Using User Sessions for Test Case Generation and Execution

Izzat Alsmadi, and Kenneth Magel

***Abstract*—the goal of software testing is to detect bugs using the sources available to the project. This paper presents utilizing user sessions for test case generation and execution. User sessions can be gathered from the application in business environments and represent user stories or scenarios. Rather than rerunning user sessions for test automation, as in capture/reply tools, this research focuses on abstracting requirements from those sessions to make it independent of the scripting language or the tool that created them. This approach is expected to improve the utilization of user sessions from being copied and reused in the same original format, which makes it complex to edit and inflexible, to a format that can be used and utilized in different applications and platforms. The suggested approach abstracts user sessions to make them more independent and reusable.**

***Index terms*—Software testing, Graphical User Interface (GUI), user sessions, and test automation**.

I. INTRODUCTION

The potential usefulness of the user-session based testing technique is on being able to exactly reproduce and execute a particular user session. User session data can also provide effective partitioning or coverage, together with using these sessions as input data that can be transformed into test cases [1]. The test cases generated from the user sessions do not replace those developed by testers. For better coverage, both alternatives should be considered.

The advantage of using session data in testing is that since it represents authentic user behavior it would be more realistic and more likely to expose actual scenario bugs [2]. Another advantage is the utilization of users' sessions for testing and hence, using the application users as testers. User session data can help produce effective test suites with very little expense [3].

Capture/replay tools such as IBM Rational Robot or WinRunner capture user sessions in a format that can be later replayed automatically during regression testing [4].

Manuscript received January 18, 2008. Izzat M Alsmadi is a PhD student in software engineering at NDSU, department of computer science, 258 IACC North Dakota State University, Fargo, North Dakota 58105, phone: 701-293-1841, email: Izzat.alsmadi@ndsu.edu.

Kenneth Magel is a professor in the department of computer science, NDSU. He is currently the department associate chair. IACC 258A21, department of computer science, 258 IACC North Dakota State University, Fargo, North Dakota 58105, phone: 701-231-8189, email: kenneth.magel@ndsu.edu.

We can utilize some of the features in capture replay tools to do more than just replaying an identical copy of user sessions. In this research, the suggested use of user session's goes beyond replaying the same saved copy (recorded manually through a user), user sessions guide, by abstraction, the test generation algorithms in order to get more realistic test cases. One problem with the capture/replay tools is in dealing with the complex generated scripts. The fact that the script is rigid and any GUI change requires editing the script or generating a new one can be relieved by abstracting the script. Information like the sequence of the controls in the script and the events is that matter and needed to be extracted. This allows the script to be used in different applications and not only in its specific scenario and scripting language. An application is developed to extract specific information from the recorded script. The execution is done using some API's to simulate user actions which replace using the script for execution in the capture/replay tools.

II. RELATED WORK

In order to use user actions or sessions, they have to be formally described or modeled. Several models are suggested to model the users' tasks' descriptions. Table I shows the resources required for every user action in a user session model. Norman's execution-evaluation model presented a similar model for human information processing [6]. Users actions can be described in three levels; goals, tasks and actions. Actions maybe directly related to the specific function, however, users may take other actions which are not motivated by tasks during interaction. User actions are often based on a predefined list. Actions that are not listed maybe considered irrelevant to that specific test. An automated extracting process may not be able to distinguish actions related to tasks from actions that are not.

Capture/replay tools are widely used in testing for test automation. Users are required to perform the initial tests; a recording tool records the script and replays it whenever it is required. In principle, this is utilizing testers' sessions for testing. The re-played session is an exact copy of the one manually executed; there is no data extracting or information processing involved. Usually any change in the user interface requires those scripts to be edited and modified.

Table I. The parameters of user session resources' model [7].

Resource	Description
Goal	The final state that the user wants to achieve
Plan	A sequence of actions that the user intends to take, in order to achieve their goal.
State	The condition or overall properties of the whole system at any given moment
Possibility	The range of possible actions which could be taken by the user
Action effect	The consequence (i.e. post condition) as a result of taking a certain action
History	The knowledge of previous actions and their post conditions

Several research projects are presented regarding the usage of user sessions in web application for validation issues [1, 2, 3, 8, and 9]. Usually it is easier to gather users' sessions from websites or applications than from regular applications. A typical test case in web applications includes one or more web page to be surfed in a certain sequence. User-session based techniques can help with this problem by transparently collecting user interactions (clients' requests) in the form of URLs and name-value pairs, and then applying strategies to these to generate test cases [8]. It is also easy to extract information from web sessions such as the link visited, the time, etc.

III. GOALS AND APPROACHES

There are two goals of using user sessions in GUI test automation. First, user sessions are used a method for test case prioritization. Information gathered from user sessions is used to specify the weight of user scenarios. Second, Abstract the user sessions output and use it as an input for generating test cases.

A. Weight controls from user sessions

We can analyze several user captured sessions (e.g. from testers or users in beta testing) to automatically weight the GUI controls or widgets [10]. User session data is the set of user actions performed on the Application Under Test (AUT) from entering the application until leaving it.

We can classify a control, or a pair of controls, according to the number of times they are repeated in a user session. User sessions are likely to detect faults in the application that are not predictable in earlier testing phases. Another advantage of testing with user sessions is that testing is possible in the absence of specifications or in the presence of incorrect and incomplete specifications, which often occurs in software development [5].

The session logs all the controls that are executed in the different scenarios. A simple count or percentage is given to each control depending on how many times it is listed in those scenarios. The test scenarios should include all primary and major use cases for the AUT. The controls' weights (calculated from user sessions) can drive the test case generation and execution. Theoretically all controls should get the same weight in the generated test suite. However, in real scenarios this may not be true. We can use the weighing method for single controls or for a sequence of controls (result from a specific use case).

We may cluster the controls, or sequence of controls, according to their usage from user sessions into three levels; heavily used, medium and low. Depending on the availability of the resources to testing, we may choose one or two categories and generate test cases that cover those controls in the categories with a proportion to their weight or occurrence.

The developed algorithm in this research is considered as a hybrid technique that uses some of the capture/ reply processes. In a capture/ reply tool, the same user session that is captured in the manual testing is executed. In this approach the controls' weights are extracted from the manual testing to guide test case generation and execution. The reason for considering this track rather than using capture/ reply test execution and validation is to avoid the dependency on the absolute location of the screen and controls that is required by capture/replay tools. Having a hybrid solution may give us the best of both and utilize the accumulative experience and knowledge in different technologies.

In order to record user events, we implemented in our C# application the interface IMessageFilter that is used to capture messages between Window applications and components. In the AUT, each GUI control that is triggered by the user is logged to a file that represents the user sessions. The minimum information required is the control, its parent and the type of event. The user session file includes the controls triggered by the user in the same sequence. Such information is an abstract of the user session sequence. In many cases, the same control is repeated several time (due to the nature of logging the window messages), The implementation will get rid of all those controls repeated right after each other. The same information can be extracted from the events written to the event log. In Fig. 1, the control is OK in the parent (i.e form) PageSetup.

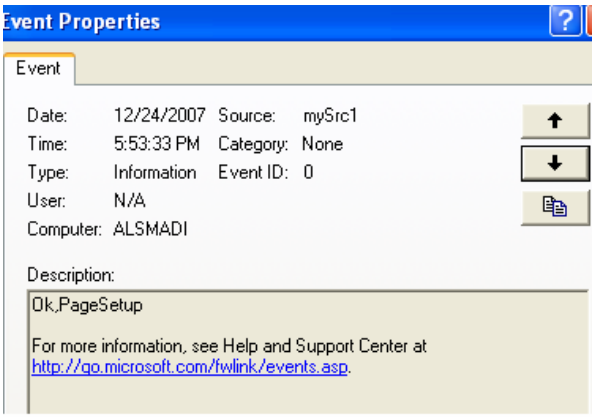

Fig. 1: An event log gathered during a user session.

Table II presents an example output from the developed algorithm for test scenarios' weights. Controls are given weight according to their occurrence in user sessions. The selected scenario includes controls from the different levels. Starting from the lowest level control, the algorithm excludes from selection all those controls that share the same parent with the selected control. This reduction shouldn't exceed half of the tree depth. For example if the depth of the tree is four levels, the algorithm should exclude controls from levels three and four only.

The developed application extracts the logging information in a format that is independent on the application. We used this output as an input to the automated test execution process.

B. Semi test execution and user sessions

In part of a full GUI test automation framework [11], we developed some test execution and verification processes that are performed automatically. In one scenario for verification, test execution is compared with the test cases used as an input for the execution process. As an alternative to comparing the execution suite with the test generated suite, we may compare the log from the execution suite that runs automatically with one that runs by a user. The advantage of this path is that user sessions are generated from real business scenarios, whereas other test case generation and execution, test case are generated by testers (usually from code or requirements). The disadvantage is that it is not automated and a user needs to manually perform the execution of test scenarios. In some cases, this can be triggered only if there are differences between the earlier suites.

Table II: Reduction percentage using user sessions weights.

Test scenarios (Reduction is accumulated from 5 consecutive scenarios)	**Percent of test reduction (%)**
Notepadmain, printer, printerbutton1,,,	
Notepadmain,save,savelabel7,,	
Notepadmain,edit,find,tabcontrol1,tabfind,find tabbtnnext	
Notepadmain,file,print,printtab,printlabel7,	
Notepadmain,save,savelabel5	65.1
Notepadmain,file,print,printtab,printlistbox1	
Notepadmain,font,fontlabel2	
Notepadmain,helptopicform,helptopics,search, button1	
Notepadmain,font,fonttextbox2,,	
Notepadmain, printer, printerbutton2,,,	41.67
Notepadmain,file,print,printtab,printgroupbox 1	
Notepadmain, pagesetup,printer,	
Notepadmain,font,fontlistbox2,,	
Notepadmain,open,openfilelabel4,,	
Notepadmain,saveas,savefilecombobox2,	51.56

This is also considered a hybrid approach between capture/replay techniques and the data model GUI test automation. Rather than making the execution process depends on manual testing (e.g. replay the tests that are created manually), they are running independently and compared with the manual test logging results. The hybrid approach can take the advantage in the capture/ reply mechanism of capturing user sessions. User session-based testing focuses on testing the parts of the application that are normally used by the user.

The advantage of this path over the already adopted capture/replay process is that the tests are object based rather than position based. This overcomes the main disadvantage of using a capture/replay tool in execution and validation as a slight change in the screen properties, changing the display resolution, or changing the control location causes the tests to fail. In the data model, the control is captured through its name and parent. As a result, any changes in the above listed characteristics will not affect locating the control.

IV. CONCLUSION AND FUTURE WORK

Utilizing user session for GUI test automation is discussed in principle in this research. An application is developed to extract user session information in a format that is independent of the tool that gathers them.

The two areas explored were using user sessions for test case prioritization and for test case execution and verification. In future, we will collect actual user sessions' data from different applications. It will be useful to compare the test effectiveness from this technique with other techniques. User sessions represent actual user scenarios, and hence they reflect the application requirements. This makes them as requirements that can be verified automatically.

REFERENCES

[1] Elbaum, Sebastian, Srikanth Karre and Gregg Rothermel, Improving web application testing with user session data, in proceedings of the 25th international conference on software engineering**.** Oregon, USA. Pages: 49 – 59. 2003.

[2] Alshahwan, Nadia. Automatic regression testing of web applications. <http://www.dcs.kcl.ac.uk/staff/mark/PastMScProjects2004/NadiaAlshahwan.pdf>. 2005.

[3] Offutt, Jeff, Ye Wu, Xiaochen Du, and Hong Huang. Bypass testing of web applications. In proceedings of the 15th international symposium on software reliability engineering (ISSRE'04) - Volume 00. Pages: 187 – 197. 2004.

[4] Hicinbothom, J. H., and W. W. Zachary. A tool for automatically generating transcripts of human-computer interaction. In proceedings of the human factors and ergonomics society 37th annual meeting, volume 2 of special sessions: Demonstrations. Page 1042. 1993.

[5] Sreedevi, Sampath. Cost effective techniques for user session based testing of web applications. Phd dissertation. University of Delaware. <128.4.133.74:8080-/dspace/bitstream/123456789/168/1/sampath.dissertation06.pdf>. 2006.

[6] Norman, Donald. The psychology of everyday things. Basic books, 1988.

[7] Chen, Eva. Resource-based user interface design. Phd thesis. The university of York. <www.cs.york.ac.uk/ftpdir/reports/YCST-2005-04.pdf>. 2005.

[8] Elbaum, Sebastian, Srikanth Karre, Gregg Rothermel, and Mark Fisher. Leveraging User-Session Data to Support Web Application Testing. IEEE Transactions on Software Engineering. Pages: 187 – 202. 2005.

[9] Yaya Wei; Chuang Lin; Fengyuan Ren; Dutkiewicz, E.; and Raad, R. Session based differentiated quality of service admission control for Web servers. Page(s): 112 – 116. ICCNMC2003.

[10] Alsmadi, I, and Kenneth Magel. GUI Path Oriented Test Generation Algorithms. In Proceeding of IASTED (569) Human-Computer Interaction. 2007.

[11] Alsmadi, I, and Kenneth Magel. An Object Oriented Framework for User Interface Test Automation. MICS07. 2007.