ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Test Driven Development Using Innovative Testing Framework for Advanced Applications

Kanika Jain, Anjan K Koundinya

Software Engineer, NetAPP, Department of Computer Science and Engineering, R.V.C.E., Bangalore, India

Assistant Professor, Department of Computer Science and Engineering, R.V.C.E., Bangalore, India

ABSTRACT: Writing auto testing is a required engineering technique that can save time and money, and help businesses better respond to changes. But if we use testing framework improperly, more problems would possibly be caused. An auto testing framework based on Selenium and FitNesse is discussed in this article which can help with those problems. The framework use selenium APIs to get page value, DbFit to init database, FitNesse to manage the test fixture, and a special DSL to write test fixture. It could greatly reduce the line numbers of testing code and the project developing period, lower the random error rate, facilitate writing fixture table, improve the coding productivity, and the quality of final product.

KEYWORDS: Auto Testing Framework; Selenium; FitNesse

I. INTRODUCTION

Test Driven Development (TDD) [1], a software development practice used sporadically for decades has gained added visibility recently as a practice of Agile Software Development such as Extreme Programming (XP)[2]. The code developed using a TDD practice showed, during functional verification and regression tests, approximately 40% fewer defects than a baseline prior product developed in a more traditional fashion [3]. TDD requires automatic testing tools and there many open source tools available for users. The auto testing frameworks must allow creation of tests ease otherwise they will not be written. It's needed to provide a reliable definite pass/fail indicator and must have the capability to allow running of tests repeatedly [4]. An auto testing framework is a set of assumptions, concepts, practices and libraries that provide support for automated software testing. The framework can help developers and testers write test codes efficiently and find bugs quickly when errors occur.

Selenium [5] is an object-oriented auto testing library based on page browser, written by folks at Thought Works. Selenium is a suite of tools to automate Web application testing across platforms. Selenium runs in several of browsers and operating systems, and can be controlled by many programming languages and testing frameworks. The use of selenium in a browser to run tests can bring lots of benefits. First, it can be used from the perspective of end users to test applications through the Selenium testing script. Second, it's easier to locate a browser's incompatibilities by running tests in a different browser

FitNesse [6] is an open-source testing and collaboration server, based on the Framework for integrated tests (FIT), and supports testing of code written by Java, .Net, Python and even some other programming languages. FitNesse is a great tool for communicating with customers and collaborating with other people. FitNesse enables customers, testers, and programmers to learn what their software should do, and to automatically compare its features with what it actually does do. It compares customers' expectations with actual results.

II. RELATED WORK

Selenium can be used for test complex AJAX-based Web user interfaces under a continuous integration system. Selenium uses JavaScript and Iframes to embed a test automation engine into a browser. It simulates the users'

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

interactive operations with Web applications. For example, clicking a button and filling out a form are very common in browser operation; you can use Selenium APIs to automate this kind of operations. Although Selenium has many advantages, there are still some weaknesses detected in practice.

- Selenium IDE can be used to record scripts and then run those scripts in the Selenium Core. These scripts is usually of poor readability and very difficult to maintain.
- Selenium Core supports multiple languages, but it needs certain of programming knowledge to writing testing scripts.
- Selenium Core supports multiple languages, but it needs certain of programming knowledge to writing testing scripts.
- The readability of the auto testing code is poor. The link between test page and code is weak. When the testing result has some errors, it is difficult to find the specific reasons for those errors.
- If the structure of the pages changes, testing work, sometimes is just a nightmare, the testing code has to be modified and the cost is very high.

The test case detection in the tool will also assist in detecting the security related threats [13] in the code. The cause may be due to the unintentional exploration of the highly vulnerable elements in the code which may be misused by unknown/know people involved in the process.

III. PROPOSED FRAMEWORK

Web pages, especially those, developed with Web development framework, can be broken down into multiple components, whose characteristics and location in that page are relatively fixed, such as navigation tree, operation bar, search bar and so on. Each of Web pages may not exactly have the same set of components, but their structure is basically the same (see Fig.1).

Figure 1. A simple page in a manage system.

Figure 1: The sample demo of using testing fixture

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

The page in Fig.1 can be divided into four parts; operation bar, search bar, data grid and detail grid. We can develop a testing component for a certain component combining Selenium with FitNesse. Through the interface conversion and error handling, the testing component provides a consistent interface for developers and testers, which makes development of the auto testing easier, more convenient, stable, and maintainable, and easier-to-read. Fig.2 illustrates the architecture of a testing framework. We use selenium APIs to get page value, DbFit to init database, and FitNesse to manage the test fixture.

A. Testing Process in framework

In order to improve the stability and consistency of test, we poll browser a few times per second by using the Wait class from Selenium RC APIs to check for asynchronous events. This process is shown in Fig.2. We normally find error occurring in the testing due to unsuccessful or unfinished page loadings when using Selenium APIs to get page value.

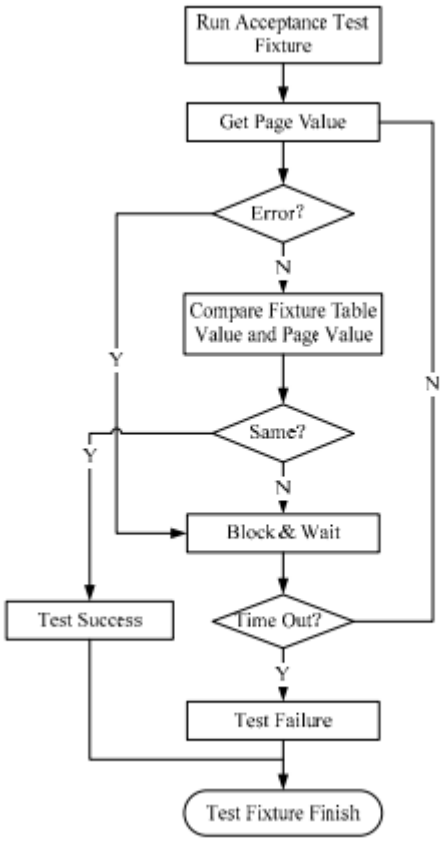


Figure 2: Testing process in framework

The framework can handle error and exceptions in the test period and implement the mechanisms of failure, wait, and retest. In the process of testing catch exceptions and errors, and restart the test until success or timed out. Minimize the failure of the test, and reduce the error rate.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

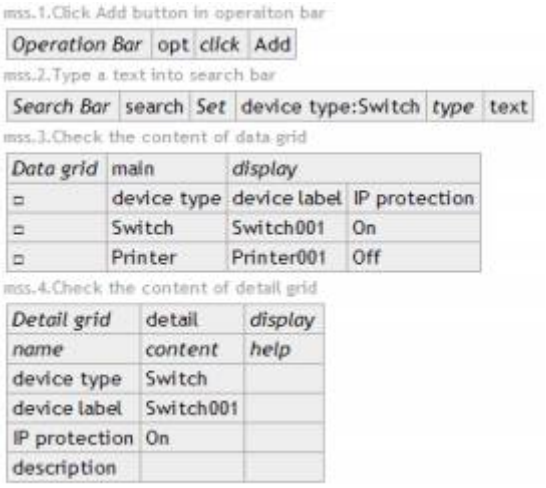
(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

B. Testing DSL

In order to using FitNesse fixtures effectively, we create a Domain Specific Languages (DSL) for the framework that developer and tester can use to specify and verify the expected behaviour of the system under test. FitNesse has the ability to express tests as declarative tables. The FitNesse wrapper gets really handy when you want to perform multiple steps in a single sentence, like repeatedly filling in a form in the same steps but with different input.

In Fig.3, It is observed that the fixture table and the testing page are very similar. The first two fixtures are single-line fixture and are procedural tables that specify the user's interaction with the system [8]. The first cell is the name of a component (Operation Bar / Search Bar), the second cell is a component's ID (opt / search), the third cell is action (click / set), the forth cell is the action's target, and the other cells are additional descriptions that make the fixture complete. The next two fixtures are multi-line fixture and they are declarative tables. The first line is the same as a single-line fixture; the other lines are the same as testing page's content.


mss.1.Click Add button in operaiton bar

Operation Bar	opt	*click*	Add

mss.2.Type a text into search bar

Search Bar	search	*Set*	device type:Switch	*type*	text

mss.3.Check the content of data grid

Data grid	main	*display*	
□	device type	device label	IP protection
□	Switch	Switch001	On
□	Printer	Printer001	Off

mss.4.Check the content of detail grid

Detail grid	detail	*display*
name	*content*	*help*
device type	Switch	
device label	Switch001	
IP protection	On	
description		

Figure 3: The architecture of testing framework.

As what is shown in Fig.3, we use component ID instead of xpath because component ID is easier to obtain and test can run faster [9] at usual. We can simply write testing fixturewith native language styled DSL rather than code such as Java, .Net and so on. In this way, everybody can write the test fixture faster and more easily, and we could increase efficiency and reduce costs in the project. If we manage the user story and user request by FitNesse, we could link the user request with the test..

IV. RESULTS

With linkage Available, an instant monitoring on the test with what the fixtures can be tested, when errors are detected, and the reason for detected errors, and with this information at hand, an in-time and fast error-fixation is possible. The grey word in Figure 4 is a part of the user story and the results achieved from the same.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Figure4:Selenium V/SPorWebFixture

The selenium framework can be use as Web auto testing framework at the initial stage (version 5.0 and 5.1). Further there can be change to the new testing framework which is called PorWebFixture described above (version 5.2 and 5.3). A huge data is collected to compare the two test frameworks, the result is shown blow in Fig.5. Through a significant increase in test cases coverage, there is reduction in the testing defect density, bringing testing defect density from 2.87D/KLOC down to 1.25D/KLOC. The detail data is shown in Table 1.

TABLE 1: TESTING DEFECT DENSITY

Version	Testing Defects	Testing Defects Density(K/LOC)	Testing Period(H)
5.0	564	2.87	2340
5.1	49	2.79	831
5.2	77	1.47	1475
5.3	98	1.25	1910

By using fixture table, we can execute multiple steps specified in a single sentence, which reduces the total line number of testing code thus shortening the total project period greatly, which is about one-third of the time spent in the older solution. The implemented mechanisms of failure, wait, and retest effectively lower the random error rate.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Writing fixture table with DSL is easier than other programming language; we improve the coding productivity and test case coverage. The pause method [10] in Selenium can reduce test errors. But with the mechanisms of failure, wait, and retest available, we disuse the pause method in PorWebFixture, which makes the test run more quickly and take less time than before.

V. CONCLUSION AND FUTURE WORK

Auto testing is an effective way to improve the project quality. In order to write and use auto testing in a better way, we must choose suitable frameworks and right method. In this article we construct a useful testing framework which combines the strength of Selenium and FitNesse. It achieves the goals which we expect and will play a greater role in the new projects to come.

ACKNOWLEDGEMENT

I take immense pleasure and very deep sense of gratitude in conveying my special, sincere and heartfelt thanks to my guide **Prof. Anjan K. Assistant Professor, Department of Computer science and Engineering, R.V.C.E., Bengaluru.**

Prof. Anjan K Koundinya would like to thank Late. Dr V K Ananthashayana, Former Head, Dept. Of CSE, MSRIT, Bengaluru for igniting the passion for research.

REFERENCES

[1]B. George, L. Williams, "An Initial Investigation of Test-Driven Development in Industry", ACM SAC, Mel, FL, 2012.
[2]J. Rasmusson, "Introducing XP into Greenfield Projects: Lessons Learned," IEEE Software, May/June, 2012, pp. 21-28.
[3]L. Williams, E.M. Maximilien, and M. Vouk, "Test-DrivenDevelopment as a Defect-Reduction Practice,"Proc. 14th Int'l Symp.Software Reliability Eng.(ISSRE 03), IEEE Press, 2010, pp. 34-45.
[4] B.S. Mattu, R. Shankar, "Test Driven Design Methodology for Component Based System", 2007 1st Annual IEEE Systems Conference, April 2007, pp.1-7.
[5] Article on Selenium at the URL http://seleniumhq.org/ last visited on 21 May 2014 at 12:30.
[6] Article on FitNesse at the URL http://www.fitnesse.org visited on 22 May 2014 at 12:30
[7]Johannes Ryser, Martin Glinz, "A scenario-Based Approach to Validating and Testing Software Systems Using Statecharts", 12th International Conference on Software and Systems Engineering and their Applicaions ICSSE 99
[8] R. V. Binder, "Testing Object-Oriented System Models, Patterns, and Tools", NY: Addison-Wesley, 1999.
[9] Kelly D.P. and Oshana, R.S., "Improving software quality using statistical testing techniques", Information and Software Technology, 42(12):801-807, 2000.
[10] Whittaker J. A and Thomason, M. G., "A Markov Chain Model for Statistical Software Testing", IEEE Transactions on Software Engineering, 20(10):812-824, 1994
[11] Kirk Sayre. Improved Techniques for Software Testing Based on Markov Chain Usage Models. PhD thesis, University of Tennessee, Knoxville, December 99.
[12] J. Grabowski, A. Wiles, C. Willcock and D. Hogrefe. On the design of the new testing language TTCN-3. Proceedings 13'h IFIP International Workshop on Testing Communication Systems (TestCom 2000), Ottawa, August 2000.
[13] Anjan K, Srinath N K, Jibi Abraham, Exploration of covert schemes and their embodiments in Hybrid Covert channel, International Journal of Advances in Computer Networks and Security (IJCNS), Vol. 5, Issue 2, ISSN-2250 – 3757, Oct 2015, pp.50-54.

BIOGRAPHY

Kanika Jain has received her M.Tech degree in Computer Network and Engineering from R V College of Engineering, Bangalore, India in 2014. Her areas of research interests include Storage and investigate the Automation Testing. She is currently working as a Software Engineer at NetApp India Pvt. Ltd. Her Job profile is to work on various configuration of Storage solutions.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Anjan K has received his B.E degree from Visveswariah Technological University, Belgavi, India in 2007 And his master degree from Department of Computer Science and Engineering, M.S. Ramaiah Institute of Technology, Bangalore, India. He has been awarded Best Performer PG 2010 for his academic excellence. He is pursuing Ph.D in Computer Science and Engineering from VTU, Belgavi. He is currently working as Assistant Professor in Dept. of Computer Science and Engineering, R V College of Engineering, Bengaluru, India.