Innersource in Test Automation

James Knowlton

jknowlton@navexglobal.com

Abstract

In implementing test automation at your organization…does any of this sound familiar?

- Organizational silos, each implementing their own automation framework and processes (or not implementing automation at all)
- “Cathedral”-based automation framework, developed by one or two engineers, understood only by them
- Highly customized framework, which requires in-house training and support

This is the situation we encountered when I was hired as QA Architect at NAVEX Global. NAVEX is an enterprise ethics and compliance software vendor based in Lake Oswego, OR. They were assembled as a company via a series of acquisitions, so they are geographically diverse, with a wide disparity in technical and development ability/experience. As will be discussed, we had an automation framework build by one person (who then left the company) which was highly customized. This made it difficult to train and support QA Engineers, and also to implement improvements.

How we addressed the challenge of implementing a common automation framework is the subject of this paper. The paper will also discuss the results we were able to produce, and next steps in our process.

Biography

James Knowlton is a QA Architect at NAVEX Global, where he oversees quality assurance processes including defining standards, methodologies, procedures and metrics used across NAVEX Global's engineering organization. His previous experience includes stints at Symantec, McAfee, ADP and eBay. He has presented on software quality at multiple conferences and is the author of the book "Python: Create, Modify, Reuse" (Wrox, 2008).

1 Introduction

1.1 Description of NAVEX Global

NAVEX Global is a developer of cloud-based ethics and compliance software. They offer hotline reporting, case management software, policy compliance software, vendor management software, online compliance training, and other tools. The company headquarters are in Lake Oswego, Oregon, with several offices in the US and an office in London, England.

1.1.1 History of NAVEX

- July 2012: The Riverside Group, a private equity firm, purchases EthicsPoint and merges it with Global Compliance (a Charlotte, North Carolina based competitor). The new company will be called NAVEX Global.
- June 2012: NAVEX Global acquires PolicyTech, a Rexburg, Idaho based policy compliance software provider.
- October 2014: Vista Equity Partners acquires NAVEX Global from the Riverside Group.
- August 2015: NAVEX Global acquires The Network, an Atlanta, Georgia based competitor, doubling the size of the company (to about 1000 employees).

Bottom line: there's been a lot of churn.

1.2 Overview of the QA team

NAVEX had introduced a company-wide rollout of Scrum/agile about a year before I arrived. It had varying levels of adoption and support. One of the challenges generally was that since the company had been cobbled together by combining several different companies, there were significant variations in culture and experience.

1.2.1 Engineering team structure

At NAVEX, the engineering teams (about 10-20 engineers per team) were vertically aligned, with the following:

- Product owner - the engineering leader (usually a manager/director) who is responsible for the overall success of the product from an engineering perspective. Some of the product owners came from an engineering background, but some actually came from project management, so they lacked the engineering background to provide guidance, especially concerning QA processes.
- DevOps engineers – this position existed for some teams. When present, the DevOps engineers managed the engineering infrastructure, including the build system and QA servers.
- Development engineers
- QA engineers

1.2.2 QA teams' knowledge/experience

There was significant variation in knowledge and experience concerning QA engineers. Some of the teams had a good mix of very experienced QA engineers with some engineers who were less experienced in testing but who had significant coding/development expertise. However, other teams had QA engineers who knew the product but were lacking not only coding knowledge (for automation), but even general technical expertise. The fact that teams were geographically dispersed made training these people difficult.

2 Challenges

2.1 Prior automation framework

2.1.1 Overview of pre-existing automation

I did not actually work with the pre-existing framework, but my understanding is that there were many limiting factors. It was developed by a single QA Engineer. It used significant customization to try to make it as easy as possible for people to write tests ("push button automation"). It was used in some teams, but other teams did their own thing for automation (for example, one team had a framework based on Java/Selenium).

2.1.2 Problems with pre-existing framework

- Because only one person maintained the framework:
    - Other engineers were prevented from contributing, thus preventing them from enhancing their morale and sense of empowerment.
    - There was a bottleneck when it came to suggesting or implementing improvements.
- Because the framework was highly customized:
    - The framework was very brittle, and not very adaptable to unforeseen conditions.
    - There were no public training materials available. This was especially problematic due to the geographic dispersal of the company and the lack of experience of some engineers (see 1.2.2 above).
- The framework was developed in Ruby and Watir (a Ruby-based web automation framework). Since Ruby wasn't actually used anywhere else in the company, there was no help available from development engineers.
- The one person who developed it…quit. The company was left with no one to maintain the framework.

2.2 Review of the challenge

We had a QA organization that:

- Was dispersed all over the company (making training more of a challenge)
- Had significant variation in the level of QA and coding knowledge/experience
- Had management (sometimes) that was willing but not able to provide experienced technical direction

We had a framework that was:

- In a language not widely used at the company
- Not used or even looked at by developers (due to being in Ruby and frequently being broken)
- Not understood enough to change or improve
- Not widely used, so public training was not available
- Brittle and prone to breaking

3 What is Innersource?

3.1 Definition

Danese Cooper of PayPal defines Innersource as “The application of best practices, processes, culture and methodologies taken from the open source world and applied to internal software development and innovation efforts”.

In other words, it consists of using the benefits of open source (collaboration, collective code ownership, documentation of code reviews and checkins) in the context of an organization’s internal projects.

3.2 Open source principles

- Engineers share their work with a wide audience, instead of just with a manager or team
- Everyone can contribute to the project
- Distributed version control is used (such as GitHub) so that collaboration and code review can happen more easily
- New code repositories (branches) can be made freely
- People at large geographical distances, at separate times, can work on the same code or contribute different files of code to the same project.
- Communication tends to be written and posted to public sites instead of shared informally by word of mouth.

3.3 Applied internally at an organization

- Innersource differs from classic open source by remaining within the view and control of a single organization.
- The “openness” of the project extends across many teams within the organization. Ideally, code is shared freely within the entire organization.

3.4 Benefits of Innersource

- Code reuse across the organization grows immensely
- Cross-team collaboration becomes relatively frictionless
- Written comments exchanged among team members, although taking up some time, more than pay for themselves by helping new engineers learn the system faster.
- Engineers learn to document their code better, both formally (as in-code comments and documentation) and informally (on discussion lists and chat channels).

4 How we applied Innersource to the NAVEX Global challenge

4.1 Project Overview – NOVA Framework

4.1.1 What is it?

The NOVA Framework is a test automation framework for UI-based testing of web applications. It is Selenium-based and makes use of Page Object Model, a Selenium design pattern designed to encourage code maintainability.

Nova is a word form which (in various languages and spellings) means “new”. So, we felt it would be a good name for our new framework.

It makes use of C#/.NET, Selenium, and NUnit.

4.1.2 Who created it?

It was initially developed by a partnership of development and QA engineers on the Ethicspoint Incident Manager product, after watching a training video on Page Object Model. After I was brought in as QA Architect, I designed the Innersource process to develop and improve NOVA companywide.

4.2 NOVA framework architecture

Component	Owned by *	Composed of	Use
Tests	Product team	Ummm…tests	Implementation of NOVA framework/product framework
Product framework	Product team	• Page objects for application under test • Helpers/code specific to application	Inherits from (and makes use of) NOVA framework
NOVA framework	Engineering org	• Common objects and helper methods • Product-independent	Foundation layer

* Development & QA

The intention is that just as with an open source project, individual product teams can code their own solutions in the product framework as they see fit, and if they have something they think can be generally useful, they can submit it for consideration in the NOVA framework

4.3 NOVA project team organization

Role	Membership	Details
Architect/Project Lead	QA Architect	• Provide overall direction and leadership • Chair of NOVA steering committee
Steering Committee	• 3-4 people • Dev & QA • Multiple products	• Plan and direct NOVA releases • Groom NOVA backlog regularly • Communicate with engineers on NOVA tasks assigned
Engineering cross-product team	Engineering org	• Anyone interested in contributing • Sign up for stories per sprint, based on interest and availability

4.3.1 The Steering Committee

The steering committee is represented by as many cross sections as possible in the engineering organization. For example, there were both dev & QA and also multiple products (and geographical sites) represented.

4.4 NOVA schedule

NOVA was implemented and maintained in a fairly typical Scrum schedule. One difference is the team did not engage in story pointing or specific estimating. They basically just signed up for stories until it felt like a good amount of work for the duration of the sprint. If we ran out of time for a story, it just got moved to the next sprint.

4.4.1 Releases

- Every six weeks
- In sync with product release schedule
- Release is comprised of two 3-week sprints (also in sync with product schedule)
- Planning day is held on a Wednesday (to prevent conflict with product planning, which is on Mondays)

4.4.2 Sprint planning

- Everyone interested is invited
- We do the following:
    - Complete current sprint, including reviewing work completed and moving in-progress work to the next sprint
    - Pull tasks from the backlog until we have what we feel is a good amount of work for three weeks
- It's "low-maintenance" planning (no time estimates, story points, etc.)
- When someone signs up for a story/task, they clone it to their product's JIRA project
    - This is so the product team can track their work
    - When they complete the task/story, they close it in both projects

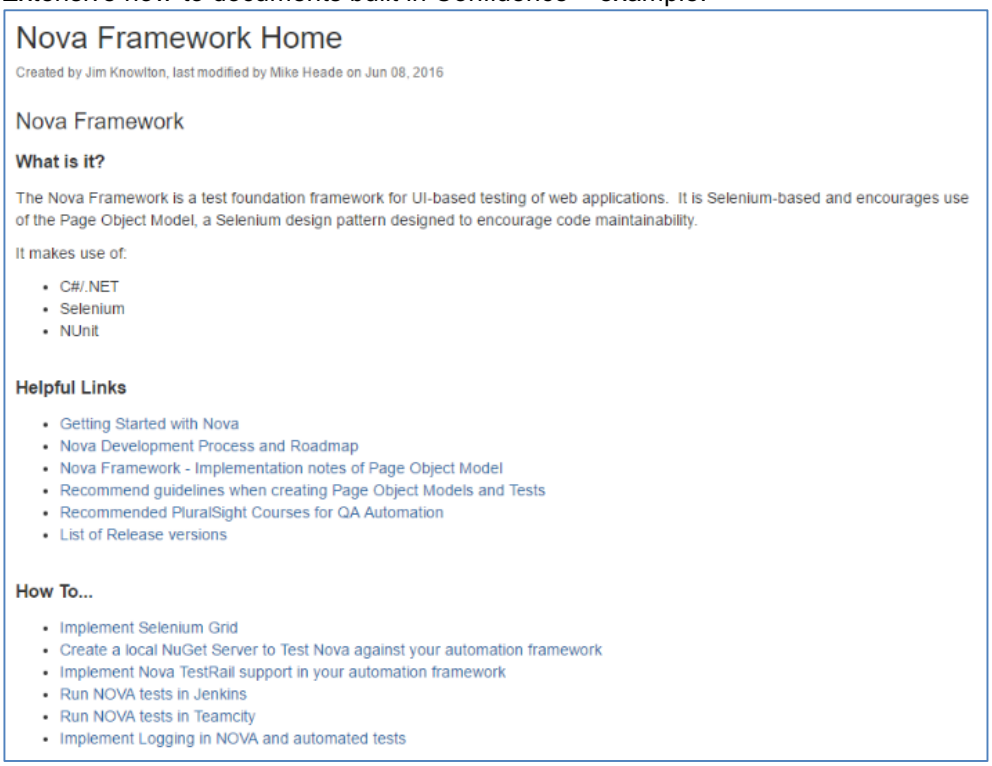
4.4.3 Backlog grooming

- Responsibility of the steering committee
- Performed weekly

4.5 NOVA training/documentation

4.5.1 Documentation

- Extensive how-to documents built in Confluence – example:



Nova Framework Home

Created by Jim Knowlton, last modified by Mike Heade on Jun 08, 2016

Nova Framework

What is it?

The Nova Framework is a test foundation framework for UI-based testing of web applications. It is Selenium-based and encourages use of the Page Object Model, a Selenium design pattern designed to encourage code maintainability.

It makes use of:

- C#/.NET
- Selenium
- NUnit

Helpful Links

- Getting Started with Nova
- Nova Development Process and Roadmap
- Nova Framework - Implementation notes of Page Object Model
- Recommend guidelines when creating Page Object Models and Tests
- Recommended PluralSight Courses for QA Automation
- List of Release versions

How To...

- Implement Selenium Grid
- Create a local NuGet Server to Test Nova against your automation framework
- Implement Nova TestRail support in your automation framework
- Run NOVA tests in Jenkins
- Run NOVA tests in Teamcity
- Implement Logging in NOVA and automated tests

- API documentation automatically generated (via Doxygen) with every build

5 Benefits and Lessons Learned

5.1 Benefits

We saw many benefits to our implementation:

- Collective code ownership spread knowledge (and accountability) across the organization
- Using a standard framework/design pattern (Selenium, Page Object) allowed us to more easily build and collaborate across the organization.
- This also made training much easier, as there are significant publicly available training materials on Selenium and Page Object Model.

5.2 Lessons learned

We learned several things as we used the open source model to implement our automation:

- We found constant collaboration to be critical both in troubleshooting issues and training people on the framework. As NAVEX uses Slack, we built Slack channels specifically for NOVA collaboration. This was a great help, especially as Slack has great integrations for GitHub.
- As each team was different in its ability to jump into implementing NOVA, we were very flexible in terms of implementation schedule. This allowed teams to plan for training, transitions from other frameworks, etc.

- One thing we intentionally planned for from the beginning is that we wanted the framework to be lightweight: the less there was in it, the less to go wrong. We communicated from the beginning that if a product team wanted some functionality that wasn't in NOVA, and we didn't feel we should add it, they were always welcome to add whatever functionality they wanted to their own product framework.

6 Conclusion

Although not a "one size fits all" solution, our experience was that Innersource can be a very effective way to collaboratively manage development of a test framework inside an organization.

References

Web Sites:

1. Portland Business Journal. "EthicsPoint acquired, to merge with rivals". http://www.bizjournals.com/portland/news/2012/02/01/ethicspoint-acquired-to-merge-with.html (accessed June 20, 2016)

2. Oregonlive. "EthicsPoint changes its name to NAVEX Global, buys Idaho-based PolicyTech".http://www.oregonlive.com/silicon-forest/index.ssf/2012/06/ethicspoint_changes_its_name_t.html (accessed June 20, 2016)

3. Danese Cooper, PayPal, "Getting Started in Innersource", (https://www.youtube.com/watch?v=r4QU1WJn9f8), 2015.