

2009

PACIFIC NW SOFTWARE
QUALITY
CONFERENCE



MOVING
QUALITY
FORWARD

OCTOBER 27-28, 2009

Conference Paper Excerpt

From the

CONFERENCE
PROCEEDINGS

Permission to copy, without fee, all or part of this material, except copyrighted material as noted, is granted provided that the copies are not made or distributed for commercial use.

Build Robust Test Automation Solutions for Web Applications

Wei Liu
Wei.Liu@erac.com

And

Dawn Wilkins
dwilkins@cs.olemiss.edu

Abstract

Test automation scripts for web applications are often fragile and lead to a high rate of false positive errors, resulting in a large amount of analysis time and maintenance effort. This paper shares our experience in improving test automation scripts for web applications and documents successful lowering of false positive (Type I) error rate which consequently reduced analysis time and maintenance effort. Causes that make test automation scripts fragile are analyzed and solutions are discussed. Notable findings include great benefit to test automation by following simple rules upstream in the software development life cycle, such as the design phase and the development phase.

Biography

Wei Liu is a senior test engineer at Enterprise Holdings, Inc. He has taken the lead role in building Enterprise's new test automation solution since he joined Enterprise two years ago. Wei gained tremendous experience in software development lifecycle and development methodology from his work as senior healthcare information system analyst at Barnes-Jewish Hospital and senior software engineer at EcommLink Corporation. He has a master's degree in computer science and is on track to obtain his doctoral degree in computer science in 2010.

Dawn Wilkins is an associate professor in the Department of Computer and Information Science at the University of Mississippi. She completed her Ph.D. in the Department of Computer Science at Vanderbilt University. Her current research interests are in the area of Bioinformatics, Machine Learning and other more conventional computer science areas.

1. Background

Ideally, any failure of an automated test script would indicate defects in the application under test. However, in reality, automated test scripts are prone to give false positive errors; that is, scripts fail through no fault of the application under test (Fewster 1999, Mosley etc. 2002). Automated test scripts for web applications are especially fragile due to the frequent changes that web applications usually undergo.

False positive errors, also called Type I errors, are costly, resulting in a large amount of analysis time and maintenance effort. Reducing false positive errors is critical in building robust automated test solutions.

Another type of error, the false negative error, is common to automated test scripts too. False negative errors, also called Type II errors, occur when an automated test script fails to find existing defects in the application under test. Reducing false negative errors is of great importance to test automation, but this paper will only focus on false positive errors.

It is worth noting that change requirements should be carefully studied and automated test scripts should be updated accordingly before regression testing of a new release. Errors that are raised by failing to properly update test scripts are not the false positive errors discussed here. The false positive errors discussed here are errors that are caused by unexpected events and cannot be prevented by studying change requirements and updating scripts beforehand.

This paper shares our experiences gained from efforts to improve automated test scripts. Though the applications that we have worked on are web-based, some of the experiences could be applied to automated test scripts in other domains too.

2. Methodology

About 200 false positive errors occurred in the past year and their respective analyses were reexamined. The errors were grouped based on their causes. Then solutions were proposed and applied for every group. The improved scripts were used in later regression tests and false positive errors were monitored and recorded. The frequency of false positive errors for the improved scripts was compared with that of original scripts.

3. Findings and Solutions

3.1 Unrecognizable Components

The most common cause that leads to automated script failures is unrecognizable components due to change of applications-under-test. In automated test scripts, webpage components are identified by combinations of their properties, such as Class, Class Index, Tab Index, Text, Value, Coordinate, ID, Name, etc. If these properties are changed, the components may no longer be recognized.

New features and hot-fixes are constantly added to new releases. GUIs are often re-designed and the look of web pages changes all the time. Not all changes can be predicted and how they affect automated test scripts cannot be determined beforehand. For instance, a developer changes the Name of a component because he thinks the Name sounds silly. Another example is adding a new component to a web page, which automatically changes the tab index property of many components on that page.

To make automated test scripts more robust in the face of changes of applications under test, unreliable properties such as class index, tab sequence index, and coordinates should not be used to identify webpage components. These unreliable properties can easily be changed without notice and it is impossible to track them in the change requests. Therefore, there is no way for automated test script maintainers to discover these changes and update scripts beforehand. These changes are not found until scripts have failed and have been analyzed. Instead, more reliable properties such as ID and Name should be used.

Moreover, the minimal set of properties should be used to identify webpage components. For example, if a component can be identified by its ID or by a combination of its class and text properties, ID should be used. This practice exposes scripts to fewer properties and lowers the chance of false positive errors.

Though sticking to the minimal set of reliable properties can make automated test scripts much more robust, false positive errors caused by unrecognizable components can be further eliminated by following the simple rules listed below at the design and development phases:

- a. Give each webpage component a unique ID or Name, if possible.
- b. Do not change a component's ID or Name unless it is necessary.
- c. Take a screen snapshot or keep a screen design for each screen.
- d. Annotate the screen snapshots or screen designs with component IDs or Names.
- e. Share this document with the Test Automation Team.
- f. Update this document after changes and notify the Test Automation Team about changed IDs or Names.

To successfully enforce the above rules in development phase good communication and coordination between test automation team and development team are necessary. Since these simple rules do not increase the work load of developers in any significant way, they should not receive much resistance from the development team.

3.2 Test Environment Change

Another source of false positive errors is test environment changes, such as operating system updates, web browser updates and automation tool updates. These changes can affect the behavior of the automated test scripts and thus introduce false positive errors. Problems caused by test environment changes can often be detected by a smoke test with a small number of automated scripts. If problems are found during the smoke test, they need to be addressed before the next regression test starts. Usually, this kind of problems can be fixed by software upgrade, reinstallation, or rollback.

3.3 Customizable Contents

Many web applications contain customizable content. Changes of the customizable content often break automated test scripts and raise false positive errors. This situation gets worse when the customized content can be changed by multiple teams, which is often the case in QA environments. In fact, this is a major source of false positive errors of test automation in the web domain. Fortunately, there are several remedies to this problem.

If an agreement that requires customizable content be restored to their default state after they are changed for whatever purpose can be reached with all the other parties involved, such as the manual test team and the development team, it can eliminate most of the false positive errors originated from this cause. However, it is difficult to enforce this agreement in reality.

An alternative is to keep a copy of clean data and use it to restore the application QA database to the “correct” state before each regression test. This solution guarantees the customizable content will be restored to the state that the automated test scripts expect. However, it needs cooperation with the DBA team and may require a lot more time and effort in the case when the database schema has changed since the last release because the saved data can no longer be simply restored.

Usually, a customizable web application has a corresponding tool, which is often web-based as well, to update its customizable content. Automated test scripts for that tool can be used to restore the customizable contents to the “correct” state before each regression test. This solution gives the automation team full control of the data and is immune to database changes. However, the automated test scripts for that tool need to be maintained.

The decision on which solution to use should be made case by case after evaluating all the pros and cons.

3.4 Exception Handling

Lack of flexibility to accommodate all possible situations the application-under-test may undergo is another source of false positive errors. Web applications may encounter many unexpected events, such as a slow network, occasional server error, and so on. Therefore, error checking and exception handling are necessary to deal with these unusual situations.

3.5 Automation Overuse

Test automation is a complicated process. The applications-under-test can be complicated, dynamic and fast evolving. Therefore, it is crucial to know what to automate and what not to automate. A common mistake is to overuse automation, which leads to enormous maintenance work and accordingly high false positive error rate.

Some tests are not good candidates for automation by nature. They either require relatively too much effort to automate or too much time to maintain while they can be easily tested manually. A typical example of this kind of test is cosmetic issues such as page layout, font size and style, and so on, which are often blamed for test failures. They are more suitable for manual test and are usually not worth the effort to automate. Therefore, it is a very delicate job to find a balance between manual testing and automated testing. Failing to do this will result in a large number of false positive errors because maintenance cannot keep up with application changes.

4. Results

The scripts which were improved in accordance with the above discussions on average have at least 80% fewer false positive errors than the original scripts in regression tests.

5. Conclusion

False positive errors are common in test automation. This paper summarizes our experience in writing more robust automated test scripts for web applications. The validity of the findings and effectiveness of the solutions discussed in this paper have been confirmed by test results.

6. References

- [1] Fewster, M., *Software Test Automation*, Dorothy Graham, Addison-Wesley Professional, ISBN 0201331403, 1999
- [2] Mosley, D. & Posey, B., *Just Enough Software Test Automation*, Prentice Hall PTR, ISBN 0130084689, 2002