Teaching An Old Dog New Tricks: Agile For Legacy Systems

Craeg Strong
CTO, Ariel Partners
June 23, 2015
Mclean, VA
Craeg Strong

Software Development since 1988
Large Commercial & Government Projects
Agile Coach / Kanban Trainer
Performance & Scalability Architect
Apache Ant Open-Source Contributor
New York & Washington DC Area

CTO, Ariel Partners
CSM, CSP, CSD, CSPO, PSM, PMI-ACP, PMP, SAFe Agilist

www.arielpartners.com
cstrong@arielpartners.com
@ckstrong1
Agenda

1. Up Front Assertions
2. Why Is Legacy Different?
3. Situational Assessment
4. Whirlwind Tour
   1. Facilities
   2. Team
   3. Agile Methodology
   4. Technical Debt
   5. Documentation
   6. “Sprint 0”
   7. Lifecycle Management Tool
   8. Roadmapping
   9. Automated Tests
   10. DevOps Automation

5. Summary
Assertions

1. Big legacy projects represent a constituency that has been largely left out of the conversation about Agile.

2. Agile methods can provide benefits for big legacy software projects – even if they have been previously successful under waterfall.

3. Agile practices may need to be customized and introduced in a different order in a legacy project.
## How Is Legacy Different?

<table>
<thead>
<tr>
<th>Category</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Criticality</td>
<td>Mission-Critical, Heavily Regulated</td>
</tr>
<tr>
<td>Governance</td>
<td>Heavyweight, Prescriptive, Manual-Intensive Reporting</td>
</tr>
<tr>
<td>Facility</td>
<td>Optimized For Non-Collaboration, Private Offices May Be Tied To Self-Worth</td>
</tr>
<tr>
<td>Staff</td>
<td>Resistant to Change, Battle Fatigue</td>
</tr>
<tr>
<td>Documentation</td>
<td>Missing or Out of Date, Significant Documentation Requirements</td>
</tr>
<tr>
<td>Testing</td>
<td>Mostly Manual, System Not Designed For Testability</td>
</tr>
<tr>
<td>Procurement</td>
<td>Lengthy Approval Process, Skeptical of Open Source</td>
</tr>
</tbody>
</table>
Situation Assessment

**Assessment Questions**
- What does our capital budget look like? OpEx vs CapEx
- How long will the capability be needed?
- How many of the original team are available?
- What is the state of the code base relative to modern best practices?
- Does the software use a strategic platform? Is it still supported?

**Conclusions**
- How much we will invest in the existing code base
- How we will allocate our budget between maintenance and re-design
- How that allocation may shift over the next 5 years
Criminal Justice Project
Whirlwind Tour

1. Facilities
2. Team
3. Agile Methodology
4. Technical Debt
5. Documentation
6. “Sprint 0”
7. Lifecycle Management Tool
8. Roadmapping
9. Automated Tests
10. DevOps, CI/CD Automation
1. Facilities

Dream

Reality
Remove Barriers to Collaboration

- Co-Locate The Team
- Cubicles → Open Spaces
- Put Whiteboards Everywhere
- Post-Its of all shapes and colors
- Dual Hi-Res Monitors
2. Staffing The Team

- New Project
- Lightweight Governance
- Little Technical Debt

- Legacy Project
- Heavyweight Governance
- Significant Technical Debt

Traditional Scrum Team
- Developers
- Testers
- Business Analyst
- CM/Build Engineer
- DBA

Specialists
- UX Designer
- DevOps Engineer
- Technical Writer
- Tool/Automation Specialist

Increasing Complexity
Team Composition

New Staff
✓ Junior
✓ Midlevel

Specialists (Senior)
✓ UX Designer
✓ DevOps Engineer
✓ Build Automation Engineer
✓ Tool Specialist
✓ Technical Writer
✓ UI Test Automation Expert
✓ BDD/TDD Developer

Project Veterans (Mid/Senior)
✓ Senior Developer
✓ Domain Expert
✓ Expert Manual Tester
✓ Training Specialist
Summary: Structuring The Team

Mix of System Knowledge & New Automation Skills Needed

- Up To 50% Experienced Staff Members
- Most New Staff Should Bring Specific Automation Skills
- Relatively Few Junior Developers
- May Need Documentation Specialists
- Expect Some Turnover, Especially Veterans
3. Methodology

*According to VersionOne*
## Choosing A Methodology

<table>
<thead>
<tr>
<th>Awareness &amp; Precedent</th>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awareness, Acceptance, or Organizational Precedent</td>
<td>No Experience With Agile, or Previous Negative Experience</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nature of the Work</th>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best for Development or “Disciplined O&amp;M”</td>
<td>O&amp;M or Development plus O&amp;M</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stakeholder Availability</th>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-Boxed Iterations provide Convenient Planning Horizon for Stakeholders</td>
<td>Stakeholders cannot commit to fixed time-frames, or can engage more frequently</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External Dependencies</th>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort fairly self-contained, Cross-Functional team</td>
<td>Lots of dependencies on external teams</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Openness to Change</th>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td>Need or Want Clean Break from past: Revolution</td>
<td>Evolutionary approach is necessary</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Team Preferences &amp; Maturity</th>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ceremonies, Time-Box Desirable</td>
<td>Escape “Tyranny of the Time-Box,” Lengthy Planning Meetings</td>
<td></td>
</tr>
</tbody>
</table>
What If You Are Kind Of In The Middle?

<table>
<thead>
<tr>
<th></th>
<th>Scrum</th>
<th>Kanban</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Awareness &amp; Precedent</strong></td>
<td>Awareness, Acceptance, or Organizational Precedent</td>
<td>No Experience With Agile, or Previous Negative Experience</td>
</tr>
<tr>
<td><strong>Nature of the Work</strong></td>
<td>Best for Development or “Disciplined O&amp;M”</td>
<td>O&amp;M or Development plus O&amp;M</td>
</tr>
<tr>
<td><strong>Stakeholder Availability</strong></td>
<td>Time-Boxed Iterations provide Convenient Planning Horizon for Stakeholders</td>
<td>Stakeholders cannot commit to fixed time-frames, or can engage more frequently</td>
</tr>
<tr>
<td><strong>External Dependencies</strong></td>
<td>Effort fairly self-contained, Cross-Functional team</td>
<td>Lots of dependencies on external teams</td>
</tr>
<tr>
<td><strong>Openness to Change</strong></td>
<td>Need or Want Clean Break from past: Revolution</td>
<td>Evolutionary approach is necessary</td>
</tr>
<tr>
<td><strong>Team Preferences &amp; Maturity</strong></td>
<td>Ceremonies, Time-Box Desirable</td>
<td>Escape “Tyranny of the Time-Box,” Lengthy Planning Meetings</td>
</tr>
</tbody>
</table>
# Consider Scrumban

Using the Kanban Method for...

<table>
<thead>
<tr>
<th><strong>Adopting Scrum</strong></th>
<th>To facilitate adoption of Scrum, starting from wherever you are right now</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Demand Analysis, Workflow Analysis</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Amplifying Scrum</strong></th>
<th>Help identify, implement, and measure improvements for an existing Scrum team</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>WIP limits, Lead Time Metrics</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Scaling Scrum</strong></th>
<th>To facilitate inter-team planning and communication</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>Visualize and manage dependencies via Kanban boards, Portfolio Kanban</em></td>
</tr>
</tbody>
</table>
How Does Kanban Improve Scrum?

<table>
<thead>
<tr>
<th>Service Pack</th>
<th>Month 1</th>
<th>Month 2</th>
<th>Month 3</th>
<th>Month 4</th>
<th>Month 5</th>
<th>Month 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sprint 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Sprint Planning
- System Testing
- Government Acceptance Testing
- Retrospective
- Change Control Board

Change Control Board Meeting
Challenges We Encountered Using Scrum For Maintenance Project

- Change Control Board Meeting
- Influx of Expedited Items
- Expectation
  - Too Many Work Items
  - Recurring Need For Estimates
  - Constant Interruptions

- Reality
- Product Backlog
- Release Backlog

Release
- Month 1
- Month 2

Sprint 1
- Sprint Backlog

Sprint 2
- Sprint Backlog

© Copyright Ariel Partners 2015
sales@arielpartners.com (646) 467-7394
Why Were We Always So Busy?

<table>
<thead>
<tr>
<th>Work Item Type</th>
<th>Class Of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business User Story</td>
<td>Standard</td>
</tr>
<tr>
<td>Infrastructure User Story</td>
<td>Standard</td>
</tr>
<tr>
<td>Tier III Escalation</td>
<td>Expedited</td>
</tr>
<tr>
<td>Production Script</td>
<td>Fixed Date</td>
</tr>
<tr>
<td>Hot Fix</td>
<td>Expedited</td>
</tr>
<tr>
<td>Escaped Defect</td>
<td>Expedited</td>
</tr>
<tr>
<td>Documentation Deliverable</td>
<td>Fixed Date</td>
</tr>
<tr>
<td>Documentation Rework</td>
<td>Expedited</td>
</tr>
<tr>
<td>Technical Chore</td>
<td>Intangible</td>
</tr>
</tbody>
</table>

Kanban gives us the analytical tools to understand these work items better and to see why they have been disrupting the smooth delivery of value.
Summary: Choosing A Methodology

Pick Best-Fit For Situation

- Learn Both Kanban and Scrum
- Full-Time Coach
- Up-Front Training “Agile Bootcamp”
- Learning Kanban Makes You Better At Scrum
4. Technical Debt

Negotiate A High Percentage of Bandwidth To Be Spent Paying Down Technical Debt
Focus On Test Automation

- Ideally we have full automated test coverage
- We can potentially ship at the end of any sprint

Version Control
  - Compile
  - Automated Unit Tests
  - Automated Performance Tests
  - Automated Security Tests
  - Static Analysis

Automated smoke & verification tests

Begin Sprint

End Sprint
But Set Aside Extra Time For Manual Testing

PROS
• Faster Technical Debt Reduction
• Faster Progress Towards Continuous Delivery
• Additional Testing
• Fewer Regressions

CONS
• Reduced Business Value Delivery
• Additional Complexity
• Inconsistent Metrics

For our project, Technical Sprints were the “training wheels” we needed to maintain high quality in the absence of test automation. We have since discarded them.
Maintain High Levels Of Quality In Spite of Low Initial Automated Test Coverage

- Achieve Agreement On Need For Infrastructure Investment
- Allocate Additional Time Required For Manual Testing
- Balance Cost of Delay with Cost of Failure
- Testing / Hardening Sprints May Require Whole-Team Participation
- Remove Barriers To Manual and Automated Testing
5. Documentation

Traditional Governance
Comprehensive documentation is a critical component of software development

Agile
Working software over comprehensive documentation
Recognize That Documentation Requirements May Be Bursty

<table>
<thead>
<tr>
<th>Sprint</th>
<th>Sprint</th>
<th>Sprint</th>
<th>Sprint</th>
<th>Sprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procurement Guidance</td>
<td>Testing Documentation</td>
<td>Testing Documentation</td>
<td>Design &amp; Technical Documentation</td>
<td>Training Documentation</td>
</tr>
<tr>
<td>Interface Documentation</td>
<td>Testing Documentation</td>
<td>Testing Documentation</td>
<td>Design &amp; Technical Documentation</td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td>Testing Documentation</td>
<td>Testing Documentation</td>
<td>Design &amp; Technical Documentation</td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td>Testing Documentation</td>
<td>Testing Documentation</td>
<td>Design &amp; Technical Documentation</td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td>Testing Documentation</td>
<td>Testing Documentation</td>
<td>Design &amp; Technical Documentation</td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Training Documentation</td>
</tr>
</tbody>
</table>
Summary: Dealing with Documentation Requirements

Maintain High Quality Documentation That Is Valuable

- Improve Documentation Capabilities
- Plan and Measure the Work
- Reduce Amount of Documentation Required
- Reduce Effort Required to Produce Documentation
6. Sprint 0

1. People
   Staffing Reaches Critical Mass (~80%)

2. Hardware
   Every Team Member Has An Imaged PC

3. Software
   Source Code Manager Installed

4. Initial Product Backlog
   Small List of Business and Technical Items

5. Initial Kanban Board
   Simple Workflow, One Post-It Per Item
Summary: When To Start Iterating?

Don’t Wait: Start Iterating As Soon As Possible

- Automated Build May Take Extra Time To Setup…
- Automated Testing May Not Be Possible At First…
- Last Few Staffing Slots May Take Time To Fill…
- …But You Will Learn Quickly
  - Kanban Workflow Board Will Evolve Rapidly
  - Gather Baseline Metrics
7. Lifecycle Management Tool

- Defect Tracking Tool
- Requirements Tool
- Testing Tool
- Traceability Cross-Reference

- Agile Tool
  - One Tool Specialist
  - Custom Reporting Likely To Be Essential
Use Manual Kanban Board

<table>
<thead>
<tr>
<th>Backlog</th>
<th>To Do</th>
<th>Development</th>
<th>Done</th>
</tr>
</thead>
</table>

- Workflow Evolves Quickly
- Excel Can Be Used For Metrics
- Immature Agile Tool May Be Burden

<table>
<thead>
<tr>
<th>Backlog</th>
<th>Specify</th>
<th>Implement</th>
<th>Peer Review</th>
<th>Test</th>
<th>Deploy</th>
<th>Validate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next</td>
<td>Queue</td>
<td>In Process</td>
<td>Queue</td>
<td>In Process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(∞)</td>
<td>(2)</td>
<td>(4)</td>
<td>(2)</td>
<td>(3)</td>
<td>(∞)</td>
<td>(∞)</td>
</tr>
</tbody>
</table>

EXPEDITE (1)
Summary: Tool Chain Support For Agile

Simplify Tool-Chain & Automate Prudently

- Tool Consolidation May Be Beneficial
- Automated Reports Preferable To Manual Reporting – But Watch Cost versus Benefit
- Always Start With Manual Kanban Boards
- Capture Metrics In Excel To Start With
8. Roadmapping

<table>
<thead>
<tr>
<th>Release N</th>
<th>Release N+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint</td>
<td>Sprint</td>
</tr>
<tr>
<td>Sprint</td>
<td>Sprint</td>
</tr>
<tr>
<td>Sprint</td>
<td>Sprint</td>
</tr>
<tr>
<td>Sprint</td>
<td>Sprint</td>
</tr>
</tbody>
</table>

- **Feature 1**
- **Feature 2**
- **Installer Improvement**
- **Automated Builds**
- **Upgrade .NET**
- **Upgrade DB Libraries**
- **Upgrade GUI Libraries**

**Business Roadmap**

**Technical Roadmap**
Technical Improvements
Address Risks

Express Technical Items In Terms of The Business Risks They Mitigate

Bad
- Investigate Artifactory Binary Repository
- Migrate To Entity Framework
- Rewrite Error Handling Component

Good
- Improve Config Mgmt: Eliminate Unused Third Party Libraries
- Retire Obsolete & Unsupported Database Libraries
- Replace COM Components Preventing Native 64-Bit Operation
# Demonstrate Progress On Technical Items Via Reports

## Third Party Artifacts

<table>
<thead>
<tr>
<th>Release</th>
<th>2015-01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report Time</td>
<td>06/05/2015 16:06:56</td>
</tr>
<tr>
<td>Product Count</td>
<td>41</td>
</tr>
<tr>
<td>Artifact Count</td>
<td>95</td>
</tr>
</tbody>
</table>

### Commercial

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>11</td>
</tr>
<tr>
<td>Obsolete</td>
<td>5</td>
</tr>
<tr>
<td>Out Of Support</td>
<td>3</td>
</tr>
<tr>
<td>Out Of Date</td>
<td>2</td>
</tr>
<tr>
<td>Total Not Current</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>21</td>
</tr>
</tbody>
</table>

### Non-Commercial

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>14</td>
</tr>
<tr>
<td>Obsolete</td>
<td>4</td>
</tr>
<tr>
<td>Out Of Support</td>
<td>N/A</td>
</tr>
<tr>
<td>Out Of Date</td>
<td>2</td>
</tr>
<tr>
<td>Total Not Current</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td>20</td>
</tr>
</tbody>
</table>

## Commercial Products

<table>
<thead>
<tr>
<th>.NET Framework</th>
<th>Organization</th>
<th>Version</th>
<th>Support Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>(see MSDN)</td>
<td>4.5.1</td>
<td>OutOfDate</td>
</tr>
<tr>
<td>Support Purchased</td>
<td>(see MSDN)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support Expiry</td>
<td>RunTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dependency Type</td>
<td>ShippedWithProduct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usage Type</td>
<td>LowRisk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Impact</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artifacts</td>
<td>Microsoft.NetFrameworkOfflineInstaller_v4.5.1_For_x86_AMD64.exe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary: Maintain Both Business and Technical Roadmaps

Roadmaps Can Help Communicate The Big Picture

- May Help Replace Gantt Charts and Earned Value Management
- Avoid Overpromising
- Start With Technical Spike To Determine Feasibility
- Focus on Removing Technical Barriers First
- Older Platforms & Libraries May Not Support Automation
- Platform and Third Party Upgrades Require Significant Testing
9. Automated Tests

- Writing Automated Tests For Legacy Applications Not Designed For Testability Is Difficult
- We Are Asking Our Customer To Make A Large Investment In Automation With Delayed Return On Investment
- We Need To Maximize Our Efforts

If only we could..

- Create Tests That Don’t Break When We Change The GUI
- Get The Whole Team Involved
- Focus On Business Scenarios, Not Source Code
- Create Tests That Will Survive A Component Rewrite
- Create Tests That Support Traceability To Requirements
Given...
When...
Then...

Behavior Driven Development

Acceptance Test Driven Development
BDD With SpecFlow

1. Write Feature

2. Generate Template

3. Fill In Code
## The Agile RTM

- All system functions expressed as user stories
- All user stories have acceptance criteria
- Each criterion translated to an automated test using structured English (Gherkin)
- Customized report matches epics and user stories to automated acceptance tests
- Test fails unless software is implemented correctly

<table>
<thead>
<tr>
<th>TFS ID</th>
<th>Theme</th>
<th>TFS ID</th>
<th>Epic</th>
<th>TFS ID</th>
<th>User Story</th>
<th>Pass/Fail</th>
<th>Feature Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>104</td>
<td>Trading</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24 passed 1 failed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2387</td>
<td>Multi Option</td>
<td>1098</td>
<td>Add Covered Call</td>
<td>18 passed 0 failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2689</td>
<td>Butterfly Spread</td>
<td>10 passed 0 failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3104</td>
<td>Compare with S&amp;P BuyWrite Index</td>
<td>3 passed 0 failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1608</td>
<td>Simple Options</td>
<td>7510</td>
<td>Enforce Trading Limits For Option Trades</td>
<td>6 passed 1 failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7513</td>
<td>Calculate Put/Call Ratio</td>
<td>2 passed 0 failed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 passed 1 failed</td>
<td></td>
</tr>
</tbody>
</table>
Summary: Focus On High-Value Automated Tests

BDD/ATDD Tools Enable High-Value Tests

- “Three Amigos” Work Together (Analyst, Tester, and Developer)
- Tests Map Directly To Acceptance Test Criteria
- Code Tags Provide Traceability To Features or User Stories
- BDD/ATDD Tools Rapidly Maturing
- More Robust, Lower Barrier To Entry Than GUI-Level Test Tools
10. DevOps Automation

How To Achieve Vision of Full Automation?
Automate Each Piece of the Pipeline: One By One

Manual
Ad-Hoc
Undocumented

Analyzed
Documented
Baselined

Initial
Automation:
Stand-Alone
Script

Improved
Automation:
Build Script adds
Validation
Pre-Conditions
Post-Conditions

Full “Lights Out”
Automation:
Reporting
Diagnostics

This Step is Essential
Summary: DevOps Automation

Establishing Continuous Integration & Continuous Delivery (CI/CD)

- DevOps Tools Maturing Rapidly
- Support Varies Widely By Platform (Linux/Unix, Windows, Mainframe)
- Legacy Projects Typically Have Many Manual & Undocumented Processes (Folklore)
- Significant Analysis & Problem Solving Skills Required
- Many High-Value Candidates For Automation Are Non-Obvious
- Avoid Paving the Cow Paths
Summary

- Legacy Projects Can Realize Benefits By Adopting Agile Practices
- Practices May Need To Be Modified
- Consider Kanban or Scrumban
- Staff Your Team Thoughtfully
- Manage Stakeholder Expectations
- Balance Cost of Delay with Cost of Failure
- Adopt Systems Thinking Approach To Addressing Technical Debt
- Reduce Complexity
- There Is No Silver Bullet
Questions?

We are available for consulting or Agile coaching
We love to compare notes and talk shop!

Ariel Partners
(646) 467-7394
sales@arielpartners.com
www.arielpartners.com
@arielpartners

Thank You!