consist in the introduction of 'agile' methods in Web application testing processes for improving their effectiveness and efficiency.

REFERENCES

[1] H. Cheng, X. Yan, J. Han, and C. Hsu. Discriminative frequent pattern analysis for effective classification. In ICDE, pages 716-725, 2007.
[2] S. Elbaum, G.Rothermel, S.Karre, and M. Fisher II. Leveraging user-session data to support web application testing. IEEE Transactions on Software Engineering, 31(3):187–202, Mar. 2005.
[3] M. H. Alalfi, J.R.Cordy, and T. R. Dean. Modelling methods for web application verification and testing: state of the art. Software Testing, Verification, and Reliability, 19(4):265–296, 2009.
[4] W.Wang,Y.Lei, S.Sampath, R.Kacker, R. Kuhn, and J. Lawrence. A combinatorial approach to building navigation graphs for dynamic web applications. In International Conference on Software Maintenance, 2009.
[5] F.Ricca and P.Tonella. Analysis and testing of web applications. In Int'l Conf. on Software Engineering (ICSE), 2001.
[6] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A.Paradkar, and M. D. Ernst, "Finding bugs in web applications using dynamic test generation and explicit state model checking," IEEE Transactions on Software Engineering, vol. 36, no. 4, pp. 474–494, 2010.
[7] R. C. Bryce, S. Sampath, and A. M. Memon, "Developing a single model and test prioritization strategies for event-driven software," IEEE Transactions on Software Engineering, vol. 37, pp. 48–64, 2011.
[8] S. Artzi, J. Dolby, F. Tip, and M. Pistoia, "Practical fault localization for dynamic web applications," in Proceedings of the International Conference on Software Engineering, May 2010.
[9] S. Elbaum, A.G.Malishevsky, and G.Rothermel, "Test case prioritization: A family of empirical studies," IEEE Transactions on Software Engineering, vol. 28, no. 2, pp. 159–182, Feb. 2002.
[10] R.Santelices and M.J.Harrold, "Applying aggressive propagation-based strategies for testing changes," in International Conference on Software Testing, Verification and Validation, Apr. 2011.
[11] K. Dobolyi and W. Weimer, "Harnessing web-based application similarities to aid in regression testing," in Proceedings of the International Symposium on Software Reliability Engineering, Nov. 2009.
[12] K. Yuji, and K. Kenji. Amberate: a framework for automated vulnerability Scanners for web applications. In JSSST trans. On computer Software (2011), vol. 28, pp. 175–195.