CASE STUDY

End-to-End Test Automation Solution Using Selenium Tool

STAG implemented an end-to-end Selenium test automation solution through the CI platform, enabling parallel execution of tests on different browsers and platforms for a global provider of simulation technology and engineering services, to effect a 95% reduction in regression test cycle time.

Domain / Category - Engineering Services

Technology - Google Web Toolkit, Windows 7, Postgre, Firefox, Chrome, IE8 and 9, Safari, Selenium, and Jenkins

CUSTOMER AND PRODUCT BACKGROUND

The customer is a global provider of simulation technology and engineering services with a 27-year track record of high-end software for engineering and computing, enterprise analytics solutions, and innovative product design and development.

The product being tested, a core component of the customer's flagship product based on the SOA platform is a web-based enterprise simulation platform. The product was built on Google Web Toolkit and had some interactions, like save as, open dialog boxes, etc. The product has been in the market for the last six years and has been used by many prestigious customers across the globe.

PROBLEM STATEMENT

While trying to resolve certain performance issues of the product the customer had to change the code frequently, and every build required a thorough regression testing cycle spanning 2 person weeks. The customer's primary concern was to ensure that users of their product were not inconvenienced by performance issues.

SOLUTION

The STAG team initiated a knowledge transfer plan for rapid understanding of the product and the test assets to set up a good baseline for automation. Selenium was chosen as the tool for automation. A 3-member team then prepared an automation architecture, which was keyword data driven, that was not only scalable and expandable but also compatible with the two other core applications of the flagship product.

The STAG team improved existing test cases for completeness and adequacy, and also removed those that were redundant. To ensure better organization and maintenance, the team categorized the test cases into different levels. The team also developed automation libraries and scripts using Java coding standards. The scripts developed were able to run with multiple URLs on multiple machines across multiple browsers, all in parallel.

The team used Selenium Webdriver with TestNG toolsets to develop and manage the automation scripts and also execute them simultaneously. It also configured Selenium scripts with the Jenkins – Continues Integration (CI) tool for scheduled automation execution and initiating execution on post-build release. Selenium's inherent limitation of supporting only browsers was resolved using smart workarounds. To handle dynamic UI controls, the team adopted dynamic XPaths.

- # Automated test scenarios: 355
- # Module level scripts: 5 (each 3500 LOC)
- # Reusable Generic Library files: 3 (5100 LOC)
- # Driver scripts: 5 (each 900 LOC)
- # Automation document files: 2
- # Lines of code (LOC) of automation script: 27,000 approx.

The STAG team automated 5 modules and designed a HTML execution report with a dashboard to highlight the test case execution time stamp. The team also developed a risk and mitigation plan and performed 2 cycles of testing monthly.

OUTCOME AND VALUE ADDITIONS

The team developed a robust automation framework that allowed parallel execution with multiple URLs on multiple machines across multiple browsers. This resulted in reduction in manual testing time by 95% (from 100 hours to 4 hours). The team also put in place well-organized test assets that allowed easy maintenance and scalability on demand.