Enterprise Test Automation: The Key to Agile QA Success
Agenda

- **Part I – Definitions**
  - Enterprise Framework
  - Dispel some myths
  - Understand the benefits
  - Defining Agile

- **Part II – Gathering Requirements**
  - Important questions
  - Filling up the backlog
  - Deciding on your technology stack

- **Part III – Foster the People**
  - The shift to engineering
  - Paired testing
  - Innovating

- **Part IV – Learning to Love the Process**
  - Governance
  - POCs
  - Metrics

- **Part V – Creating the Vision**
  - Objectives
  - Strategies & Tactics
  - Roadmap

- **Part VI – Implementation**
  - Be Agile
  - Refactor often
  - Focus on the “core”

- **Wrap Up**
What’s in a Name?
Automation Framework

- The basic structure underlining the system (NUnit, TestNG, MSTest)
- The engine allows you to run the car (GitHub, TFS, Maven, Jenkins)
- The steering wheel navigates through the UI (Selenium, UFT, TestComplete)
- The tires keep you safely on the road (Governance & best practices)
- A driver manages the execution (defect management, refactoring)
- You need fuel to run the car (Test Data Management)
- Your dashboard tells you the health of your car (KPIs)
- Once you’ve built it, you can customize it with “upgrades”
Enterprise Framework

Project Team X

Project Team Y
Dispelling Myths

What are some of the myths you hear?

- It’s too expensive to implement (ROI)
- It’s too hard to maintain
- You have to be a developer to make it work
- Our system is too complicated to automate
- We can’t do in-sprint automation, we focus on regression
- Automation slows our velocity, it’s more important to focus on completing the Sprint
Benefits of an Enterprise Framework

Increased Testing Effectiveness:
- Increased Standardization
- Increased Testing Consistency
- Enables Total Quality
- Reduction in Human Error

Increased Testing Efficiency:
- Accelerated Test Execution (Scripting Uptime)
- Increased Integrated Test Coverage (Eliminate Duplication Across Products)
- Involvement of Business Upfront (BI)

Increased Testing Throughput

Better Agile Support:
- Quicker Feedback Loops
- Earlier Testing
- More Complete Testing
Ancillary Benefits

**Reduction of Testing Cost:**
- Test Execution
- Maintenance
- Early Detection of Defects
- Workforce (Mobilize an Integrated Team)

**Ancillary Benefits:**
- Integration of other technology solutions like Performance Testing
- Never starting over and quickly reinventing solutions
- Leveraging scripts across products (eliminate duplicated work)
- Having a Basis of Estimates (Historical context for decisions)

Increased Testing ROI

More Robust TQM
Achieving the Right Balance of Quality

Typical Practice

- Manual
- UI
- Integration
- Unit

Quality Control (Testing)
Reactive, late defect/risk/issue appraisal
Meets Requirements

Best Practice

- Manual
- UI
- Integration
- Unit

Quality Assurance (Process)
Proactive, early defect/risk/issue prevention
Fit for Use & Meets Requirements
What is Agile

Sourced from http://www.wirtek.com/service-delivery/
Gathering Requirements
“Important” Questions

- Does everyone need to be happy?
  NO – Being heard is more important than being happy

- Does the framework have to address everything?
  NO – Iterating and refactoring are your friends

- Does cost factor into our decisions about the framework?
  ABSOLUTELY – Making the right budget choices is key

- What should I be thinking about?
  A LOT OF THINGS! Come up with a questionnaire
Filling Up the Backlog

Your primary goal is to define the problem you’re trying to solve:

- Should I match the development stack?
- Do I need to consider the enterprise architecture?
- How complex is my software? How complex will it get?
- How do I want to handle Test Data Management?
- How can I handle CI?
- How mobile is our future?
- How maintainable should my framework be?
- Is my internal staff technical enough?
- How mature are our user stories?
Building Modularity

- **Core** – provides the reporting framework and logging facilities for the other modules
- **Web** – robust and reliable testing of web based applications in all common browsers
- **Mobile** – testing of mobile applications or mobile browsers
- **API** – REST or SOAP
- **BDD** – the BDD language layer can be added to any combination of modules
- **Database** – connects to any database
What is Core?
Building from the Core Out

Example .NET Core

- VSTS
- MSTEST
- ALLURE
- SELENIUM
- SQL SERVER
- AUT

What is absolutely necessary for your framework to run?

Example Java Core

- GIT
- JENKINS
- MAVEN
- TESTNG
- ALLURE
- SELENIUM
- SQL SERVER
- AUT

What pieces of the automation will be leveraged by every project?
Technology Stacks

**SOURCE CONTROL MANAGEMENT SYSTEMS (SCM)**
- GIT
- BITBUCKET
- VSTS

**CI TOOLS**
- JENKINS
- BAMBOO
- VSTS

**BUILD TOOLS**
- ANT
- MAVEN
- VSTS

**BDD/TDD TOOLS**
- CUCUMBER
- JBEHAVE
- FITNESSE

**TEST FRAMEWORKS**
- NUNIT
- TESTNG
- MSTEST

**TEST / DEFECT MANAGEMENT TOOLS**
- JIRA
- ALM
- MTM

**SERVICE ARCHITECTURE TOOLS**
- SOAPUI
- READYAPI
- RESTSHARP

**GUI AUTOMATION TOOLS**
- SELENIUM
- UFT
- WINIUM

**MOBILE AUTOMATION TOOLS**
- APPIUM
- SELENDROID

**APPLICATION**
- AUT
Deciding Your Stack

VSTS
Find flexibility in a tool especially those that cover multiple bases

JENKINS
Find something well supported and commonly used (only tackle something new if you have time or no other solution)

AUT
The enterprise architecture matters, so mirror it as much as possible

APPIUM
Keep an eye on the future

GRID
Find tools that enable you to be more efficient and always be mindful of purpose
Foster the People
Acknowledging the Shift

Confusion & Delay

Being evil for the greater good?
# Typical Agile QA Timeline

<table>
<thead>
<tr>
<th>Pre-Sprint</th>
<th>Sprint Planning (Day 1)</th>
<th>Test Prep &amp; Run (Day 2-4)</th>
<th>Test Execution (Day 5-7)</th>
<th>Sprint Completion (Day 8-10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Risk Recommendation</td>
<td>• Risk Acceptance</td>
<td>• Automation Scripting</td>
<td>• Smoke Testing</td>
<td>• Feature Testing</td>
</tr>
<tr>
<td>• Grooming</td>
<td>• Test Strategy</td>
<td>• Test Data</td>
<td>• Automated Testing</td>
<td>• End-to-end Testing</td>
</tr>
<tr>
<td></td>
<td>• Test Prep</td>
<td>• Feature Testing</td>
<td>• Feature Testing</td>
<td>• Performance Testing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functional/Regression Prioritization</td>
<td>• Functional/Regression Testing</td>
<td>• Defect Management</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Smoke Testing Execution (CI)</td>
<td>• Defect Management</td>
<td>• UAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Demo</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Sign-off (Sprint)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Retrospective</td>
</tr>
</tbody>
</table>
Paired Testing

Who:
- QA analyst – SME
- QA engineers – Technical solutions

What:
- Knowledge transfer
- Peer reviews
- Test case design sessions

Why:
- Allay fears and give purpose
- More complete technical solutions
- Better total quality
- Collaboration

How:
- One hour a day, every day
- Rotate the focus
- Keep a log

Best Practices:
- Rotate partners every other Sprint
- Communication should be dialogues
- Be customer-driven
- Think operationally
- Create a community of practice forum
- Encourage “natural” synergies
# Shared QA Responsibilities in Agile

## Quality Belongs to the Team

### Analyst (Any)
- Story
- Test Data
- Functional/Regression Prioritization
- Feature Testing
- Transparency, updates to sprint goals

### Shared
- Risk Recommendation
- Backlog Maintenance
- Risk Acceptance
- Test Strategy
- Definition of Ready
- Definition of Done
- Sizing
- Defect Management
- Commitment to Sprint goals
- End-to-end Testing
- QA Sign-off (Story)
- Demo
- Retrospective

### Engineer (Any)
- Automation Scripting
- Smoke Testing
- Automated Testing
- Functional/Regression Testing
- Performance Testing
- Alignment to architecture
Innovate to Motivate

Let’s talk about reality for a minute:
- QA can be a little boring
- QA can be a little tedious
- QA can be a little underappreciated
- QA can be a little frustrating

Engineers are different:
- Focused on creating
- Focused on experimenting
- Focused on finding solutions

Innovation is a great motivator:
- Feeds their “needs”
- Allows them to showcase cool things
- Keeps them relevant and skilled

When do engineers feel like their job is at its most “difficult”?

When everything has been built and the automation goes to maintenance mode.
Learning to Love the Process
Governance and Delivery

Project Team X
Architect (Oversight)
Senior Engineer X
Engineers

Project Team Y
Architect (Oversight)
Senior Engineer Y
Engineers
Roles & Responsibilities

Enterprise Team Roles:

- **Enterprise automation architect:**
  - Final decision maker
  - Visionary & strategist
  - Tools expert

- **Senior automation engineers:**
  - Day-to-day leaders
  - Stewards of standards (audits, process reviews, etc.)
  - Tackle “most complex” problems

- **Engineers:**
  - Project workers
  - Script writers & executors
  - Defect Resolution
  - Temporary?

Leadership Responsibilities:

- **Leadership (Singular Voice)**
  - Governance
  - Oversight
  - Advocacy

- **Technology Innovation**
  - Solution Architecting
  - PoCs
  - Program/Project Support

- **Community of Practice**
  - Researching
  - Training
  - Networking
  - Leading the Race
Proof of Concepts

**Best Practices:**

- Have a clear objective
- Know your requirements
- Have clear criteria for success
- Have a solution in mind
- Try something complex
- Time box the POC
- Build an MVP (limit your scope)
- Document everything... keep a running log
- Failure isn’t always bad
- Develop a pros and cons matrix
- Make recommendation
Everyone is typically worried about Cost of Quality simply from the perspective of dollars saved and while that is good, don’t forget to highlight how that money gets saved (governance, best practices, people, planning) and what it enables.
Quality throughput is just as important as Cost of Quality because if your enterprise automation framework doesn’t enable you to increase throughput then you haven’t been successful.
Making sure your enterprise automation framework enables “shift left” by your QA teams is also important to track. Keeping an eye on where defects are discovered is a great indicator of shift left.
Creating a Vision
Arriving at a Shared Vision

- Before you get to strategy or implementation, you need a shared vision, organizational goals for quality you can align your automate to. But getting to that shared vision can be difficult:
  - Everyone has their own priority
  - You have inflight work that you need to make sure doesn’t fall behind
  - You don’t have the right people, process, tools
- Have an independent voice lead the way
- Do it in two phases
- Be armed with facts
- Hear everyone out
- Work together to arrive at consensus
Let’s Get Tactical

As an organization, our Quality Objective is to decrease the cost of quality by 12% in FY2017. We will measure the success of this Objective by achieving at least a $1.2M savings in the cost of quality against the FY2016 budget.

<table>
<thead>
<tr>
<th>Tactic</th>
<th>Measurement</th>
<th>Owner</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Achieve x% automation of test cases for each project</td>
<td>Automation burn-down/up</td>
<td></td>
</tr>
<tr>
<td>2. Conduct Risk-based Testing</td>
<td>Schedule adherence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Field escapes</td>
<td></td>
</tr>
<tr>
<td>3. Document business workflows &amp; incorporate into testing</td>
<td>Field escapes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defect discovery accuracy</td>
<td></td>
</tr>
<tr>
<td>4. Start testing at the discovery phase</td>
<td>Defect discovery rate by phase</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Defect severity rate by phase</td>
<td></td>
</tr>
</tbody>
</table>
Roadmap

<table>
<thead>
<tr>
<th>Winter 2016</th>
<th>Spring 2017</th>
<th>Summer 2017</th>
<th>Fall 2017</th>
<th>Winter 2017</th>
<th>Spring 2018</th>
<th>Summer 2018</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objectives/Strategies</strong></td>
<td><strong>Objectives/Strategies</strong></td>
<td><strong>Objectives/Strategies</strong></td>
<td><strong>Objectives/Strategies</strong></td>
<td><strong>Objectives/Strategies</strong></td>
<td><strong>Objectives/Strategies</strong></td>
<td><strong>Objectives/Strategies</strong></td>
</tr>
<tr>
<td><em>Short Term (Highest Priority)</em></td>
<td><em>Medium Term</em></td>
<td><em>Long Term</em></td>
<td><em>Long Term</em></td>
<td><em>Long Term</em></td>
<td><em>Long Term</em></td>
<td><em>Long Term</em></td>
</tr>
<tr>
<td>3–6 months</td>
<td>6–12 months</td>
<td>12–18 months</td>
<td>12–18 months</td>
<td>12–18 months</td>
<td>12–18 months</td>
<td>12–18 months</td>
</tr>
<tr>
<td>- Emphasize a quality objective</td>
<td>- Implement test scheduling to align release and environment needs</td>
<td>- Improve Non-Functional Testing</td>
<td>- Improve Non-Functional Testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Emphasize a strategy focused on quality outcomes and delivery</td>
<td>- Implement automated CI processes</td>
<td>- Improve QA estimation model</td>
<td>- Improve QA estimation model</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Implement an enterprise test automation framework</td>
<td>- Increase engineering skillsets within the QA teams</td>
<td>- Enable QA teams to better organize and plan</td>
<td>- Enable QA teams to better organize and plan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Improve configuration and deployment management</td>
<td>- Double down on technology services especially test automation</td>
<td>- Improve defect management</td>
<td>- Improve defect management</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Identify business workflows, document them, and train on them</td>
<td>- Focus test case design and manual testing on business workflows</td>
<td>- Incorporate QA reviews into the SDLC and take it seriously</td>
<td>- Incorporate QA reviews into the SDLC and take it seriously</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Focus test automation on business workflows and test data</td>
<td>- Create, approve, and implement an Enterprise QA dashboard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2017 Trissential LLC – An entity of SQS Group

Confidential 34
Just Do It
Implementation Best Practices

- **Be Agile**
  - Develop in Sprints
  - Tackle the backlog
  - Be mindful of technical debt
  - Stay SIMPLE

- **Refactor often**
  - Have a purpose
  - Consider the cost

- **Focus on the core**
  - Keep as much functionality in the core as possible

- **Push updates to the core**
  - Allow project teams to focus on the project tasks
  - Control what flows up
Summary

As we wrap up today, see if you have answers to the following questions:

- How EFFECTIVE is your current testing approach?
- How EFFICIENT is your current testing approach?
- How do you SUPPORT faster Agile delivery?
- What are you doing to IMPROVE it?

Enterprise Test Automation
Q&A

rgormley@trissential.com
Thank you for your attention.