Dynamic Heuristic Approach of An Acceptance Test Data Generation with Behavior Driven For Selenium Test

Vivek Kaushikbhai Shah

Abstract- Automated acceptance testing is a quite recent addition to testing in Dynamic Heuristic Approach holding great promise of improving communication and collaboration. This paper summarizes existing literature and also presents a case study from industry on the use of automated acceptance testing with Easyb. The aim of this paper is to establish one of Selenium's stated goals is to become the de facto open-source tools Groovy and easyb. In particular interest to the Dynamic community is that it offers the possibility of test-first design of web applications, Pass/Fail result for customer acceptance tests, and an automated regression test bed for the web tier.The process of risk analysis with this tools are continuous and applies to many different levels, at once identifying system-level vulnerabilities, assigning probability and impact, and determining reasonable mitigation strategies. By considering the resulting ranked risks, business stakeholders can determine how to manage particular risks and what the most cost-effective controls might be that were determine effectiveness of acceptance tests.

1 Introduction

Automating acceptance tests is one of the traditional problems facing software development projects. Whereas automated UI and Unit testing has achieved deep traction, in Dynamic and non-Dynamic projects alike, acceptance testing frequently is still done manually. In recent years, tools such as easyb and groovy helped make it easier to automate acceptance testing of software applications. However, web applications have long remainedDifficult to test because of multitier architecture, multiple browsers, and web technologies such as JavaScript.It is very expensive in manual testing. Selenium, originally developed by thought Works, has gained attention as a possible functional and acceptance problem of automated testing for Web Applications. We have been using Selenium on five different web browsers. In addition, we have used Easyb is a behavior driven development framework for the Java platform. By using a specification based Domain Specific Language; easyb aims to enable executable, yet readable documentation.Easyb specifications are written in Groovy and run via a Java runner that can be invoked via the command line. What's more, easyb supports a few different styles of specifications ranging from RSpec's it to a story based DSL with givens, when and then. In dynamic languages, Groovy is a dynamic language for the Java Virtual Machine builds upon the strengths of Java but has additional power features inspired by languages like Python, Ruby and Smalltalk makes modern programming features available to Java developers with almost-zero learning curve supports Domain-Specific Languages and other compact syntax so your code becomes easy to read and maintain makes writing shell and build scripts easy with its powerful processing primitives, OO abilities and an Ant DSL increases developer productivity by reducing scaffolding code when developing web, GUI, database or console applications simplifies testing by supporting unit testing and mocking outofthebox seamlessly integrates with all existing Java classes and libraries compiles straight to Java byte code so you can use it anywhere you can use Java.

1.1 Background

Our teams practice a dynamic Heuristic language; we are committed to writing acceptance tests in Bit bucket before beginning development on stories. Passing these tests signals completion of a story, so we write functional tests that capture these acceptance criteria. For a web application, a functional test could be simply that a user manually navigates through the application to verify the application behaves as expected. We try to automate our application in acceptance tests whenever we can. And our holy groovy is to be able to write the tests before development, so that the development team can have a runnable verification that the story is complete. Our team came to Selenium after using several other tools to automate web testing. We tried the open source tools Groovy and easyb them to be sufficient, as they could not handle most instances of in-page JavaScript. The JavaScript problem is solved by the behavior driven development and record-and-play of test scripts and also runs tests directly in a browser.

1.2 Selenium with Easyb

Since Selenium was introduced a few years back, it has continued to wow developers with how easily a user acceptance test can be knocked outsimply fire up an instance of a Selenium server in the background and then either write a table. Web driver style testing is particularly powerful as you have full access to programming languages for instance, with RC, you can write a functional web test in Java by leveraging a framework like JUnit or TestNG. But what's often lacking with testing frameworks is a more natural way of expressing behavior or indeed, scenarios and stories. For instance, a user acceptance test is really a scenario a user logs into a website, purchases an item, pays, and logs out. That was a sunny day scenario there are other scenarios that deal with various other paths user fails to pay, credit card was invalid, etc. All of these scenarios are logically a storya story about buying something. Using a standard scenario language, one can more specifically write a scenario (in a story regarding a website Using a standard scenario language, one can more specifically write a scenario (in a story regarding a website for race registrations) like so:

- Given a user is on the Dachisgroup login page
- When someone fills username and password in the report who has signed up for home pages
- That is a happy day scenario isn't it? One particular negative path would be:
- Given a user is on the Login page
- When someone fills username and password in the report who hasn't signed up for Home pages.

Using easyb then, one can create a story file, which contains two scenarios the file could be called Login.
Story and will have two scenarios:
Scenario "a valid person has been entered", {}
Scenario "an invalid person has been entered", {}

1.3 Getting Started

Selenium is easy to setup in any computer, although there is a catch. The basic installation of Selenium must be hosted by the same web server as the Application under Test (AUT) for acceptance testing. In advanced level of easyb comes with a command line runner that takes the name of a particular behavior or story you wish to run. You can optionally pass in a few different flag options to output various report formats as well.

```
c:>javaorg.easyb.BehaviorRunnermy/path/to/MyStory.groo
vy
```

2 Literature Survey

2.1 Selenium Open Source Tool

Selenium is a set of different software utensil each with a different approach to supporting test automation. Most Selenium QA Engineers focus on the one or two utensil that most meet the needs of their project, however learning all the tools will give you many different options for approaching different test automation problems. The entire suite of tools results in a rich set of testing functions specifically geared to the needs of testing of web applications of all types. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One of Selenium's key features is the support for executing one's tests on multiple browser platforms.

2.2 Easyb for BDD

You have explored the language's approach to BDD and seen how it helps you think about your system as a collection of behaviors across various levels of granularity. By keeping the focus on behaviors instead of tests for specific classes and methods, BDD and easyb encourage you to write tests that are more descriptive of what the system should do and less tightly coupled to particular implementations.Additionally, easyb's approach for capturing user stories is less restrictive than the programmer oriented syntaxes of comparable tools. Easyb aims to stay out of the way of the story writing process so that the center of attention is on the conversation taking place, not on the tool that is being used to capture that conversation. Furthermore, easyb does make it easy to bind these expressions of behavior to the system under test as a mechanism for validating the conversations. As easyb draws closer to its 1.0 launch, now is a great time to give it a test drive and see for yourself how BDD and easyb can help you build better systems that more closely match your customers' needs.

2.3 Value of Heuristic with Behavior Application

In software testing, loops are important spot for error detection. Execution of program spend large amount of time in loops. Without covering Xpaths going through loops we cannot get better code coverage. Most of the mistakes are made in loops of programs. Infinite loop creates lots of problem in detecting the errors. In fact, Value of Heuristic is possible to factorization of risk in Dachis application with different behavior. It is impossible to detect all kinds of infinite looping automatically. Test data generation is more challenging if loops are nested. Automated test data is generated using symbolic value, actual value, and combining both. One of the main problems in test data generation is detection of infeasible path. Statistics reveals that many paths of a program can be infeasible. The behavior execution method is simpler to execute any scenarios. Infeasible path detection due to non-availability of efficient constraint solver and path feasibility detector.

2.4 Test First?

Selenium behavior tests are easy to write, a tester or analyzer can write the shell of Selenium with easyb test very quickly without knowing what the implementation will be. Although we hoped that we could code to these tests, the test would seldom turn Pass after development. The reasons for this were usually minor: a field wasn't naturally identified by the name or Xpaths the tester chose, or the test command used needed to be waiting Ajax or click event on instead of just click, etc. As a result, we did not usually require that the developer code to the test, and our process of writing the test before development (for its acceptance value), but getting the test behavior pass immediately after development, emerged.

2.5 IDs, Xpath & Acceptance

The project we used Selenium for was a JAVA project, which automatically assigns IDs for every element on the page. However, when we switched to JSP-based applications, the IDs were only present when the code specified it. Selenium supports several different techniques for identifying page, including names, IDs, Xpath, Element CSS and more. For the most part it was not difficult to find unique identification for an element, but occasionally parsing the HTML was tedious. Some tools we found like fire finder in Firefox browser valuable for this purpose were the DOM inspector, Xpath hacker. Selenium IDE is even better at this task. However, frequently these tools do not find the "easiest" way. This would frequently contribute to longer-running tests, which was one of the most significant problems we encountered using Selenium. In general, using Xpath expressions tends to make tests take much longer to run, especially in Internet Explorer. So we used IDs or names where they existed, and went out of our way to add them into our JSP code when possible. Due to the processing and memory needs of browsers running Selenium, as our suites grew larger and contained more tests, we needed to have a more robust environment. For example, one suite running on a VPNVista600 MHz machine with 1GB of RAM took over 1hr and 40Minutes to run. Upgrading our test environment to a VPN to Wing 7, 3 GHz machine with 2GB RAM took the time required down under 30Minutes.

2.6 Ability to test the entireacceptance Requirement

We have reused a partial number of testing tools in the past and Selenium with easyb is among the best, if not the best, at being able to perform every browser for acceptance testing action that a user can perform, including such events as on Mouse Over and on Key Press acceptance testing. In addition, because Selenium allows users to write their own extensions, it is easy to create custom actions that do sophisticated manipulations. There are some JavaScript-restricted actions, such as downloading or uploading files, that are not supported, but even for creating custom methods of supported these actions have some workarounds uploading and handling on window. Easyb and groovy are the techniques. One recent addition to the Selenium world, Selenium Remote Control, addresses some of these issues by allowing the user to write Selenium acceptance tests in other programming languages like ASP.Net, Perl and others, thus leveraging the power of Selenium within more traditional automated acceptance tests.

3 Running Easyb Plugin in Eclipse and Writing Effective Test

We have created an easyb plugin for Eclipse that makes it super easyb to work with easyb specifications and to run them. Installing the plugin works as follows:

- Go to the Help menu and then select Software Updates...
- Hit the Add Site. button and in the resulting dialog, for the location type: http://easyb.googlecode.com/svn/trunk/eclipse-plugins/org.easyb.eclipse.updatesite/
- Hit the Ok button
- Select the easyb Eclipse Update Site in the list box and then hit the Install... button

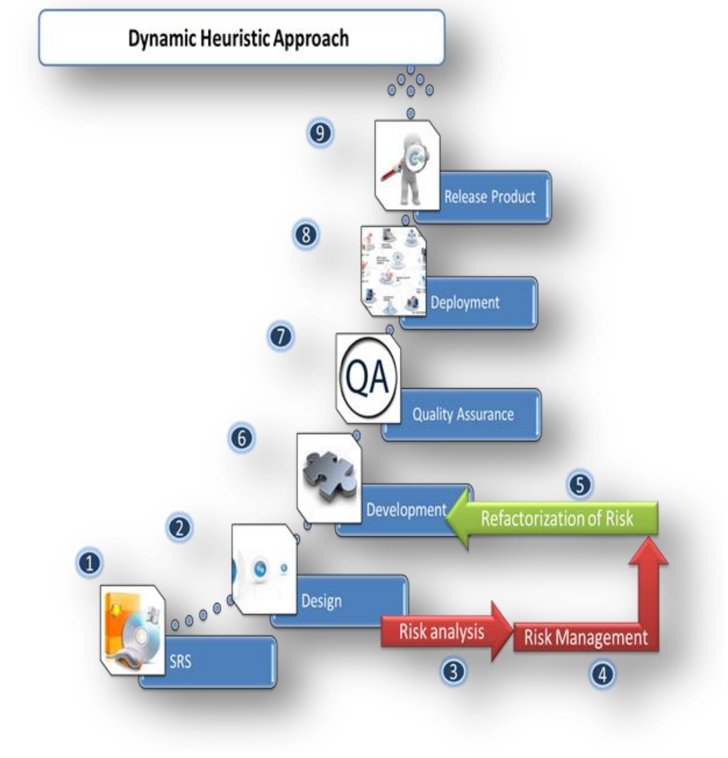
For writing effective test: BDD with dynamic approach principles support the notion of stories quite nicely-- you can think of a story as narrative between a stakeholder and development (almost like a use case). In short, think of a story as a description of a requirement, which has an associated benefit and criteria for validation. Stories can be made up of scenarios that group specifications. The specifications are essential-- they are essentially steps that are friendly to read. They are:

- Given (a context)
- When (something happens)
- Then (something else happens)

Stories in action: The default convention for stories in easyb is to place each story in a file ending with Login. Story. So if you have story regarding shipping calculations, for example, you'd have a file named Login. Story. The code below shows a story in easyb in action-the code has two scenarios which reside in a story file named. **EmptyStack.dachisgroup.story.**

```
import.org.easyb.bdd.dachisgroup
scenario "null is pushed onto empty stack",
{
given "an empty stack",
{
stack = new Stack ()
}
when "null is pushed",
{
push null =
{
Stack. Push (null)
}
}
```

3.1 Our Approach: Dynamic Heuristic



Steps of Dynamic Heuristic Approach:

[1] SRS:

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform.

[2] Design:

Software design is a process of problem solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes low-level component and algorithm implementation issues as well as the architectural view.

[3] Risk Analysis:

Risk analysis is a technique to identify and assess factors that may jeopardize the success of a project or achieving a goal. This technique also helps to define preventive measures to reduce the probability of these factors from occurring and identify countermeasures to successfully deal with these constraints when they develop to avert possible negative effects on the competitiveness of the company. Reference class forecasting was developed to increase accuracy in risk analysis.

[4] Risk management:

The process of identification, analysis and either acceptance or mitigation of uncertainty in investment decision-making. Essentially, risk management occurs anytime an investor or fund manager analyzes and attempts to quantify the potential for losses in an investment and then takes the appropriate action (or inaction) given their investment objectives and risk tolerance.

[5] Refactorization of Risk:

Refactorization is to factor out repeated coding patterns into new abstractions and thus avoid their repetition resulting in less code to maintain and finite substitute implementation for any types of risks. We have refectories all risks one by one. Major part of risks included in this part of step.

[6] Development:

Term "software development" may be used to refer to the activity of computer programming, which is the process of writing and maintaining the source code, but in a broader sense of the term it includes all that is involved between the conception of the desired software through to the final manifestation of the software, ideally in a planned and structured process

[7] QA/QC:

Quality assurance and Quality control has been re-framed and re-worded by different quality experts from time to time. It also varies from industry to industry.

[8] Deployment:

Software deployment is all of the activities that make a software system available for use. The general deployment process consists of several interrelated activities with possible transitions between them. These activities can occur at the producer site or at the consumer site or both. Because every software system is unique, the precise processes or procedures within each activity can hardly be defined. Therefore, "deployment" should be interpreted as a general process that has to be customized according to specific requirements or characteristics.

[9] Product Release:

Product Release term used when software is ready for or has been delivered or provided to the customer.

3.2 Writing our own Framework:

WebDriver uses a different underlying framework from Selenium's JavaScript Selenium-Core. It also provides an alternative API with functionality not supported in Selenium-RC. WebDriver does not depend on a JavaScript core embedded within the browser; therefore it is able to avoid some longrunning Selenium limitations.

WebDriver's goal is to provide an API that establishes:

- A welldesigned standard programming interface for Web-app testing.
- Improved consistency between browsers.
- Additional functionality addressing testing problems not well-supported in Selenium- 1.0.
- Framework provides Multi-browser testing including improved functionality for browsers not well-supported by Selenium-1.0.
- Handling multiple frames, multiple browser windows, popups, and alerts.
- Page Navigation.
- Drag-and-drop.
- AJAX-based UI elements.

4 Putting Acceptance Selenium Test into Dachisgroup:

After writing acceptance tests to have a successful implementation for our Dachisdev projects, we found that there were a few recurring critical points:

- Keeping the tests organized was a risk analyze;
- Writing the tests in JAVA was unnatural and;
- Using variables, Xpaths across all multiple tests was tricky.

We realized early on that using variables would prevent us from having to change every test when a field ID, metrics would change, but it was tricky enough that we weren't doing it. By leveraging the power of our project of Dachisgroup. A Dachisgroup is a collaborative web environment where any user can change the pages. It is our company's standard project of brand's social performance center where we capture our companies' entire competitor, brand and manage portfolio.

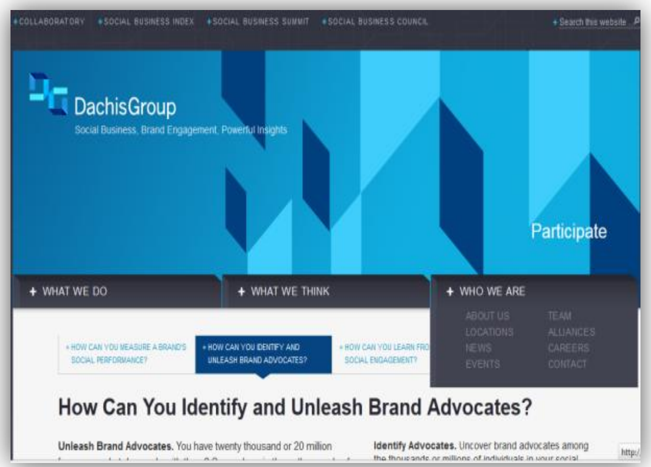
4.1 Groovy & Easyb

Groovy simplifies easyb testing, making it Groovier, in several ways, including:

- Easyb is built into the groovy runtime, so you can script easyb tests for your Groovy and Java classes using Groovy syntax.
- Groovy provides many additional easyb assertion statements like, should, should be and should not be etc...
- Groovy unit tests are easily scriptable with Ant / Maven

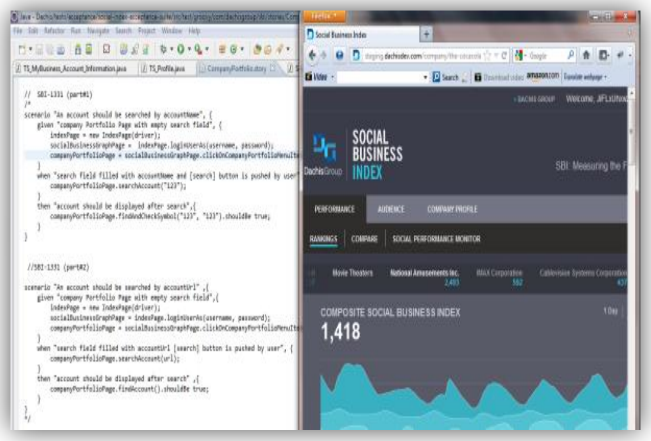
4.2Implementing Easyb and Groovy

Groovy is like a super version of Java. It can leverage Java's enterprise capabilities but also has cool productivity features like closures, builders and dynamic typing. If you are a developer, tester or script guru, you have to love Groovy. Running application with easyb and groovy.


COLLABORATORY
SOCIAL BUSINESS INDEX
SOCIAL BUSINESS SUMMIT
SOCIAL BUSINESS COUNCIL
Search this website
DachisGroup
Social Business, Brand Engagement, Powerful Insights
Participate
WHAT WE DO
WHAT WE THINK
WHO WE ARE
ABOUT US
LOCATIONS
NEWS
EVENTS
TEAM
ALLIANCES
CAREERS
CONTACT
HOW CAN YOU MEASURE A BRAND'S SOCIAL PERFORMANCE?
HOW CAN YOU IDENTIFY AND UNLEASH BRAND ADVOCATES?
HOW CAN YOU LEARN FROM SOCIAL ENGAGEMENT?
How Can You Identify and Unleash Brand Advocates?
Unleash Brand Advocates. You have twenty thousand or 20 million
Identify Advocates. Uncover brand advocates among

4.3 Eclipse behavior

It's a behavior of Dachisgroup application.


File Edit Refactor Run Navigate Search Project Window Help
TS_MyBusiness_Account_Information.java
TS_Profile.java
CompanyPortfolio.story
SOCIAL BUSINESS INDEX
PERFORMANCE
AUDIENCE
COMPANY PROFILE
RANKINGS
COMPARE
SOCIAL PERFORMANCE MONITOR
COMPOSITE SOCIAL BUSINESS INDEX
1,418

```
//  SBI-1331 (part#1)
/*
scenario "An account should be searched by accountName", {
    given "company Portfolio Page with empty search field", {
        indexPage = new IndexPage(driver);
        socialBusinessGraphPage =  indexPage.loginUserAs(username, password);
        companyPortfolioPage = socialBusinessGraphPage.clickOnCompanyPortfolioMenuIt
    }
    when "search field filled with accountName and [search] button is pushed by user"
        companyPortfolioPage.searchAccount("123");
    }
    then "account should be displayed after search",{
        companyPortfolioPage.findAndCheckSymbol("123", "123").shouldBe true;
    }
}


 //SBI-1331 (part#2)

scenario "An account should be searched by accountUrl" ,{
    given "company Portfolio Page with empty search field",{
        indexPage = new IndexPage(driver);
        socialBusinessGraphPage = indexPage.loginUserAs(username, password);
        companyPortfolioPage = socialBusinessGraphPage.clickOnCompanyPortfolioMenuIt
    }
    when "search field filled with accountUrl [search] button is pushed by user", {
        companyPortfolioPage.searchAccount(url);
    }
    then "account should be displayed after search" ,{
        companyPortfolioPage.findAccount().shouldBe true;
    }
}
*/
```

5 Conclusion

Risk analysis is, at best, a good general-purpose yardstick by which we can judge our security design's effectiveness. Because roughly 50 percent of security problems are the result of design flaws, performing a risk analysis at the design level is an important part of a solid Software security program. Taking the trouble to apply risk-analysis methods at the design level for any application often yields valuable, business relevant results. The process of risk analysis is continuous and applies to many different levels, at once identifying system-level vulnerabilities, assigning probability and impact, and determining reasonable mitigation strategies. By considering the resulting ranked risks, business stakeholders can determine how to manage particular risks and what the most cost-effective controls might be.Via using Dynamic heuristic approach risk of the application is well balanced in acceptance testing. It handles many of the problems very well and doesn't add significant new ones. The active and growing communities of users have Easyb and groovy to fill a need for a variety of different user types and widespread adoption seems imminent. Selenium is certainly worth evaluating for anyone looking to add a powerful web acceptance testing tool to their toolkit.

6 Acknowledgement

We would like to thank *Prof. M. B. Chaudhari* for his valuable feedback and support in reviewing this paper throughout its evolution. Additional thanks to *Prof. B. V. Buddhdev*for his encouragement and advice.

7References

[1] Software Engineering & Methodology:
A Practitioner's Approach, 7/e
By, Roger S. Pressman
[2] Dachisgroup:
(Website: http://www.staging.dachisdev.com)
[3] Social Business Index:
(Website: http://www.socialbusinessindex.com)
[4] Selenium:
(Website: http://www.openqa.org/selenium)
[5] Easyb:
(Website: http://www.easyb.org/)
[6] Groovy:
(Website: http://groovy.codehaus.org/)
[7] Risk Analysis in Software Design
(Published By the IEEE Computer Society, 1540-7993/04)