Charting the Best IT Decisions for Your Business
ENTERPRISE AGILE TRANSFORMATION AT THE US POSTAL SERVICE

MAY 24, 2013
Agenda

Scope of USPS IT Support

Challenges With IT Projects

USPS Agile Objectives / Scope

USPS Agile Roadmap / Status

Agile Engineering Practices

Lessons Learn

Realized Benefits / Challenges

What’s Next?
Scope of USPS IT Support

✓ 1300 IT Employees
✓ 150K Computers within 40K+ Facilities
✓ 894 Business Applications Across 4 Solution Centers
✓ One of the Largest Computing Infrastructure in the World
✓ Connects 38K Post Offices and 65K Retail Terminals
✓ 40K Web Pages on USPS.com - 384M Visits Per Year
Challenges With IT Projects

- Value of Business Requirements Not Defined / Prioritized
- Inconsistent Communication Between IT and the Customer
- Customers Don’t Always Know What They Want
- Time Spent Developing Large Requirements that Change
- Change is Viewed as Not Being Successful
- Limited Real-Time Transparency on Project Cost and Schedule
- Project Delivery / Quality Needs Improvement
USPS Agile Objectives

- Improve Communication
  - Constant Communication and Collaboration Between the Business and Across IT

- Provide Full Visibility
  - Projects Managed Based on Continuous Inspections & Useful Metrics

- Increase Project Success
  - Projects Completed on Time & Budget in Line with Customer Needs

- Improve Project Quality
  - Teams Use Continuous Integration Software & Automated Testing

- Speed To Market
  - Projects Centered Around Business Value Realized Quickly
Agile Program Scope

- Customer Satisfaction
- Speed to Market
- System Quality
- Project Success

- Business Planning
- Scrum Methodology
- Engineering Best Practices
USPS Agile Roadmap

- Continuous Improvement
- Light Coaching / Assessments
- End-to-End Automated Testing

- Mandate All Projects Agile
- Medium Coaching / Metrics / COP
- Expand Agile Planning / Engineering

- Change Policy / Agile Training
- Heavy Coaching / Start Engineering
- Controlled Agile Project Selection

- Assess The Organization
- Define Agile Roadmap
- Start Communications
Over 60 Agile Projects Delivered or In Progress

IT Policies and Procedures Reviewed and Updated with Agile

Standard Agile SW Support Tool Selected - Training Started

Metrics Developed to Monitor Agile Maturity / Engineering Progress

15 Projects Using Engineering Practices (i.e. CI, TDD, Code Quality)

USPS Continues to Build a Large Agile Knowledge Base

550+ People Trained in Scrum – 135 in CI, TDD, Code Quality Practices
Two Methodologies

Waterfall Development Methodology Phases

- Initiate / Plan
- Requirements
- Analysis/Design
- Build
- Sprints 1-N
  - Sprint 0
  - Sprint Planning
  - 2 Week Sprints
  - Sprint Build
  - Sprint Retrospective
  - Sprint CAT
  - Sprint SIT
  - SIT
  - CAT
  - Governance Compliance
  - Release Mgmt.

Agile Scrum Development Methodology Phases
Project Management

Agile Milestone Tracking

- Scorecard Indicator
- WBS
- Task Name

1.1.1 - Initiate and Plan
1.1.2 - Initiate - Analysis and Design
1.3.1 - Development - Sprint 1 through n
1.4.1 - Systems Integration Test
1.5.1 - Customer Acceptance Test
1.6.1 - Governance
1.7.1 - Release

Release & Sprint Velocity

Sprint Taskboard

- Backlog
  - In Progress
  - Completed
- Summary
- Test Results:
  - To Do:

Daily Burndown

Release Burndown

Issues / Risks

Senior Management View

Agile Project Team View

Release & Sprint Velocity

Detail Project Planning

Metrics SW

SDFF, Inc.
Agile Engineering Practices

**Phase 1 Objective:** Train St. Louis 118 and 15 Wilkes Barre Development Staff in Continuous Integration, Test Driven Development, & Code Quality Best Practices

**Training Approach:**
- 18 hours of classroom training / Hands-on Coaching for 4-8 Weeks
- Supplemental Training based on Skill / SW
- Measures Monitored Weekly and Discussed

---

**Open Source Software Tools Being Used**

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUnit</td>
<td>Automated Java Unit Tests</td>
</tr>
<tr>
<td>Mockito</td>
<td>Testing Modules in Isolation</td>
</tr>
<tr>
<td>EclEmma</td>
<td>Measuring Code Coverage</td>
</tr>
<tr>
<td>Cobertura</td>
<td>Measuring Code Coverage</td>
</tr>
<tr>
<td>DBUnit</td>
<td>Testing Data Access Layer</td>
</tr>
<tr>
<td>PJUnit</td>
<td>Testing PL/SQL Code</td>
</tr>
<tr>
<td>FindBugs</td>
<td>Static Analysis of Java Best Practices Violations</td>
</tr>
<tr>
<td>PMD</td>
<td>Static Analysis of Java Best Practices Violations</td>
</tr>
<tr>
<td>CheckStyle</td>
<td>Static Analysis of Java Convention Violations</td>
</tr>
<tr>
<td>Jenkins</td>
<td>Continuous Integration / Dashboard Platform</td>
</tr>
<tr>
<td>HttpUnit</td>
<td>Functional Testing of Web Applications</td>
</tr>
<tr>
<td>Selenium</td>
<td>Functional Testing of Web Applications / Browsers</td>
</tr>
<tr>
<td>Ant</td>
<td>Build Scripting Tool</td>
</tr>
<tr>
<td>Maven</td>
<td>Dependency Management and Build Tool</td>
</tr>
<tr>
<td>Gradle</td>
<td>Dependency Management and Build Tool</td>
</tr>
</tbody>
</table>

---

**Training Courses Provided**

- Unit Testing with JUnit 4
- Test-Driven Development with JUnit
- Testing Classes in Isolation with Mockito
- Functional Web Testing with HttpUnit
- Functional Web Testing with Selenium WebDriver
- Improving the Structure of a System with Refactoring
- Improving Code Quality with SOLID design principles
- Automated Build Principles and Practices
- Continuous Integration with Jenkins
- Continuous Integration with Ant and Gradle
Engineering Levels

<table>
<thead>
<tr>
<th>Training</th>
<th>Novice</th>
<th>Journeyman</th>
<th>Expert</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Receives Overview of USPS Development Standards</td>
<td>• Practices USPS Development Standards</td>
<td>• Understands Simple Design</td>
<td>• Practices No Big Design Up Front</td>
</tr>
<tr>
<td>• Learns Basic TDD</td>
<td>• Practices Collaborative Code Ownership</td>
<td>• Participates in Peer Code Reviews</td>
<td>• Practices Pair Programming</td>
</tr>
<tr>
<td>• Learns about Team Development</td>
<td>• Practices TDD</td>
<td>• Practices Regular Refactoring</td>
<td>• Uses Build Reports Daily</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test</th>
<th>Build</th>
<th>Deploy</th>
<th>Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Learns JUnit 4</td>
<td>• Learn what Continuous Integration is</td>
<td>• Learns how to Automate Builds</td>
<td>• Learns how to use Tool Output</td>
</tr>
<tr>
<td>• Learns how to create Mock Objects with Mockido</td>
<td>• Learns how to leverage tools for Continuous Integration</td>
<td>• Uses Automated Builds</td>
<td>• Uses Helper Scripts for Deployments</td>
</tr>
<tr>
<td>• Understands Different Types of Tests</td>
<td>• Learns Test Automation</td>
<td>• Outputs Build Process Reports</td>
<td>• Uses Automated Scripts</td>
</tr>
<tr>
<td>• Learns Test Automation</td>
<td></td>
<td></td>
<td>• Uses Automated Deployments</td>
</tr>
</tbody>
</table>

| | | | |
| • Uses Automated Unit Tests | • Uses Automated Builds | • Uses Automated Scripts | • Leverages Jenkins for Automated Deployments |
| • Always Tests Before Committing | • Outputs Build Process Reports | • Uses Automated Deployments (DEV) | • Produces Unified Deployments included release notes and bug reports |
| | • Automates Nightly Snapshot Builds | • Uses Manual Deployments (SIT, CAT, PROD) | |
| | • Leverages Jenkins Heavily | | • Leverages Jenkins |
| | • Automates Dependency Management | | • Integrates Test Output with Bug Tracking System |
| | | | • Increases Tests based on Risk |
| | | | • Automates Regression Tests |
| | | | • Automates Build Output Reports |
| | | | • Integrates Additional Reporting and Feedback Mechanisms |
| | | | • Automates Build Process Output |
| | | | • Uses Reports as Input |
| | | | • Project Outputs are Visible to entire Organization |
| | | | • Project Outputs are Used as Team Measuring |
Lessons Learned

- Mandating Agile Practices May be Necessary
- Executive Management Must Support and Encourage
- Assist the Customer in Embracing Agile
- Use Metrics to Monitor Progress and Maturity
- Mandating Agile Practices May be Necessary
- Experienced Coaching is Critical – Don’t Just Train
- Higher Risk of Failure Blending Waterfall & Agile
- Do Not Modify Agile Processes Around Existing SW
- Communicate Successes, Program Progress, Direction
Realized Benefits

- More Projects Completing On-Time / Within Budget
- Issues Identified Earlier and Being Escalated
- Improved Project Communications Overall
- Customer Addressing High Business Value First
- Starting to See Code Quality Improvement
- Increased Customer Satisfaction
Challenges Continue

- Continued Resistance to Change
- Dedicated Project Resources
- Expanding Agile on Large Projects
- Firm Fixed Price Contracts
- Change in Employee Skills / Job Descriptions
What's Next?

- Speed To Market
- Quality Business Solutions

- Expand Agile to All Projects
- Measure, Baseline, And Continuously Improve
- Continue to Train and Coach IT Employees in Agile
- Enhance Quality through Engineering Best Practices