



Regression Testing with Selenium



NLIT Summit, May 2017

Lucille Forster, lforste@sandia.gov

John Beare, jpbear@sandia.gov



Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Agenda

- Who Are We?
- Problems from Inadequate Testing
- Selenium for Automated Testing
- Framework for Automated Testing
- Lessons Learned
- Resources
- Q&A: Closing Discussion

Who are we?

- Role
- Background
- Experience




Problems from Inadequate Testing

1. \$59.5 **billion** lost per year due to software defects. Source: 2002 federal study.
2. \$100 **billion** lost annually in U.S. due to cybercrime, global total is \$575 billion. Source: 2014 McAfee report.
3. 43% of software projects cost more, take longer, or do less; 18% of software projects fail. Source: 2013 Chaos Manifesto

Links to Sources:

1. <https://www.nist.gov/sites/default/files/documents/director/planning/report02-3.pdf>
2. <https://www.csis.org/events/2014-mcafee-report-global-cost-cybercrime>
3. <https://larlet.fr/static/david/stream/ChaosManifesto2013.pdf>

Selenium Website



The screenshot shows the SeleniumHQ website header. It includes the SeleniumHQ logo with the text 'SeleniumHQ Browser Automation'. To the right is a search bar with the text 'edit this page' and 'search selenium:'. Below the search bar are navigation buttons for 'Projects', 'Download', 'Documentation', 'Support', and 'About'.

What is Selenium?

Selenium automates browsers. That's it! What you do with that power is entirely up to you. Primarily, it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) also be automated as well.

Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser. It is also the core technology in countless other browser automation tools, APIs and frameworks.

Which part of Selenium is appropriate for me?

Selenium WebDriver



If you want to

- create robust, browser-based regression automation suites and tests
- scale and distribute scripts across many environments

Then you want to use [Selenium WebDriver](#); a collection of language specific bindings to drive a browser -- the way it is meant to be driven.

Selenium WebDriver is the successor of [Selenium Remote Control](#) which has been officially deprecated. The Selenium Server (used by both WebDriver and Remote Control) now also includes built-in grid capabilities.

Selenium IDE



If you want to

- create quick bug reproduction scripts
- create scripts to aid in automation-aided exploratory testing

Then you want to use [Selenium IDE](#); a Firefox add-on that will do simple record-and-playback of interactions with the browser.



Selenium is a suite of tools to automate web browsers across many platforms.

Selenium...

- runs in [many browsers](#) and [operating systems](#)
- can be controlled by many [programming languages](#) and [testing frameworks](#).



Donate to Selenium

with PayPal

[Donate](#)



through sponsorship

You can [sponsor the Selenium project](#) if you'd like some public

Selenium Website: <http://docs.seleniumhq.org>

Selenium for Automated Testing

- Selenium is a great open source tool for Web UI testing with a large community that supports it, including Google
- Supports C, C#, Java, Python and Ruby languages
- Support IE, FireFox and Chrome browsers
- Selenium provides support for integration of open source frameworks like TestNG, JUnit, and NUnit
 - We use NUnit for our framework
 - We've done some additional testing with JUnit
- Large community of users
- IDE for recording in Firefox

Steps for Using Selenium IDE

1. Download and install software from website.
2. Add plug-in to Firefox.
3. Open Firefox and open website to be tested.
4. Select Selenium icon from Firefox toolbar (screen shot on next slide).
5. Selenium IDE launches in recording mode (screen shot two slides ahead).
6. Stop recording and save using buttons in IDE (screen shot two slides ahead).
7. Replay Selenium script using buttons in IDE (screen shot two slides ahead).

Firefox: SNL External Webpage



Sandia National Laboratories
Exceptional service in the national interest

ABOUT PROGRAMS RESEARCH WORKING WITH SANDIA NEWS CAREERS

Latest News [Predicting the limits of friction: Sandia looks at properties of material](#)

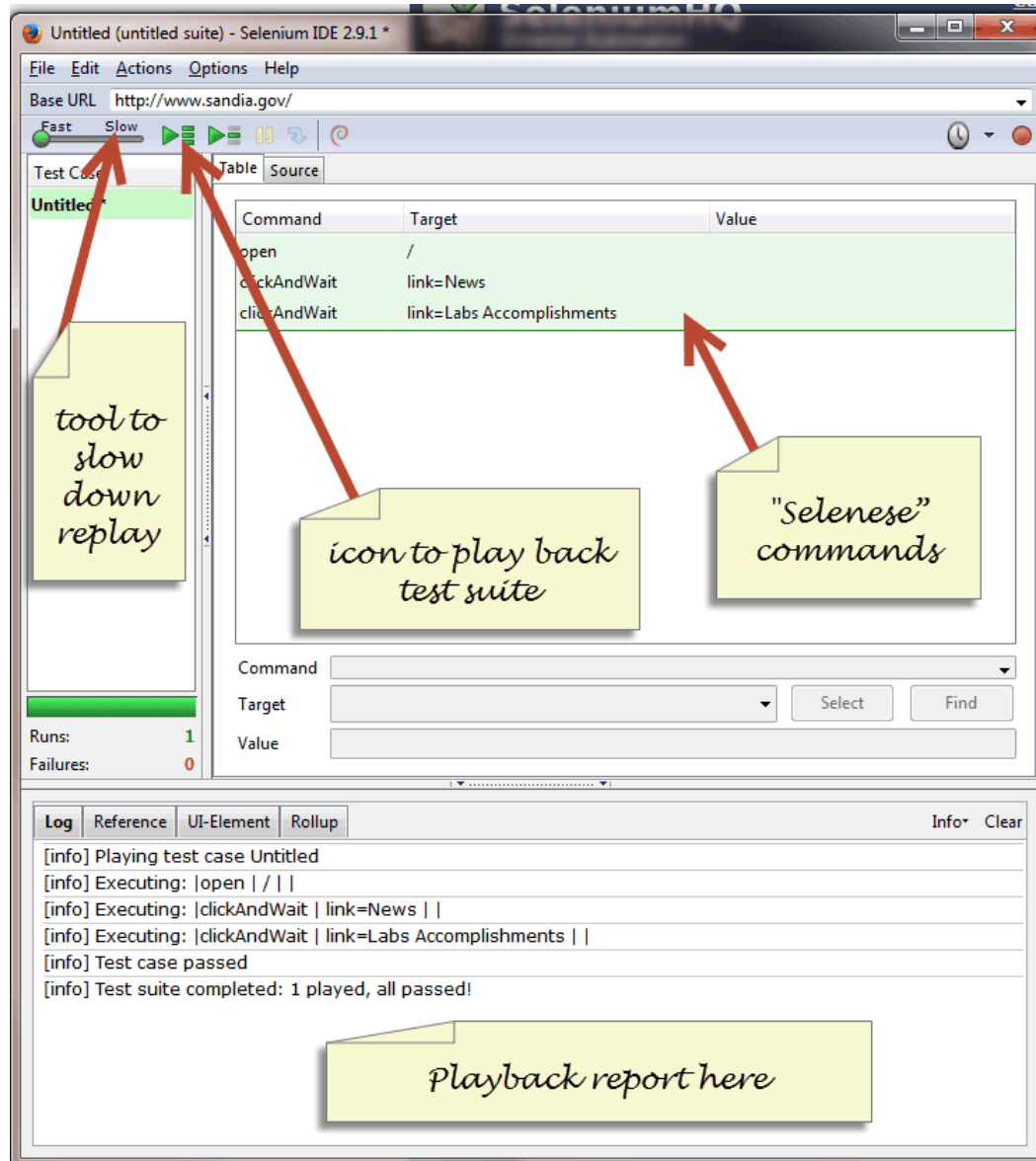
Follow Us

NATIONAL PRIORITIES

- NUCLEAR WEAPONS
- ENERGY
- NONPROLIFERATION
- DEFENSE

Selenium icon in Firefox

Selenium IDE invoked from Firefox



The screenshot shows the Selenium IDE 2.9.1 interface. The top toolbar includes a speed control slider (Fast/Slow) and a play button. The main area contains a table of test commands:

Command	Target	Value
open	/	
clickAndWait	link=News	
clickAndWait	link=Labs Accomplishments	

Annotations with arrows point to the speed control slider (labeled "tool to slow down replay"), the play button (labeled "icon to play back test suite"), and the "clickAndWait" commands (labeled "'Selenese' commands").

The bottom section shows a log of the test execution:

```
[info] Playing test case Untitled
[info] Executing: |open | / | |
[info] Executing: |clickAndWait | link=News | |
[info] Executing: |clickAndWait | link=Labs Accomplishments | |
[info] Test case passed
[info] Test suite completed: 1 played, all passed!
```

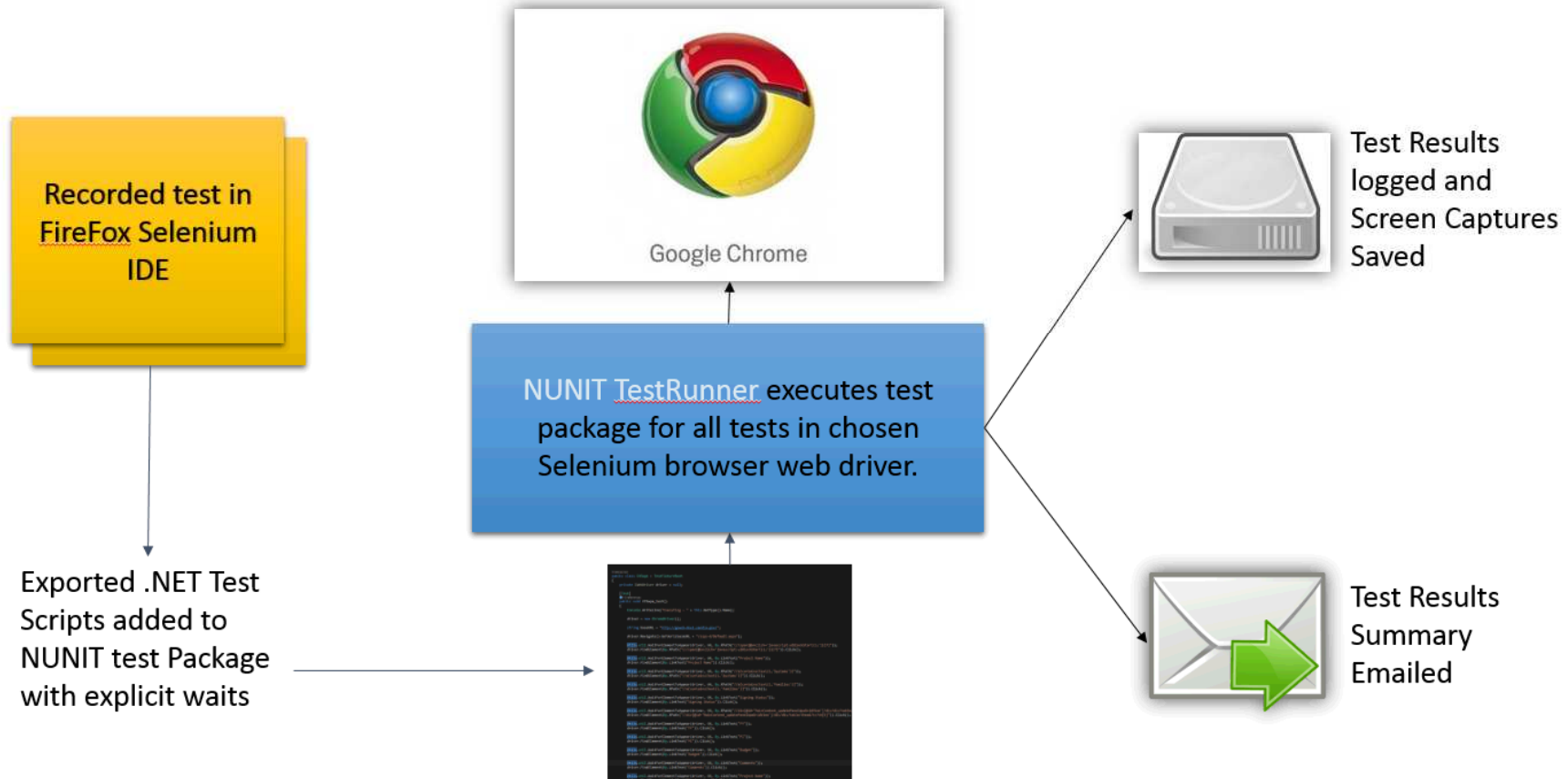
An annotation labeled "Playback report here" points to the log output.

Framework for Automated Testing

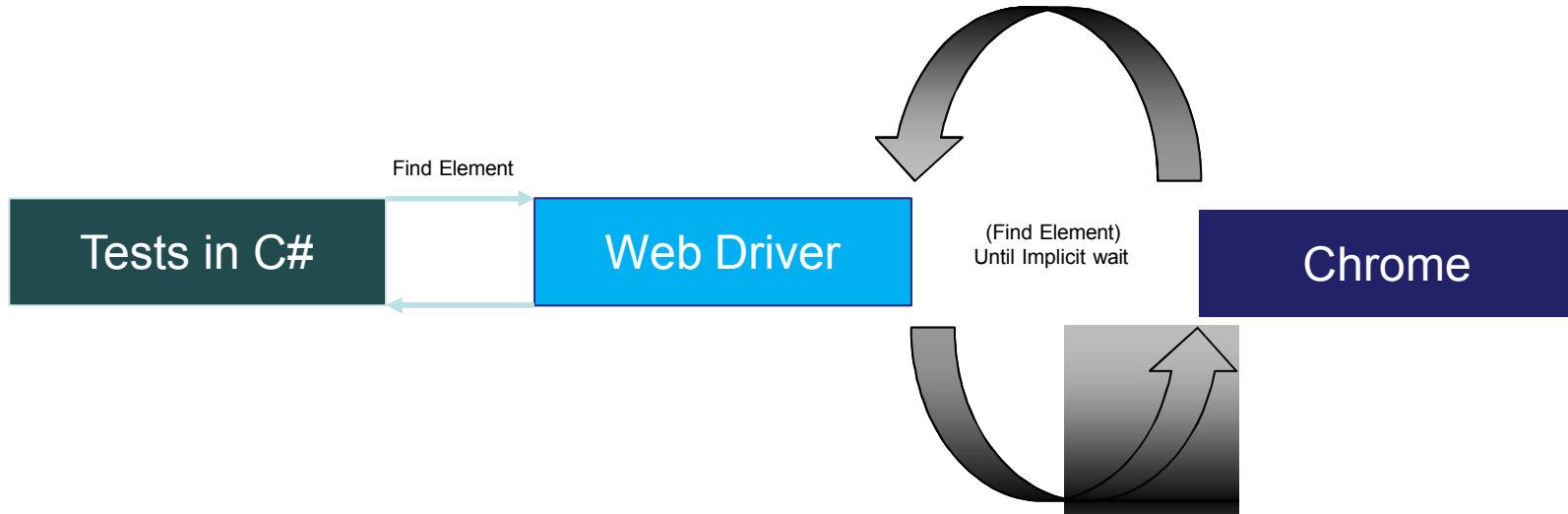
- The team developed a framework to utilize Selenium's IDE and export the tests as C# to NUnit for testing.
- The framework runs the tests nightly and emails the team if any tests failed.
- The framework produces a log of the errors found, along with a screenshot of where the error originated from.

Framework for Automated Testing

SNL NUNIT SYSTEM



Framework for Automated Testing



Lessons Learned

- Browser drivers are constantly being updated which causes us to have to update our framework
- Chrome works best for our testing
- We've had to add wait times in our framework when running the tests because elements would be delayed when getting displayed on the page. Network bandwidth plays role in delays
- JUnit has more drivers, including headless drivers
 - Authentication can be a problem with automated testing.

Lessons Learned (cont.)

- Long, manual tests need to be broken up for automation
- Automated tests that are dependent on element names or values that change are likely to cause the tests to fail.
- Selenium often fails to properly export to NUnit with tabbed applications
- If applications have role-based security, tests need to be intentionally designed to use different users or change user permissions.

Resources

- Selenium website: <http://docs.seleniumhq.org/>
- Selenium documentation: <http://docs.seleniumhq.org/docs/>
- Selenese commands (Selenium scripting language):
http://docs.seleniumhq.org/docs/02_selenium_ide.jsp#selenium-commands-selenese
- Books & classes: search engine, Amazon, etc with term “Selenium programming” or “Selenium Automated Testing”

Closing Discussion, Q&A

- Discussion
 - What lessons can you share?
 - What tools and practices work for you?
 - What are your issues?
- Questions?

