An Iteration in the Life of an Agile Tester

Better Software 2008

Lisa Crispin
With Material from Janet Gregory
Introduction

- Me: Coding, testing
- Joined first agile team in 2000
  - Tester's place in agile unclear!
- Currently on Scrum/XP team developing Java-based web app
  - Since 2003
- Help agile teams/testers
Introduction

How about you?
- What do you hope to take from this tutorial?
Goals

When you leave, you'll know:
- How to plan testing for agile projects
- How agile testers hit the ground running
- How testers collaborate, and who with
- How to ensure a successful release
- How agile testers help their teams improve their product
- Other items you brought up
Let's Follow an Agile Tester...

...through an iteration, and more:

- Release/Theme Planning
- Laying the Ground Work
- Iteration Planning
- Coding and Testing
- Successful Delivery
- Wrap Up the Iteration
Preparing for the next few iterations
- Sizing stories
- Help business set priorities
- 20,000 ft. view
- Test planning

Different teams take different approaches
- Only plan first few stories
- “Lean”
Release/Theme Planning (2)

- Avoid too much work up front
  - Priorities change!
- Identify stories to be done first
  - Steel thread or thin slice
- Consider impact on larger system
- Don't forget load, security, reliability, other utilities
Sizing Stories

Estimate the relative size of each story
- Planning poker
- Reduce scope as needed
- Consider risk
- Quick process (they're just estimates!)

“Planning poker” originated with Mike Cohn, Mountain Goat Software
How Testers Contribute (1)

Consider different viewpoints:
- What problem is this solving?
- What business value does it deliver?
- How will the end user use it?
- What's the worst thing that can happen?
- Programmer's perspective – how to implement
- Ask open-ended questions
How Testers Contribute (2)

Identify hidden assumptions
- Mismatch between different viewpoints
- “Obvious” from customer or programmer perspective
- Non-functional components
  - Security
  - Performance
  - Usability
  - ...
How Testers Contribute (3)

- Infrastructure considerations
  - Test environments, data
  - Tools
  - Resources
- Embedded systems?
- Outdated hardware?
- Third parties?
- Legacy code?
Group Exercise

Divide into small groups. Appoint one person to act as the product owner and answer questions. Size each story as small, medium or large. What assumptions might be missed? What are some different perspectives? Any special concerns that affect size?

- As an online shopper, I want a “smart” toy shopping tool so I can find an appropriate gift
- As an online shopper, I want a simple way to search for a brand or type of toy so I can see a list of results
- As an online shopper, I want to be able to see the cost of different shipping options when I check out, so I can choose
A Little Prep Work

- Agile tries to work “just in time”
- But, need to hit the ground running
- Some preparation may save time during iteration
- Don't do it if it doesn't save time
- Be proactive
- Help customer prepare
“Pre-Planning”

- Development, Customer Team together
- Discuss each story
  - Examples, user acceptance tests
  - Brainstorm – risks, dependencies, design
- Identify unusual testing needs
  - eg. load, security
- Help customers get advance clarity
  - accommodate different needs
  - but achieve consensus
Small Chunks, Steel Threads

Identify steel threads / thin slices / tracer bullets

- Test/code/test basic path that delivers value
  - Quick feedback on design, test approach
- Then test/code/test next little chunk
- Continue until story is complete
Steel Thread Example

A steel thread diagram, with “passes” numbered 1, 2, 3, 4.
Distributed Teams

- Keep remote team members, customers in loop
  - Get their input for planning
  - Divide up work
- May need extra roles
  - “Functional analysts”, proxies
- Tools for collaboration, communication
  - Tracking tasks
  - Requirements, tests
  - Conferencing, pairing
Resources, Technical Solutions

- Identify expertise not already on team
  - Bring in specialists if needed
  - Or budget time to develop expertise
- Spike technical solutions
- Story to research technologies, architecture
- Tools, infrastructure
Do You Really Need Pre-Planning?

Consider if:
- Distributed teams
- Less experienced team
- High risk stories

Maybe not if:
- These activities can be part of iteration
- “Lean” approach
Group Exercise

Given the following story or feature set, what do you think is the “steel thread”? What features might be added on once the steel thread works?

As an Internet shopper, I want to select shipping options for my items during checkout and see the shipping cost.

Assumptions: User has already entered shipping address. User will be able to choose different options for different items. The options are USPS, Ground, 2 day and Overnight. PO Boxes are USPS only. Items > 20 lbs are Ground only. API to cost calculator available, takes postal code and weight.
Iteration Kickoff

- Iteration Planning
- Tasks, estimates
- High level tests
- Examples
- Big picture
- Collaborate with customers
- Review with developers
- Test data
Iteration Planning (1)

- Address stories in priority order
- If no pre-planning, product owner:
  - Explains purpose
  - Gives examples of how it will be used
  - Provides mockups, wireframes
Iteration Planning (2)

- If not already done:
  - Steel thread exercise
- Evaluate story size
  - Break up if too large
- Watch for “scope creep”, “bling”
- Keep “big picture” in mind
How Testers Contribute (1)

- Again, consider all viewpoints
  - Stakeholder
  - User
  - Programmer
  - Technical writer

- Focus on examples
- Suggest steel thread exercise
- Evaluate story size – break up if too big
How Testers Contribute (2)

Ask questions

- What's the business goal?
- Can the user mess up?
- How do they fix it?
- Are we working with a vendor or specialist?
  - How do we coordinate testing?
- Can we obtain data for testing?
Planning Tasks (1)

- Write development and testing cards together
  - Some teams start with tests
- Some teams write testing tasks on development cards
- Other teams write separate testing cards
  - Color coding helpful
    - Physical or virtual
Planning Tasks (2)

- Write cards for anything that might be forgotten
  - Unit tests
  - Showing UI to customers
  - Send test files to vendor
- Write cards for performance, reliability, other 'ility' testing
- Identify information needed for test cases
  - Vendor API requirements
Estimating Tasks

- Use hours (remember, just estimates!)
- Consider alternatives
  - Scale tasks so they’re all one day's work
  - Write big task card to be broken down when more is known
- Be conservative
- Consider lean approach
Testable Stories

- Think about how you can test each story
- Work from examples
- Team solves testing, testability problems
  - Ex. Way to override server date/time
- Design code in layers
  - Testable at each layer
High Level Tests and Examples (1)

- Start with tests written together
- Start with “big picture”
- Turn examples into tests
- Lots of options
  - Spreadsheets
  - Bullet points
  - Graphical
  - Use cases
  - Matrix

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Errors</th>
<th>Warnings</th>
<th>Product 1</th>
<th>Product 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
High Level Tests and Examples (2)

- Capture somewhere entire team can see
  - Wiki
  - Whiteboard
- Use screenshots, mockups, visual aids
- Customer-friendly format
- Forms core of documentation
- Include non-functional requirements
  - eg. Security, usability
High Level Tests and Examples (3)

- Review with developers
  - Face to face communication best
  - When remote, use tools
    - As direct as possible
    - Adjust work schedule
- Review with customers
  - Collaborate closely
- Leave details for later
  - When coding starts
Story: As an Internet shopper, I want to select shipping options for my items during checkout and see the shipping cost.

- Divide your team into testers and customers.
- Write high level tests for the story. Use whatever format and techniques you like.
- What are some examples of desired behavior?
- What would help make the story testable?
- What requirements might change later?
Coding and Testing

- Coding and testing are part of one process
- Write detailed, executable tests
- Collaborate with developers
- Automate tests
- Do exploratory testing
- Keep up
Write Detailed Tests (1)

- High level tests get programmers started
- Don't let coding get way ahead of testing
- Write executable tests
- But think about exploratory testing scenarios
Write Detailed Tests (2)

- Start simple
  - Simplest happy-path test case
    - Developer helps automate
  - Once passing, add more complex tests
  - Add passing tests to build
- We can't keep up without automating regression tests
- No time for manual exploratory testing without automation
Driving Development with Tests

▪ All team members collaborate on stories
  ▪ Focus on completing one story at a time
▪ Iterative process
▪ Revise tests as needed
▪ Use risk analysis as guide
▪ Power of three
  ▪ Tester, developer, customer discuss questions, issues
Dealing with Bugs

- Aim for zero-defect development
- Show developer the problem
  - Write bug if it might be forgotten
- Pair with developer to find bugs
  - Customer, too
- Look for patterns
- Use simplest tool for tracking bugs
Testing Tasks

- Anyone on team can take one
  - Important if there's a crunch
- If lacking automated regression tests -
  - Everyone on team should do manual regression tests, each iteration
  - Great motivation for designing testable code, solving automation problems
Talk to Customers

- Mock up UI's, reports
  - Simply – on paper, whiteboard
- Take time to understand the business
  - Sit with customers
  - Learn their jobs
  - Get their input

Roth 1099R Tax Information*

<table>
<thead>
<tr>
<th>Roth 1099R Tax Information*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roth Syr Holding Period Start Date:</td>
</tr>
<tr>
<td>Qualified Roth Distribution:</td>
</tr>
<tr>
<td>Roth Deferral Contributions:</td>
</tr>
<tr>
<td>Roth Deferral Earnings:</td>
</tr>
<tr>
<td>Fee Amount:</td>
</tr>
<tr>
<td>Gross Distribution (Box 1):</td>
</tr>
<tr>
<td>Taxable Amount (Box 2b):</td>
</tr>
<tr>
<td>Tax Withheld (Box 4):</td>
</tr>
<tr>
<td>Roth Contributions (Box 5):</td>
</tr>
<tr>
<td>Form 1099R Year:</td>
</tr>
</tbody>
</table>

*sum of contributions and earnings (14465.07)
Regression Tests = Safety Net

- Multiple builds
  - Unit
  - Functional
- No build process?
  - Team should address
- Keep the builds "green"
- Keep feedback loop short
Exploratory Testing

- Lets you learn more about the features
- May produce more executable tests
- Leverage automation to facilitate
- Follow “smells”, your instincts
- Take notes, record results
- Time box
Group Exercise

Story: As an Internet shopper, I want to select shipping options for my items during checkout and see the shipping cost.

Pick a role and devise exploratory scenarios which that role might get into (don't limit yourself to the list below)

- High-stakes gambler
- Blackjack dealer
- Blue Man Group
- Penn and/or Teller
- Stand-up comedian
Successful Delivery

- The End Game
- UAT
- Packaging
- Production support
- Training
- Customer expectations
- Releasing
The End Game

- Some teams release each iteration
- Others wait for enough value
- End game may be a few hours, or days
- Not a bug fix cycle
- Don't code right to the last minute
- Doesn't have to be panic mode
User Acceptance Testing

- Performed by all affected business groups
- Verify existing and new business functionality
- Bugs deferred for future iterations
  - (except showstoppers)
- Shrink-wrapped software
  - Plan UAT at customer site
- Installation testing
Staging Release

- Test database changes, migrations
- Test with other systems
  - Inside, outside organization
- May be tied to client's release schedule
- Opportunity for final exploratory, end to end testing
- Write task cards for release tasks
- Learn system down time required
Deliverables (1)

- Who's accepting the product?
- What are their expectations?
- EX. Sarbanes-Oxley compliance measures
- How much documentation is enough?
  - Who is it for?
  - How are they using it?
Deliverables (2)

- Internal customers
  - What will make their jobs easier?
    - Workflows to understand new features
    - Workarounds for problems

- Training
  - Formal training sessions
  - Online help, tutorials
Releasing (1)

- Make product available to customers
  - Update website
  - Packaging
  - Deliver custom app to customers
  - Shrink-wrapped and delivered or downloaded
  - Start early

- Acceptance criteria
  - Define when product is “done”
  - Enough value
Releasing (2)

- Release management may fall to testers
  - Release readiness meeting
  - Release readiness checklist
  - Identify risks
  - Understand impact on business

- Release notes
  - Should fit needs of audience
  - Useful for future tests
We're releasing the stories to allow shoppers to select shipping options and see the shipping costs during checkout. The shipper provides the API for the cost calculation. New columns will be added to the database to collect the shipping costs.

- What events and activities would you plan for the staging release?
- What might be some of the deliverables?
- What plans would you make to ensure a successful production release?
Wrap Up the Iteration

- Iteration review
- Improving your team's process
- Celebrating success
- Dealing with change
Iteration Review

- Demo completed stories to customers
  - Real, live, working code
  - Customers can ask questions and give feedback
  - Sense of accomplishment
- Testers may conduct the review
Improving Your Process

- Team retrospective, every iteration
- Review previous list of “start, stop, continue” items
- Identify what worked, what didn't
- Write task cards or set guidelines to fix issues
- Aim for steady, sustainable pace over course of iteration
Celebrate Successes

- Recognize achievements
  - 3,000 Unit Tests
  - Value delivered to business
- Reward small accomplishments
- Celebrate individual successes
- Celebrate achievements of other teams as well

No donkeys were harmed in the production of this tutorial
Change Is Hard

- Need time, training to master new skills
  - Value quality over speed
- Build your credibility
  - Show how you add value
  - Avoid “quality police” mentality
- Find creative ways to encourage change
  - Make it a team problem to solve
  - Look for areas of greatest pain
Group Exercise

What are you doing now to work more closely with developers? With customers?

Compile a “top three” list from your group to share.
Some Agile Testing Resources

- lisa.crispin.home.att.net
- www.agilealliance.org
- www.testing.com
- agile-testing@yahoogroups.com
- www.fitnesse.org
- webtest.canoo.com
- fit.c2.com
Exploratory Testing Resources

- Testing Computer Software, Kaner
- Lessons Learned in Software Testing; Kaner, Bach, Pettichord
- www.testinglessons.com
- http://groups.yahoo.com/group/software-testing/
- http://www.satisfice.com
Agile Resources

User Stories Applied
by Mike Cohn
Agile Resources

Agile Estimating and Planning

By Mike Cohn
Collaboration

Collaboration Explained: Facilitation Skills for Software Project Leaders
By Jean Tabaka
Available on Amazon
Implementing Change

Fearless Change: Patterns for introducing new ideas
By Linda Rising and Mary Lynn Manns
Available on Amazon
Agile Testing Resources

Available on Amazon
Coming in 2009!

Agile Testing: The Role of the Tester in Agile Projects

By Lisa Crispin and Janet Gregory

www.agiletester.ca
Goal

Have fun, whatever you do!