Automation Testing Experience with Blaise

Kay Brenner, Gayu Subramanian and Rittu Jittu, Westat

Presenter: Mangal Subramanian

1. Abstract

Automation testing provides many benefits to projects that include time and cost savings, efficiency with regression testing, and consistency across test runs. It also provides better scalability and analysis reports on test coverage. We review our experiences using various automation solutions with Blaise 4.8 and 5, show several demonstrations of these products, and discuss the advantages and disadvantages of the different approaches investigated.

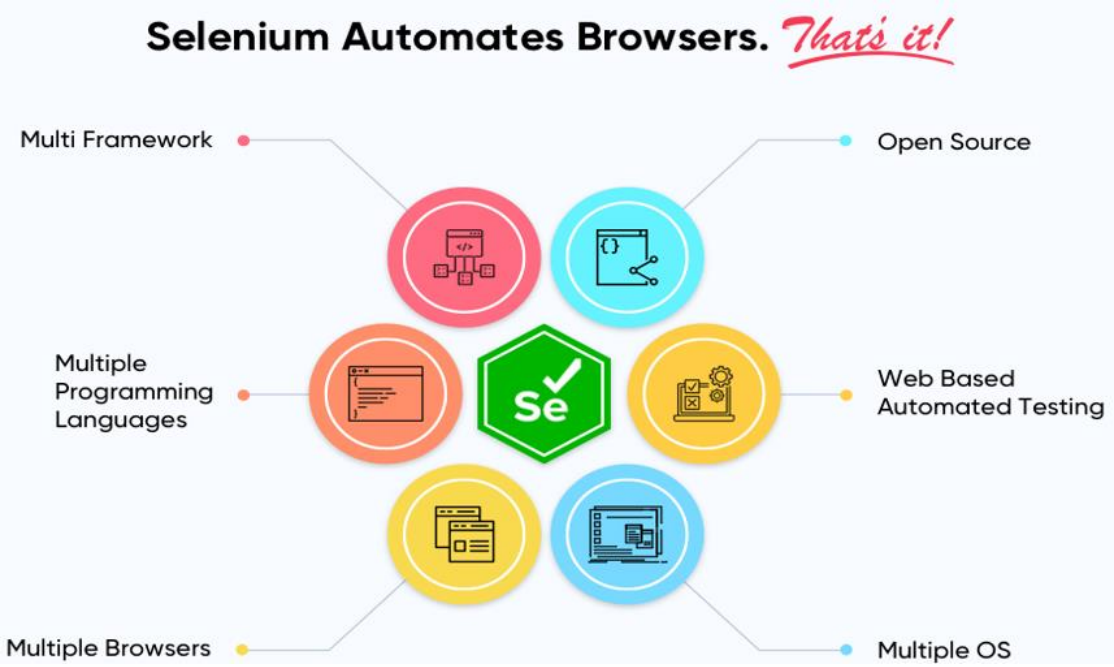
2. Introduction

This past year, we reviewed various automation solutions with Blaise 4.8 and 5, including Selenium, Winium, and Ranorex, and looked at the innovations coming out of Statistics Netherlands this year. Each of the products is discussed below.

2.1 Selenium

It is clear from Selenium's tagline that it is a testing tool for automating web application testing. When it comes to web automation testing tools, Selenium is one of the best. It is an outstanding open-source automation testing tool that can be executed in multiple browsers and operating systems, supporting a considerable amount of programming languages. It is the base for most of the other software testing tools. We are using Selenium WebDriver for automating websites and web surveys. Our Selenium script checks text validations (comparing actual text with expected text), image validations, buttons enabled, web elements displayed, hyperlinks, colors, and bolding of text.

Figure 1. Selenium Automates Browsers



2.2 Winium

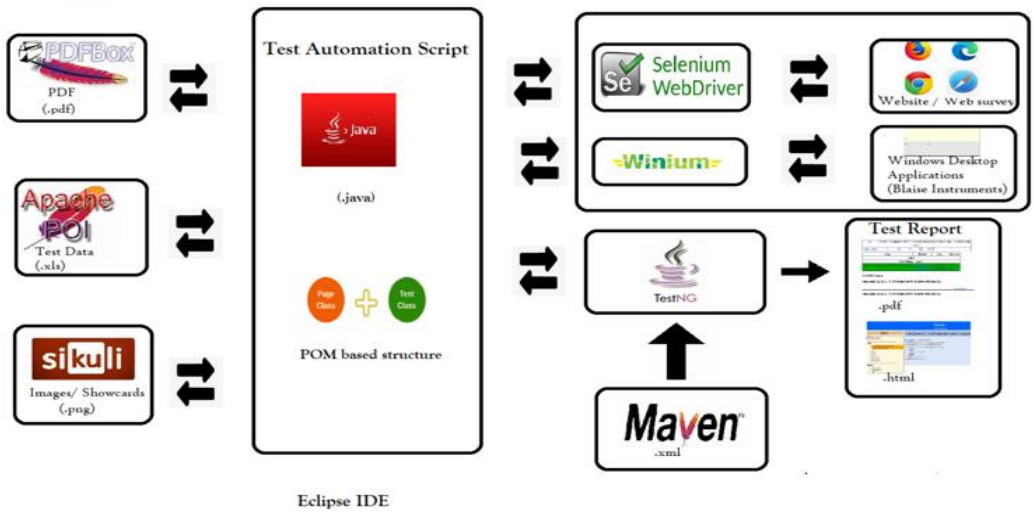
When it comes to testing and automating desktop applications on Windows, Winium is the ideal option. Winium—built on Selenium—is an open-source automation framework used for interacting with Windows applications. This framework can automate any desktop application developed on Windows Presentation Foundation or on Winforms.

Figure 2. Winium

Winium works very well for automating both CAPI and ACASI Blaise surveys. We have created automation scripts for surveys in Blaise 5.13. These scripts run in under five minutes, testing the same task that manually could take 1–2 days to test. Our script checks text validations, image validations, routing, skips, and e-signature entry.

Figure 3 shows the test automation framework.

Figure 3. Test Automation Framework



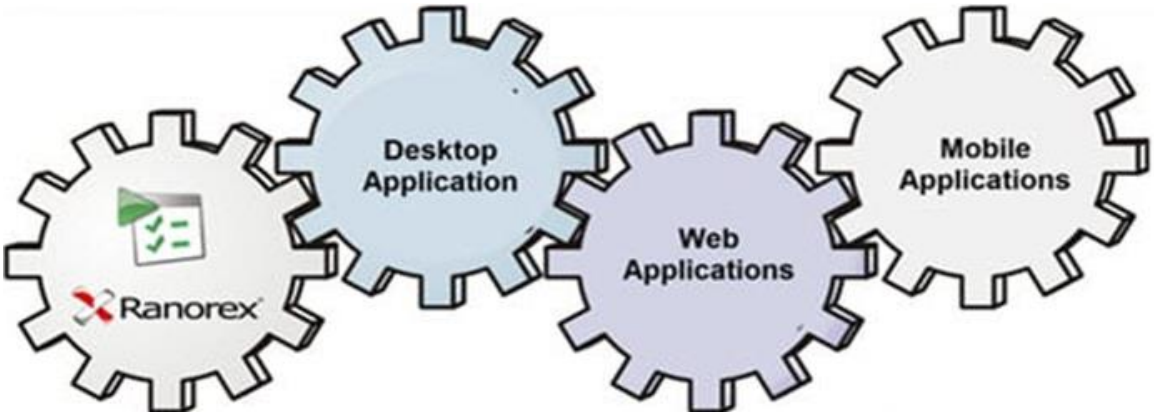
Java is the programming language used to write the test scripts. We use the TestNG Framework for managing test cases and generating detailed test reports, and we store the project in GitLab for version control. We also use a number of integration tools, for example:

- Sikuli: validates images, such as show cards
- Apache POI: reads data from Excel files, like preloads values
- Opencsv: reads data from CSV files (data files)
- Apache PDFBox: validates PDF text, such as consent forms

2.3 Ranorex

Ranorex Studio is a very powerful tool to automate tests for web, desktop, and mobile applications. It is less programming intensive, with a record/playback structure, so even noncoders can begin to create tests. It supports all technologies (.Net, Java, Flex, HTML) and works in different browsers (IE, Chrome, Firefox) and mobile applications (Android, iOS).

Figure 4. Ranorex



We use Ranorex to test the same validations as Selenium and Winium (text, image, routing validations, etc.) but in ACASI.

Figure 5. Pros and Cons of Different Blaise Automation Solutions

	Test Applications	Cost	Time	Coding Skills
Selenium	Web	Free	Set up + Scripting	Advanced skills
Winium	Desktop	Free	Set up + Scripting	Advanced skills
Ranorex	Web, desktop, mobile	$$$	Scripting	Minimum skills

Comparing our three automation solutions, Selenium gave us the least resistance to identifying Blaise objects when dedicating a Java-experienced test resource, but it's a web-only tool, so let's look at other considerations.

2.3.1 Cost

The open-source tools make it hard to justify the cost of a purchased testing automation tool and measure return on investment if you have short-term goals.

2.3.2 Time

Automated testing is essentially writing code to test other code. Added to the cost is the time spent setting up and maintaining the framework. Sometimes it will result in having to spend more time writing code than actually testing. Having a stable system is imperative; otherwise, resources constantly spend time maintaining test scripts instead of testing. The time payback comes with running automated scripts on a long-term regression system when tests can run in minutes instead of days. Now multiply that by the ability to run tests overnight while your manual testers are sleeping, and you can really reap the benefits of automation.

2.3.3 Skill

A good automation tester needs either basic programming skills and experience or time and aptitude for online learning. Even “record and playback” tools require script adjustments, and scripts need maintenance if something changes in the system under test.

2.4 Future Generation of Blaise Automation

Statistics Netherlands programmers are working on an automation solution that could replace, or at least enhance, our current automation tool(s). The Test Records Generation (TRG) provides a record of test cases that will account for navigating each questionnaire statement at least once. The TRG test record values can then be used to get the minimum number of test cases that exercise all statements without having to run every, or even duplicate, routing tests. A test log file will contain errors found while implementing the test records in a Blaise engine, including missing expected values and other test record values. Blaise 5.14 will introduce a separate test tool that offers TRG, and other solutions are upcoming.

2.5 Summary

We found a combination of tools suited our needs and settled on Selenium for websites and web surveys and Ranorex for Blaise 4.8 and 5.13 CAPI and ACASI surveys. The biggest payback on automation resources comes when you have scripts that can run overnight on a stable system. We use a premium Ranorex license for creating and maintaining test scripts and runtime licenses for running the test scripts.

When working on a stable system, testers and management will see benefits from automation testing over manual testing in these expected ways:

2.5.1 Fast and Efficient Testing

Automation testing runs tests much faster and more efficiently than manual testing. A large number of tests running in a shorter amount of time allows for faster detection of issues.

2.5.2 Consistent and Reliable

Automation testing eliminates human error and executes the same tests every time, improving the accuracy and consistency of the test results.

2.5.3 Cost-Effective

Although there is an initial investment in developing automated test scripts, if you have a stable system, it is more cost-effective in the long run than manual testing. The repetition of running automated tests without additional costs ensures the cost per test decreases over time.

2.5.4 Better Regression Testing

Automation testing can quickly identify regression issues and detect whether new changes have affected existing functionalities, thus reducing the risk of introducing new bugs or issues.

2.5.5 Increased Test Coverage

Manual testing puts limits on how many tests you can verify. Automation allows you to spend time writing new tests and adding them to your automated test suite. This increases the test coverage and reduces the risk of defects.

2.5.6 Scalable

Project requirements can determine the number of users, modules under test, executions, and test cases, making it an ideal solution for projects with stable systems.