Webinar Highlights: Open Source Test Automation: Riding the Second Wave

David Dang, Zenergy's VP of Automation Solutions, provides valuable information on what factors should be considered when evaluating an open source vs. packaged automation solution. He delves into the pros and cons of both types of automation solutions and gives detailed insight into both the personal and organizational mindsets that will ultimately make the adoption successful.

What is an open source test automation framework/tool?

- Free software that is used and shared by the community
- Currently on www.opensourcetesting.org/functional.php, there are around 135 open source test automation tools/frameworks available

Why is open source test automation gaining in popularity?

- It is FREE!!!
- IoT, Agile, Popularity of Open Source Development tools
- More applications are being converted to web-based
- Development teams are pushing for it
- Companies are getting more comfortable with open source. (Eclipse, Java, Jenkins, OpenStack, Drupal, etc)

First Wave of Open Source Test Automation Tools/Frameworks

- **FitNesse:** Operates below the user interface level. The user provides various inputs to the application and determines if the correct results are returned.
- **Ruby Watir:** Ruby libraries used to automate web browsers. It allows users to automate clicks, fill in forms, press buttons, etc.
- **Selenium RC**: Writes automated web application UI tests. RC comes in two parts: server acts as an HTTP proxy for web, and client libraries with the web elements.

Why did the First Wave Taper Off?

- Tools were too fragmented, difficult to integrate into complete solutions
- Level of instrumentation increased the technical challenge and decreased reliability

- Learning curve tended to be higher due to overly complex and less refined solutions
- IT landscape wasn't ready to embrace open source at the time

Current Open Source Test Automation Tools/Frameworks

- **Selenium WebDriver**: Drives a browser directly using its built-in support for automation. ****This is a major benefit of using Selenium. **. Can create automation using different languages: Java, Python, C#, PHH…etc.** *If you work in a company that uses one of those as primary development language, you can go to development for help.*
- **Cucumber**: Runs automated acceptance tests written in a behavior-driven development (BDD) style. Write in English, and turn into automation script. Uses Ruby to create step definition *(Can also use with Selenium)*
- **Robot Framework**: Generic test automation framework based on keyword-driven testing approach. The core framework is implemented using Python

Why are the Second Wave Tools More Popular?

- WebDriver laid foundation for numerous additional tools
- Increased support from well-known organizations
    - WebDriver
        - W3C has adopted it as the standard for browser automation
        - Protractor is built on top of Selenium WebDriver and allows you to work with Angular code.
    - Appium (framework to work with mobile devices) built on top of Selenium WebDriver to test Mobile. Supported by Sauce Labs.
- Supports popular methodologies (Ex. Cucumber BDD)
- Cleaner abstraction layer and more concise API's
- Less instrumentation required
- Better integration with other toolsets (Jenkins, Junit)
- Integration with CI/CD is important. Selenium is a valuable part of this integration.

Selenium RC vs Selenium WebDriver: Architecture

- Selenium WebDriver interacts directly with browser
- Commands are easier with WebDriver than with RC
- Easier to work with AUT (Application Under Test)

Differences between Open Source and Packaged Tools

- **Environment:** Packages tools are integrated. Selenium is more of a framework. It is made up of a number of packages/utilities. Capabilities have to be loaded into an IDE. Need IDE to actually access utility of that framework. You can't just open Selenium and start coding tests.
- **Browser Compatibility:** Open Source handles this better than packaged tools. Users in the community code up what they need and share with others in the community.
- **Language Support:** Selenium supports more languages than package tools
- **Reporting:** With Selenium you might have to download some custom reporting to give meaning to execution. Looking at stack trace is very difficult to track down issues. With packaged tools, reporting is included.
- **Test Management:** Package tools have integration with ALM/QC. With Selenium, you would have to write your own open source architecture to upload results back into ALM/QA.
- **Continuous Integration:** Package tools are integrated, but with Selenium have to use separate packages like Jenkins or Bamboo.
- **Support:** Packaged Tools have a Customer Support 800 number whereas with Open Source, support comes from the Community. Before reaching out to the Community, you must do your own due diligence to try and research/resolve issue.
- **Object Management:** Packaged tools have this included. With Selenium, you will have to build your own Object Map or GUIMap
- **Test Control:** Packaged Tools include this but in Selenium you have to build your own separate package such as TestNG, phpUnit, RSpecu

What are the challenges?

People

- Requires more technical resources

- Resources must be comfortable with object-oriented programming
- Resources must understand framework/tool configuration and environment setup
- Resources must be able to diagnose and fix technical issues without vendor support
- May have to identify and even fix bugs in the tool
- May need to contribute to the community with new features
- May need to work more closely with development team **(No support to call. Would have to ask Dev to expose widget, etc…) You will build camaraderie with Development! This is great in an agile environment.**

Technology

- Open source requires more time and effort to get started and maintain
- No official support, only the community
- Components are not built-in; they must use other libraries or be developed from scratch
- Some technologies are not well supported by open source tools
- Some technologies have features that lack support from otherwise useful open source tools

Process

- Requires more dedicated resources
- Tool should fit your existing process
    - For example, if you are .NET shop you should use .NET
    - Tool like Cucumber will have greater value if adopted early in the SDLC
- Requires a detailed plan for implementation and maintenance
- Requires a test automation approach such as page object models or keyword

Framework Development Process

- Gather Information
- Architecture Design
- Core Framework Implementation
- Ongoing Development
- Ongoing Maintenance

Should you Catch the Second Wave?

Before you catch the Second Wave you need to look at the following:

- **Process**
  - Do you have the dedicated resources to ensure that automation is successful?
  - Do you have a plan for creating stable, maintainable framework?
  - From a budget standpoint does Open Source make sense?
  - Does Open Source fit with your organization's overall IT strategy?

- **People**
  - Is your team comfortable with object-oriented programming?
  - Can you team deal with technical challenges and minimal support?
  - Can your team contribute to the tool with bug fixes and enhancements?
  - Is there a plan for training and mentoring?

- **Technology**
  - Did you choose a tool that fits with your platform?
  - Is the Community for that tool active? Can you turn to it for help?
  - Is the tool updated frequently and responsive to platform changes?
  - Will investing in open source line up with your organization's technology road map?
  - Are you comfortable supporting multiple tool sets?

Q&A-The following Questions came from the attendees:

- **Can you use Selenium WebDriver on a non-Web application?**
  - No, it can only be used on Web applications
- **Can you use Selenium packages on Eclipse?**
  - Yes you can use as an IDE
- **Is Robot framework screen shot from isps/Mainframe screen? Or did you specifically use those colors?**
  - Creating test and running from command line

- **My company is transitioning from QTP to Selenium and we are struggling because it seems more technical and piecemeal. Do you have any suggestions?**
    - Look at the People, Process and Technology and pinpoint challenges
    - Make sure that you understand the objective of Open Source. If you want to mirror QTP you will have to mirror some of the packages.

- **Where should automated functional test cases be placed?**
    - Depends on how many test cases- You can integrate tests as part of your project or put them in their own package. (Less than 1,000 tests, put in their own package)
    - If there are more than 1,000 test cases, you can create driver concept where you access the test cases from another package
    - If you are using tests to integrate with other applications, you will need to take into consideration when you build framework.

- **We have integrated open source with CI/CD and are having problems running them continuously. Do you have any suggestions?**
    - In the old world, tests would fail and then be re-executed
    - Now, project teams sees failures too
    - Look at automation suite and see which tests can run with no manual intervention. Can't have false negative. In some cases, object might take too long to render, so test fails. Be a lot more focused on your testing. Fine-Tune so that you are hitting the main point of the test cases.