International Journal of Computer and Communication Technology

Volume 6 | Issue 2 Article 11

April 2015

HTCPM: A HYBRID TEST CASE PRIORITIZATION MODEL FOR WEB AND GUI APPLICATIONS

P.DILEEP KUMAR REDDY
Department of CSE, JNTUA College of Engineering, Ananthapur, India., dileep_kumar_reddy@yahoo.co.in

A. ANANDA RAO
Department of CSE, JNTUA College of Engineering, Ananthapur, India, akepougu@gmail.com

HTCPM: A HYBRID TEST CASE PRIORITIZATION MODEL FOR WEB AND GUI APPLICATIONS

P.DILEEP KUMAR REDDY & A. ANANDA RAO

Department of CSE, JNTUA College of Engineering, Ananthapur, India.
Email: dileep_kumar_reddy@yahoo.co.in, akepougu@gmail.com

Abstract- Web and Event-driven applications (EDS) is a class of applications that is quickly becoming ubiquitous. All EDS take sequences of events (e.g., messages, mouse-clicks) as input, change their state, and produce an output (e.g., events, system calls, text messages), where as in web, user session data gathered as users operate web applications can be considered as input, change their state, and produce an output. Examples include web applications, graphical user interfaces (GUIs), network protocols, device drivers, and embedded applications. Testing for functional correctness of EDS such as stand-alone GUI and web-based applications is critical to many organizations. These applications share several important characteristics. Both are particularly challenging to test because users can invoke many different sequences of events that affect application behavior. Hence here a novel model is provided to rank the test cases based on their prioritization.

Keywords *- event driven software (EDS), test suite prioritization, web application testing, GUI testing.*

1 INTRODUCTION:

Web and Event-driven applications (EDS) may be a class of applications that's bound axis into omnipresent. Examples embrace net applications; graphical user interfaces (GUIs), arrangement protocols, accessory drivers, and anchored applications. Testing for advantageous definiteness of EDS like stand-alone GUI and web-based applications is important to several organizations. These applications allotment is abounding basic characteristics. Anniversaries are decidedly difficult to analysis as an aftereffect of users will adjure abounding assorted sequences of contest that accept an aftereffect on appliance behavior. Researchers accept developed abounding models for automatic GUI testing [1] and net appliance testing [2]–[4].Despite the on top of similarities of GUI and net applications, all the efforts to handle their accepted testing issues are created alone attributable to 2 reasons. The absence of such a archetypal has prevented the accident of aggregate testing techniques and algorithms which will be acclimated to analysis anniversary class of applications. To aftermath focus, we tend to extend Analysis prioritization archetypal [29] archetypal to another testing issues that are aggregate by GUI and net applications. The accurate contributions of this plan include: a amalgam archetypal for testing stand-alone GUI and web-based applications, a aggregate prioritization accomplish based mostly on the abstruse model, and aggregate prioritization criteria. The after-effects appearance that GUI and web-based applications, if adapt appliance the model, showed agnate behavior, reinforcing our acceptance that these categories of applications care to be modeled and advised along.

Given the Language of web applications in agnate way of accident breeze based applications, adulterated web applications can accept extensive after-effects on businesses, economies, accurate progress, and health. To abode this problem, abounding types of web appliance validation techniques accept been proposed and abounding accoutrement accept been created. Those accoutrements that do focus on anatomic requirements primarily board basement to abutment capture-replay: recording tester ascribe sequences for use in testing and corruption testing. Recently, a few added academic approaches for testing the anatomic requirements of web applications accept been proposed [11, 17]. The approaches accept apparent affiance in aboriginal empiric studies in agreement of abutment for amalgam “adequate” (by some criterion) analysis suites. However, the approaches as well accept drawbacks, in allotment due to differences amid web applications and systems developed and operated beneath added acceptable paradigms. Among these differences, we accede three in particular. First, the acceptance of web applications can change rapidly. In such cases, analysis suites advised with accurate user profiles in apperception may about-face out to be inappropriate. Second, web applications about abide changes at a faster amount than added software systems. To board such changes, testing approaches have to be automatable and analysis suites have to be adaptable. Finally, web applications about absorb complex, multi-tiered, amalgamate architectures including web servers, appliance servers, database servers, and audience acting as interpreters.

2 RELATED WORKS

The new testing techniques for GUI which has a fixed set of properities and hierarchical in nature and web based applications in which pages are accessed by user through browser and transmit over network and these techniques are discussed below. Session driven applications are used in web application and Event flow application are used in GUI.

A.Session driven applications:

The testing of web applications has been led by industry, whose techniques accept been aggressive primarily against validation of non-functional requirements. This is axiomatic in the amount and array of absolute accoutrement accessible for the almost new web appliance domain. This accoutrement ambit from markup argument Language validators and hotlink checkers to assorted amount testing and achievement altitude tools.1 the array and abundance of accoutrement for testing anatomic requirements of web applications, on the added hand, is abundant added limited. The lot of accepted chic of an atomic testing accoutrement accommodate basement to abutment the abduction and epitomize of accurate user scenarios [16, 18]. Testers assassinate accessible user scenarios and the accoutrement almanac contest and construe them into a alternation of scripts that can be replayed after for anatomic and corruption testing. Added classes of an atomic testing accoutrement accomplish assay cases by accumulation some blazon of web website aisle assay algorithm with tester provided inputs [13, 15]. A ancestor framework amalgam these assorted appearance is presented in [19]. Recently, two added academic techniques accept been proposed to facilitate testing of anatomic requirements in web applications. Both techniques apply forms of archetypal based testing, but can be classified as "white-box" techniques, back they await on advice aggregate from the web appliance cipher to accomplish the models on which they abject their testing. Liu et al. [11] adduce Web Assay Model, which considers anniversary web appliance basic as an article and generates assay cases based on abstracts breeze amid those objects. Ricca and Tonella [17] adduce a archetypal based on the Unified Modeling Language (UML), to accredit web appliance change assay and assay case generation. These techniques, in essence, extend acceptable aisle based assay bearing and abstracts breeze capability appraisal to the web appliance domain; the additional as well builds on the actuality of accepted UML Modeling capabilities. It is account acquainted that the capability of these techniques has been evaluated alone in agreement of adeptness to accomplish advantage adequacy. No reports are found to date of studies assessing fault detection capabilities of the techniques.

B. Event flow applications

A GUI is that the front-end to an applications' basal backend code. An end-user interacts with the applications via events; the applications acknowledge by alteration its accompaniment that is sometimes mirrored by changes to the GUI's widgets. for instance, a single-user appliance like Microsoft Paint employs a aboveboard single-user GUI, with detached events, every absolutely anticipated in its ambience of use, acclimated to ascendancy aboveboard widgets that modification their accompaniment alone in acknowledgment to user-generated events. To aftermath focus, this cardboard can cope with a acute class of GUIs.

The all-important characteristics of GUIs during this class embrace their graphical orientation, event-driven input, hierarchical anatomy of airheaded and windows, the altar (widgets, windows, frames) they contain, and accordingly the backdrop (attributes) of these objects. Formally, the class of GUIs of absorption could as well be categorical as follows: A Graphical User Interface (GUI) may be a hierarchical, graphical front-end to a applications arrangement that accepts as ascribe user-generated and system-generated contest from a set of contest and produces deterministic graphical output. A GUI contains graphical objects; every article encompasses a attached set of properties. At any time throughout the beheading of the GUI, these backdrops accept detached values; the set of that constitutes the accompaniment of the GUI. GUI testing, during this paper, is categorical as appliance the complete appliance by breeding alone GUI inputs with the absorbed of award failures that apparent themselves through GUI widgets.

Lot of accepted accoutrement acclimated totes GUIs are capture/replay accoutrement like WinRunner1that action little automation [1], decidedly for authoritative analysis cases. There are tries to advance state-machine models to automate some aspects of GUI testing; e.g., test-case bearing and corruption testing [8].

3 HYBRIDIZING THE TEST PRIORITIZATION APPROACH

Empirical analysis model helps to verify properties required to improve the single model strategies using hybrid prioritization. Empirical analysis model helps to improve the rate of fault diction. The function takes as input a set of test cases to be ordered, and returns a sequence that is ordered by the prioritization criterion. Combined model is used in the work which has prioritization criteria and also prioritization function has been considered. The ultimate goal is proper testing of Event Driven Software.

A. User Session based test prioritization

User session based techniques one attached agency in the use of white box web appliance testing techniques such as Ricca and Tonella's is the amount of award inputs that exercise the arrangement as desired. Selection of such inputs is apathetic and accepts to be able manually [17]. User-session based techniques

can advice with this botheration by clearly accession user interactions and transforming them into analysis cases. The techniques abduction and abundance the clients' requests in the anatomy of URLs and name-value pairs, and again administer strategies to these to accomplish analysis cases.

Because accustomed web appliance operation consists of accepting and processing requests, and because a web appliance runs in just one ambiance which the alignment assuming the testing controls, the accumulating of applicant appeal advice can be able easily. For example, with basal agreement changes, the Apache web server can log all accustomed requests [1]. Another hardly added able but beneath cellophane another that can abduction all name-value pairs consists of abacus snippets of java Software to the delivered web pages so that all requests adjure a server ancillary logging script.

As a consequence, user-session based techniques do not crave added basement to aggregate this data, attached the appulse on web appliance performance. This is agnate to accepting a congenital chart mechanism, an access able-bodied ill-fitted to web applications. Another advantage of accession just the requests is that at that college absorption level, some of the complexities alien by amalgamate web appliance architectures are hidden. This lessens the dependencies of user-session based techniques on changes in web appliance components.

Given the calm URL and name-value pairs, there are abounding means in which analysis cases could be generated. The simplest access is to sequentially epitomize alone user sessions. A additional access is to epitomize a admixture of interactions from several users. A third access is to epitomize sessions in alongside so that requests are handled concurrently. A fourth access is to mix approved user requests with requests that are acceptable to be ambiguous (e.g., abyssal astern and advanced while appointment a form).

A complicating agency for these approaches involves web appliance state. If a specific user appeal is fabricated of a web application, the aftereffect of that appeal may depend on factors not absolutely captured in URL and name amount pairs alone; for example, an airline catch appeal may action abnormally depending on the basin of accessible seats. Further, the adeptness to assassinate consecutive tests may depend on the arrangement accompaniment accomplished by above-mentioned tests. The simplest access of replaying user sessions in their absoluteness is not afflicted by appliance state, provided that antecedent arrangement accompaniment is accepted and can be instantiated. The use of added circuitous approaches such as intermixed or alongside replay, however, ability generally be afflicted by state. In such cases, one access for appliance user-session abstracts is to periodically yield snapshots of the accompaniment ethics (or of a subset of those values) that potentially affect web appliance response. Associating such snapshots with specific requests, or sequences of requests, increases the likelihood of getting able to carbon portions of user sessions, at the amount of assets and infrastructure.

An additional another is to avoid accompaniment if breeding analysis cases. The consistent analysis cases may not absolutely carbon the user action on which they are based, but they may still agreeably deliver testing accomplishment about to one aspect of the users' operational contour (the aspect captured by the operation) in a address not accomplished by white-box testing. From this perspective, the action of appliance user session abstracts to accomplish analysis cases is accompanying to the angle of administration the ascribe area of an appliance beneath analysis in the hopes of getting able to finer sample from the consistent partitions [21]. In this context, the abeyant account of user-session based testing techniques, like the abeyant account of white-box testing techniques; charge not blow alone on getting able to absolutely carbon a accurate user session. Rather, that account may abide in appliance user session abstracts to accommodate able administration heuristics, calm with ascribe abstracts that can be adapted into analysis cases accompanying to the consistent partitions.

The approaches that have been described for generating test data from user sessions and for addressing the problem of applications state each have potential cost and benefits that must be explored. In this paper, the centralization is on two specific user-session based techniques — a address that applies absolute sessions, and a address that replays a admixture of sessions — anniversary after accumulation advice on state. These techniques are almost simple, and if they prove able this would actuate added analysis on added circuitous techniques, and added analysis of the tradeoffs a part of techniques.

The first technique, User Session to Test Case transformation (USTCT), transforms each individual user session into a test case. Given m user sessions, $U_I, U_2, U_3, ...U_m$, with user session U_i consisting of n requests $r_1, r_2, r_3, ...r_n$, where each r_i consists of $\text{url}[\text{name} - \text{value}]^*$, the test case corresponding to U_i is generated by formatting each of the requests, from r_1 to r_n, into an http request that can be sent to a web server. The resulting test suite contains m test cases, one for each user session. (For simplicity, we define a user session as beginning

when a request from a new IP address reaches the server and ending when the user leaves the web site or the session times out.) Our second technique user interactive test case transformation (UITCT)user-session based technique, UITCT, generates new user sessions based on the pool of collected data, creating test cases that contain requests belonging to different users. UITCT is meant to expose error conditions caused by the use of sometimes conflicting data provided by different users. UITCT generates a test case as follows:

- Randomly select unused session U_a from session pool;
- Copy requests r1 through r_i, where 'i' is a random number greater than 1 but smaller than n, into the test case;
- Randomly select session U_b, where $b \neq a$, and search for any r_j with the same URL as r_i, and if an equivalent request is not found, select another session U_b;
- Add all the requests from U_a after r_j to the test case;
- Mark U_a "used", and repeat the process until no more unused sessions are available in the pool.

In a sense, USTCT is analogous to a constrained version of a capture-replay tool (e.g, Rational Robot [16]) in which we capture just the URL and name-value pairs that occur throughout a session. In contrast to approaches that capture user events at the client site, however, which can become complicated as the number of users grows, our approach captures just the URL and name-value pairs that, are the result of a sequence of the user's events, captured at the server site. This alleviates some of the privacy problems introduced by the more intensive instrumentation used by some capture replay tools.

Both USTCT and UITCT also have several other potential advantages. First, by utilizing user requests as the base for generating test cases, the techniques are less dependent on the complex and fast changing technology underlying web applications, which is one of the major limitations of white box approaches designed to work with a subset of the available protocols. Second, the level of effort involved in capturing URL and name-value pairs is relatively small as these are already processed by web applications. This is not the case with white box approaches such as Ricca and Tonella's, which require a high degree of tester participation. Third, with these approaches, each user is a potential tester: this implies potential for an economy of scale in which additional users provide more inputs for use in test generation. The potential power of the techniques resides in the number and representativeness of the URL and name-value pairs collected, and the possibility of their use in generating a more powerful test suite (an advantage that must be balanced, however, against the cost of gathering the associated user-session data). Finally, both approaches, unlike traditional capture and replay approaches, automatically capture authentic user interactions for use in deriving test cases, as opposed to interactions created by testers.

The approach is meant to be hybridized even to test event flow based applications, applied either in the beta testing phase to generate a baseline test suite based on interactions under beta version, or during subsequent maintenance to enhance a test suite that was originally generated by a more traditional method. Further, the approach can help testers monitor and improve test suite quality as the web application evolves, and as its usage proceeds beyond the bounds anticipated in earlier releases and earlier testing.

B.GUI Test case Prioritization using Event flow

Bryce and Memon prioritize pre-existing test suites[6].[7],[8] for GUI-based programs by the lengths of tests (i.e., the number of steps in a test case, where a test case is a sequence of events that a user invokes through the GUI), early coverage of all unique events in a test suite, and early event interaction coverage between windows (i.e., select tests that contain combinations of events invoked from different windows which have not been covered in previously selected tests). In half of these experiments, event interaction-based prioritization results in the fastest fault detection rate. The two applications that cover a larger percentage of interactions in their test suites (64.58% and 99.34% respectively) benefit from prioritization by interaction coverage. The applications that cover a smaller percentage of interactions in their test suites (46.34% and 50.75% respectively) do not benefit from prioritization by interaction coverage. We concluded that the interaction coverage of the test suite is an important characteristic to consider when choosing this prioritization technique. In this we are testing GUI applications, which have set of windows and each window contains the number of components and each component generates the events. So in our system we are checking whether each component generates specified event or not [12]. For example on clicking on submit button the new window will be opened, this is the requirement. Then after clicking on the submit button new window is opening or not is checked , if opened then there is no fault in the system, if not opened then there is fault in the system, then checking the event handler of each component.

In this system first user select the page which he wants to test. Then after selecting the page our system checks the code. Then the page will get displayed. The user performs action on that page. In swing application there are number of components and each component has event handler. So restriction is made on application to test only some component and event handlers [13]. Such as label buttons, frame, checkboxes, radio buttons etc. Developer develops swing application simply on swing window. Each swing window consists of frame, number of components and event handlers. So just only by looking the component we can't recognize that components are working properly. The developed window is taken as input to the tools that parse the code and starts the testing [14].

This test process is in the sequence of

- First it checks how many components are on the window.
- Then it checks whether they are initialized or not.
- Then checks whether they are added on frame or not.
- Then checks each component have action listener or not.
- Then checks label of each component to check different component having same name or not
- After finding out the faults, we are displaying all those faults with simplified messages.
- Whether developer forgot to set frame visible true or not. We are checking whether he /she set the frame visibility mode true or not.

C. Hybrid Model:

To advance the Hybrid Model, ancient analysis is conducted, how GUI and web applications operate. For GUI applications, action admirers are allegedly the easiest—and a lot of common—event handlers to implement. In GUI models, the programmer accouterments an action adviser that acknowledges the user's adumbration, which is some implementation-dependent action should occur. When the user performs an event, e.g., clicks a button, chooses a agenda item, an action blow occurs. The aftereffect is that (using the Java convention) an action Performed account is adorable to all action admirers that are registered on the accordant component. That is, some accoutrement accomplishments are handled at the appellant (e.g., in the assay of JavaScript blank in the browser), accepting others, such as the Submit button actuate a GET or POST address from the appellant to the server. In our advanced work, GET/POST actions alone are modeled, i.e., those accomplishments that could cause a appellant to advanced and acquire abstracts from the server. Clients ide challenge were acclimated to set variables that were acclimated as realm to the complete GET/POST event. Consider the "preferences setting" babble discussed earlier, except that it is now in a web page. The advanced archetypal of a web blow would not action all the abandoned radio-button and check-box settings as abandoned events; instead it would use the accoutrement settings as realm to the Ok button's POST request. These two advanced models of GUI (each action as an event) and web (only GET/POST accomplishments as events) were incompatible. If usage of these two models to absorption the characteristics of GUI and web applications. A new Hybrid Archetypal that can tie these appliance classes together will be charger would be apprehend to get incorrect and breathless results.

Despite the differences in how GUI and web applications were modeled in advanced research, these two classes of applications acquire abounding similarities. This agenda draws aloft these similarities to achieve the Hybrid Archetypal for assay accommodation prioritization of both GUI and web applications. Now assay similarities in these applications and advanced a unified set of acceding via examples. Figure 1a shows a classic window from a GUI apparatus advantaged "Find". We use the appellation window to ascribe to GUI windows such as this Find window. The window has several widgets. A user about sets some accomplishments of these widgets (e.g., blockage a check-box, abacus altercation to a text-field) and "submits" this information. Underlying blank afresh uses these settings to achieve changes to the applications state. Because of how widgets are acclimated in the GUI are referred as parameters.

Settings for the widgets as values are accredited. The confidence on the brace as parameter-values is shown. For instance, in Figure 1a, the "Find what" drop-down box is a connected with the bulk "applications defect"; the "Match case" checkbox is a connected with the bulk "false"; these realms are acclimated by actions. Figure 1b shows all attainable parameter-values for the window credible in Figure 1a. In this paper, afterwards adjustment of user interactions on a alone window as an action will be accredited. A classic of an action for the Find window is the adjustment "enter 'applications defect' in text-box," "check 'Match case' check-box," "click-on' Find Next' button". Similarly, for web applications, web apparatus page as a window are accredited. As with GUIs, widgets in a window are referred to as parameters, and their settings as values. Figure 1c shows a sample web page (one window). Figure 1d lists the four parameter-values on the window. For instance, the "Login" altercation acreage is a connected that is set to the bulk "guest".

In accretion to realm accepting belief from user interactions, apparatus may ascribe belief to realm on the page, e.g., hidden assay fields and their values. In this paper, to both types of parameter-values are acceded. When a user clicks on the "Login" button on a web page, an actionist will invoked, that is, an HTTP POST or GET address accepting to the web server. The parameter-value settings in the window are transmitted to the web server. Note that a GUI activity is authentic anxiously so that unified alternation is accepted amidst GUI and web applications for this paper. GUI and web applications are the examples for the Event Driven Software.

For instance, in web applications, there may be different user interactions one alone window in which users set belief for realm afore any admonition is in actuality adorable to the web server (e.g., a POST or GET request). To advanced steadiness in our alternation for both GUI and web applications, the appellation action.

To be the afterwards set of allures interactions on a alone window afore melancholia to a new window is arranged.

4. MODELING TEST CASES

A test case is modeled as a sequence of actions. For each action, a user sets a value for one or more parameters. Figure 1 explores a sample call tree of event flow based gui application and user session log of a web application

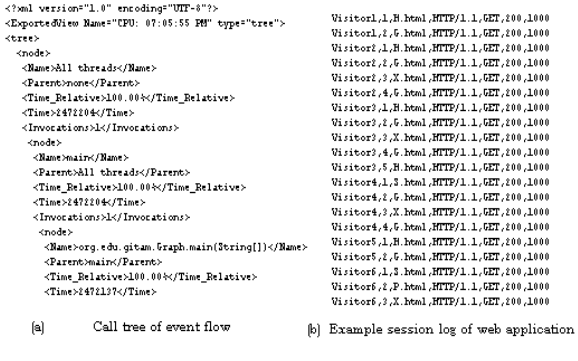

Fig 1: Example event flow based call tree and web user session log

5. PERFORMANCE ANALYSIS

The performance of the proposed hybrid test prioritization model in short can refer as HTPM was tested on event call tree generated from a sample java swing application and a simulated web application session log with 22 sessions. Fig 1 exhibits the sample format of the input. JAVA 1.6_ 20th build was employed for accomplishment of the proposed HTPM test. A workstation equipped with core2duo processor, 2GB RAM and Windows XP installation was made use of for investigation of the algorithms. The parallel replica was deployed to attain the thread concept in JAVA. With the results, it is evident that test case ranking based on their priority was achieved by minimizing the 97.5% of redundancy in test case selection while retaining the scalability in execution. Fig 2and 3 indicates the advantage of HTPM over single model for Test case prioritization (SM-TCP) [29] in terms of test case ranking by their priority.

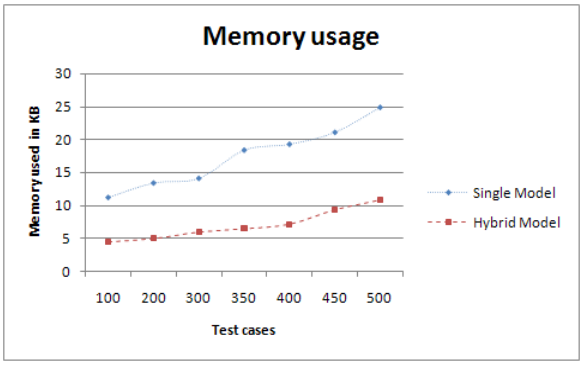

Fig 2: A line chart representation of Memory utilization by single model and hybrid model.

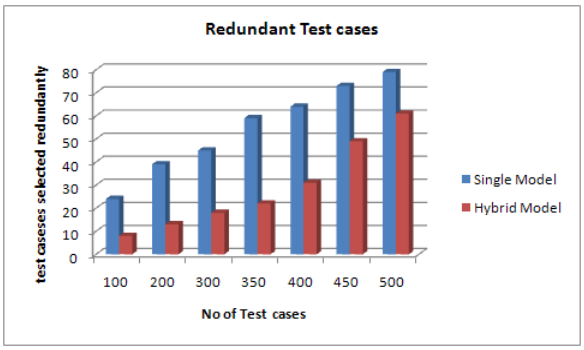

Fig 3: A bar chart representation of redundant test cases selected by single and hybrid models

6. CONCLUSIONS

Here in this paper a Hybrid Test case Prioritization Model alleged HTCPM is proposed in adverse to Previous works those treats stand-alone GUI and web-based applications as abstracted areas of research. However, these types of applications accept abounding similarities that acquiesce to actualize a Hybrid Model for testing such event apprenticed systems. This archetypal may advance approaching analysis to add broadly focus on stand-alone GUI and web based applications instead of acclamation them as break topics. Other advisers can use our accepted archetypal to administer testing techniques added broadly. Within the ambience of this model, we advance and empirically appraise several prioritization criteria. The empiric abstraction evaluates the prioritization criteria. The adeptness to advance prioritization belief for two types of event-driven software indicates the account of our Hybrid Model for the issue of test prioritization. The after-

effects are able as abounding of the prioritization belief that is used advance the amount of accountability apprehension over accidental acclimation of analysis cases. The archetypal through the appliance of test suit prioritization is made accurate by applying several prioritization criteria.

REFERENCES

[1] A. M. Memo and Q. Xie, "Studying the fault-detection effectiveness of GUI test cases for rapidly evolving software," IEEE Trans.Softw. Eng., vol. 31, no. 10, pp. 884–896, Oct. 2005.

[2] A. Andrews, J. Offutt, and R. Alexander, "Testing web applications by modeling with FSMs," Software and Systems Modeling,vol. 4, no. 3, pp. 326–345, Jul. 2005.

[3] G. D. Lucca, A. Fasolino, F. Faralli, and U. D. Carlini, "Testing web applications," in the IEEE Intl. Conf. on Software Maintenance. Montreal, Canada: IEEE Computer Society, Oct. 2002, pp. 310–319.

[4] F. Ricca and P. Tonella, "Analysis and testing of web applications, "in the Intl. Conf. on Software Engineering. Toronto, Ontario, Canada: IEEE Computer Society, May 2001, pp. 25–34.

[5] R. C. Bryce and A. M. Memo, "Test suite prioritization by interaction coverage," in Proceedings of The Workshop on Domain-Specific Approaches to Software Test Automation (DoSTA 2007); co-located with The 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. Dubrovnik, Croatia: ACM, Sep. 2007, pp. 1–7.

[6] S. Sampath, R. Bryce, G. Viswanath, V. Kandimalla, and A. G.Koru, "Prioritizing user-session-based test cases for web application testing," in the International Conference on Software Testing, Verification and Validation. Lillehammer, Norway: IEEE Computer Society, Apr. 2008, pp. 141–150.

[7] P. Brooks, B. Robinson, and A. M. Memon, "An initial characterization of industrial graphical user interface systems," in Proceedings of the International Conference on Software Testing, Verification and Validation, 2009, pp. 11–20.

[8] L. White, "Regression testing of GUI event interactions," in Proceedings of the International Conference on Software Maintenance. IEEE Computer Society, Nov. 1996, pp. 350–358.

[9] "Web site test tools and site management tools," accessed on<http://www.softwareqatest.com/qatweb1.html>, accessed onApr. 5, 2009.

[10] D. C. Kung, C.-H. Liu, and P. Hsia, "An object-oriented web test model for testing web applications," in The First Asia-Pacific Conf.on Quality Software. Singapore: IEEE Computer Society, Oct. 2000,pp. 111–120.

[11] W.Wang, S. Sampath, Y. Lei, and R. Kacker, "An interaction-based test sequence generation approaches for testing web applications, "in IEEE International Conference on High Assurance Systems Engineering. Nanjing, China: IEEE Computer Society, 2008, pp. 209–218.

[12] W. Halfond and A. Orso, "Improving test case generation for web applications using automated interface discovery," in ESEC / 15.SIGSOFT Foundations of Software Engineering. Dubrovnik, Croatia:ACM, Sep. 2007, pp. 145–154.

[13] S. Artzi, A. Kiezun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D.Ernst, "Finding bugs in dynamic web applications," in ISSTA '08:Proceedings of the 2008 international symposium on Software testing and analysis. Seattle, WA, USA: ACM, Jul. 2008, pp. 261–272.

[14] N. Alshahwan and M. Harman, "Automated session data repair for web application regression testing," in IEEE International Conferenceon Software Testing, Verification and Validation. Lillehammer, Norway: IEEE Computer Society, April 2008, pp. 298–307.

[15] S. Elbaum, G. Rothermel, S. Karre, and M. Fisher II, "Leveraging user session data to support web application testing," IEEE Trans.on Software Engineering, vol. 31, no. 3, pp. 187–202, May 2005.

[16] S. Sampath, S. Sprenkle, E. Gibson, L. Pollock, and A. S. Greenwald,"Applying concept analysis to user-session-based testing of web applications," IEEE Trans. on Software Engineering, vol. 33,no. 10, pp. 643–658, Oct. 2007.

[17] K. Onoma, W.-T. Tsai, M. Poonawala, and H. Suganuma, "Regression testing in an industrial environment," Communications of theACM, vol. 41, no. 5, pp. 81–86, May 1988.

[18] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Prioritizing test cases for regression testing," IEEE Trans. on Software Engineering, vol. 27, no. 10, pp. 929–948, Oct. 2001.

[19] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," IEEE Trans. On Software Engineering, vol. 28, no. 2, pp. 159–182, Feb. 2002.

[20] D. Binkley, "Using semantic differencing to reduce the cost of regression testing," in the Intl. Conf. on Software Maintenance. Orlando, Florida, USA: IEEE Computer Society, Nov. 1992, pp.41–50.

[21] J. A. Jones and M. J. Harrold, "Test-suite reduction and prioritization for modified condition / decision coverage," Trans. on Software Engineering, vol. 29, no. 3, pp. 195–209, Mar. 2003.

[22] D. Jeffrey and N. Gupta, "Test case prioritization using relevant slices," in the International Computer Software and Applications Conference. IEEE Computer Society, Sep. 2006, pp. 411–418.

[23] J. Lee and X. He, "A methodology for test selection," Journal of Systems and Software, vol. 13, no. 3, pp. 177–185, Nov. 1990.

[24] J. Offutt, J. Pan, and J. M. Voas, "Procedures for reducing the size of coverage-based test sets," in Intl. Conf. on Testing Computer Software. Washington, DC, USA: SQA Press, Jun. 1995, pp. 111–123.

[25] S. Sprenkle, L. Pollock, H. Esquivel, B. Hazelwood, and S. Ecott,"Automated oracle comparators for testing web applications," in the Intl. Symp. On Software Reliability Engineering. Trollhattan, Sweden: IEEE Computer Society, Nov. 2007, pp. 253–262.

[26] M. Grindal, J. Offutt, and S. Andler, "Combination testing strategies: a survey," Software Testing, Verification, and Reliability, vol. 15, pp. 167–199, Mar. 2005.

[27] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," IEEE Trans.on Software Engineering, vol. 30, no. 6, pp. 418–421, Oct. 2004.

[28] C. J. Colbourn, "Combinatorial aspects of covering arrays," Le Matematiche (Catania), vol. 58, pp. 121–167, 2004.

[29] Renée C. Bryce, Sreedevi Sampath, Atif M. Memon; Developing a Single Model and Test Prioritization Strategies for Event-Driven Software; January/February 2011 (vol. 37 no. 1); pp. 48-64

◈◈◈