

Cover Your Testing Bases with the Agile Testing Quadrants



STARWEST 2008

Lisa Crispin

*With Material from Janet Gregory
and Brian Marick's Agile Testing Matrix*

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



Goals

- When you leave, you'll know how to use the agile testing quadrants to:
 - Identify the types of testing needed
 - Identify who should do each type, and when
 - How best to accomplish each type
 - Where to start



Goals

How about you?

- What else do you hope to take from this tutorial?



Agenda

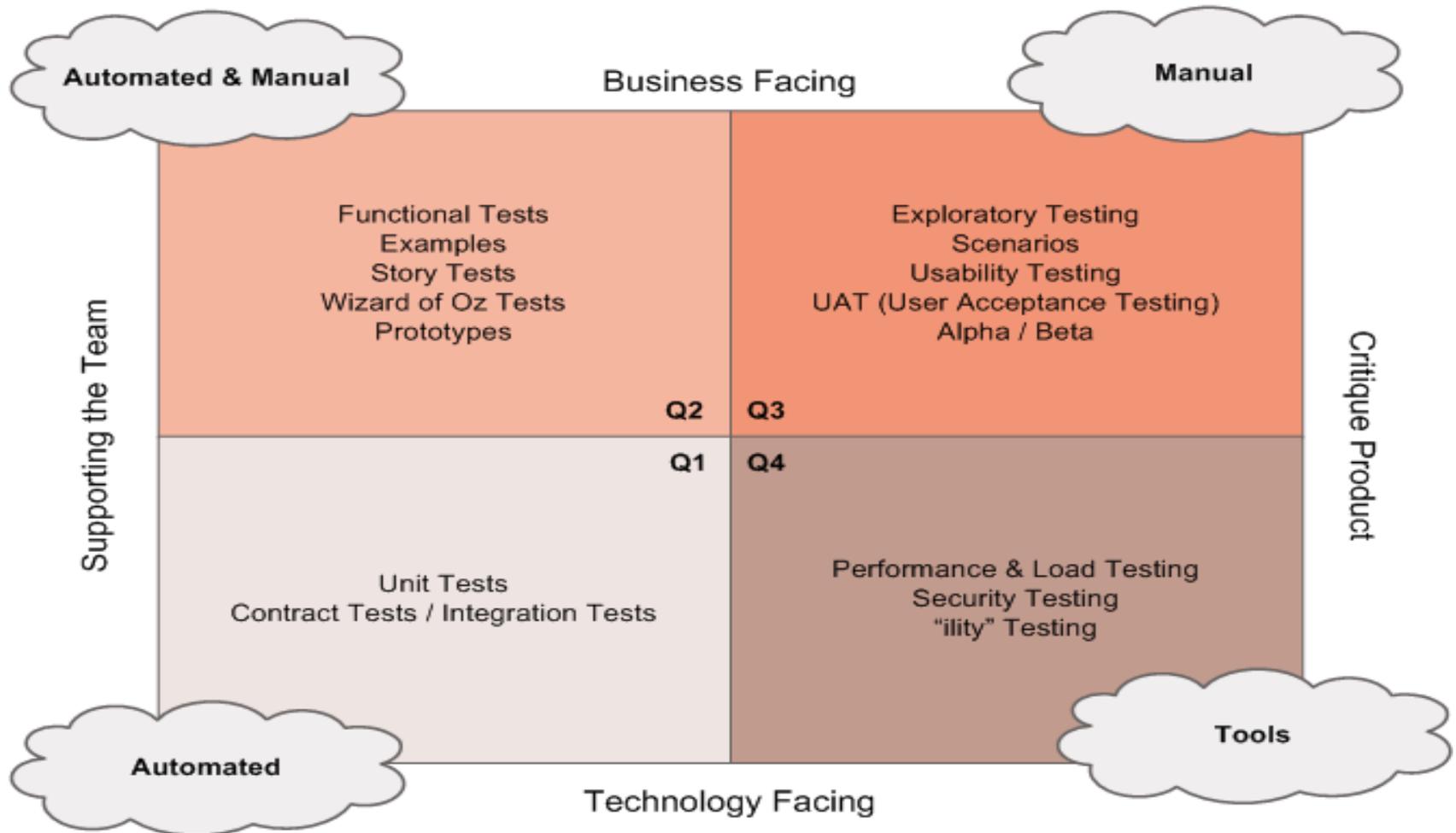


- Overview of Quadrants
 - Purpose of testing
- Quadrant 1:
 - Technology-facing tests that support the team
- Quadrant 2:
 - Business-facing tests that support the team
- Quadrant 3:
 - Business-facing tests that critique the product
- Quadrant 4:
 - Technology-facing tests that critique the product
- Planning your strategy



The Agile Testing Quadrants

Agile Testing Quadrants



The Purpose of Testing

Different goals, for example:

- Find bugs
- Make sure system is reliable
- Learn about the application
- See if UI is usable
- Feedback to future stories
- Doneness
- Manage technical debt
 - deferred work
 - hacks, untidy work
 - Slows the team down



Using the Matrix

- Quadrant matrix helps ensure we accomplish all goals
 - Support team
 - Critique product
 - Ensure business needs met
 - Ensure technological needs met
- Shared responsibility
 - Special skills may be needed
 - Focus on collaboration



Quadrant One

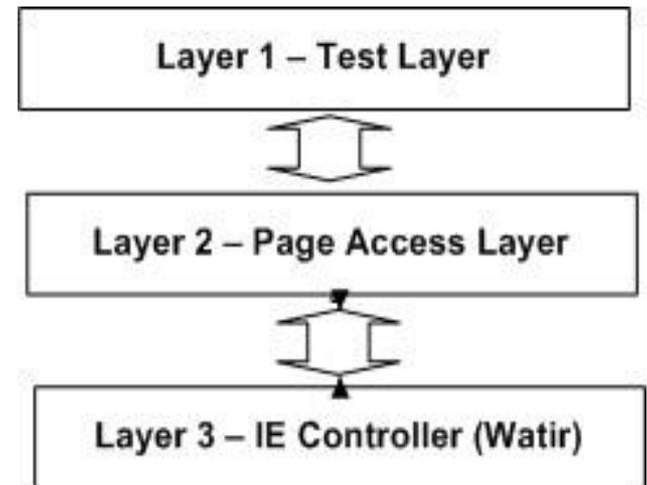
- Technology-facing Tests that Support the Team
 - Goals, reasons we do these tests
 - Who does these tests, when
 - What if your team doesn't do them?
 - Toolkit



Goal of Quadrant One Tests

Testability

- Layered or “componentized”
 - APIs
- Test database access, updates
- Business logic and presentation separated
- Isolate tests
 - allows isolating problems



Quadrant One Test Benefits

- Go faster, do more
 - Unit tests provide safety net
 - Refactoring support
 - Improve design & maintainability without changing functionality
- Quality, not speed, is goal
- Courage
- Confidence in design



Quadrant One Core

Test-Driven Development

- Write test for small piece of functionality
- Write code to make test pass
- Iterate on next small piece
- Design - “developer intent”
- Executable specification



Types of Tests in Quadrant One

- **Unit Tests**
 - Developer intent – program design
 - Small piece of code does what it should
- **Component Tests**
 - Architect intent – system design
 - Components work together correctly



Quadrant One Test

- Internal quality
 - “Zero defect” goal
 - Time for exploratory testing
- Supporting infrastructure
 - Source code control
 - Continuous Integration
 - Stable builds for testing
 - Feedback throughout project



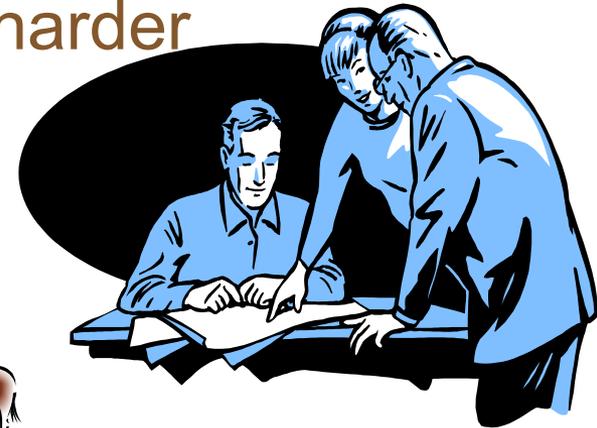
Who and When

- Programmers do Q1 tests
 - Design tool
 - Internal quality
- Done from start of each iteration
 - Test-first
- Fixing bugs
 - Write a test first
- Continually refactor
 - Maintainability
 - Minimize technical debt



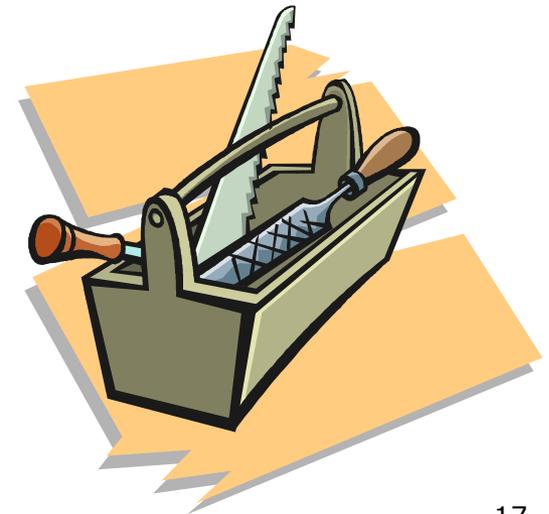
If Your Team Doesn't Do These ...

- It's a team problem
 - Brainstorm as a team
- Find areas of greatest pain
- Testers writing unit tests isn't the answer
- Managers must provide time to learn
- Without Quadrant One,
 - the other quadrants will be much harder



Quadrant One Toolkit

- **Source code management**
 - Version control
 - Know what has been changed, by whom
 - Be able to restore earlier version
- **Integrated development environment**
 - compile, debug, build GUI, refactor
 - eg. Eclipse, IntelliJ Idea, NetBeans
- **Build tools**
 - eg. CruiseControl, Hudson
- **Unit test tools**
 - xUnit
 - Mocking tools



Questions?



Quadrant Two

Business-facing Tests that Support the Team

- Purpose of Quadrant Two tests
- Customer test driven development
- Incremental, iterative development
- Doneness
- Who does them, when
- Toolkit



Purpose of Quadrant Two

- Drive development with business-facing tests
- Obtain enough requirements to start coding
- Help customers achieve advance clarity
- Capture examples, express as executable tests
- External quality
- Know when we're done



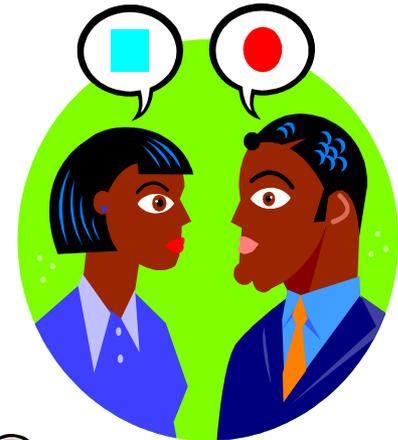
Driving Development with Tests

- The requirements struggle
 - High level tests give big picture to start coding
 - Detailed examples flesh out requirements
- Ask the right questions
 - Find hidden assumptions
- Capture examples in executable tests
 - eg. Fit/FitNesse tests



Creating “Story Tests”

- **Consider all viewpoints**
 - Business – what problem being solved?
 - End user – what will they do before, during, after?
 - Programmer – what's the simplest implementation?
- **Identify mismatches**
 - Can we implement a solution that doesn't solve problem?
 - Consider ROI



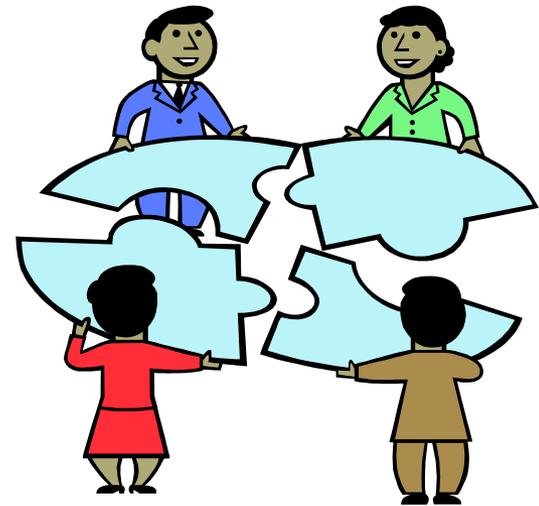
Enough Requirements to Start Coding

- “Advance Clarity” for customers
 - speak with one voice
- Convey the “big picture”
 - Examples
- Start with “happy path”
- Complete simplest end to end thread
 - coding
 - testing
- Then add more complex and edge cases



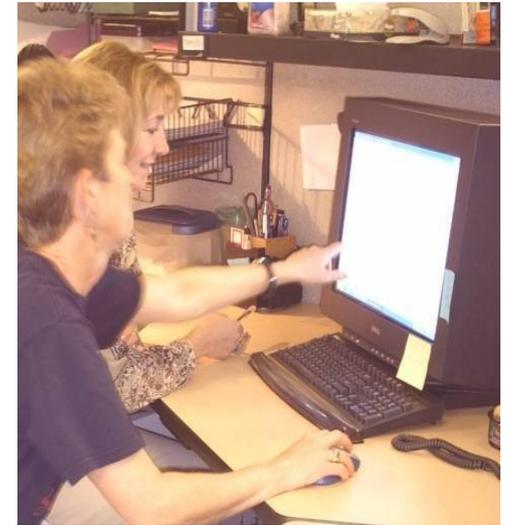
Doneness

- No story is done until tested
- Automated regression tests
- Customer requirements captured as passing tests
- Delivers value
- Doneness in all quadrants



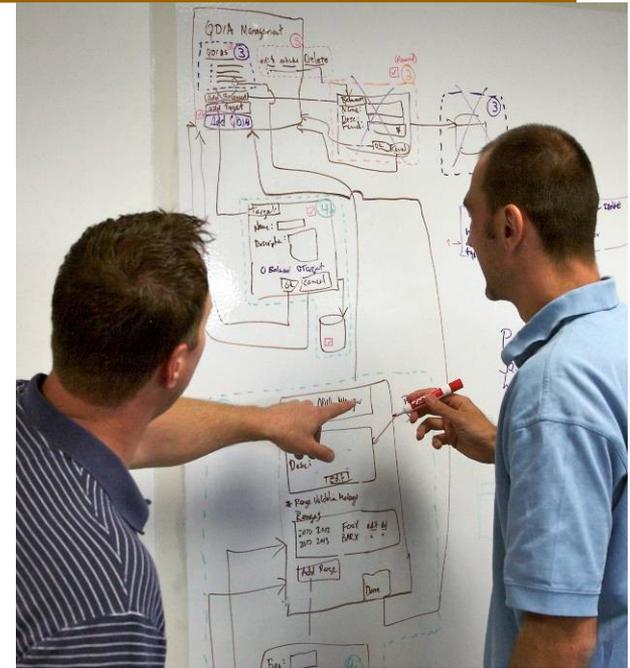
Who Does Quadrant 2 Tests, When?

- Testers have special expertise
- Collaboration with customers
- Team responsibility
 - Programmers
 - DBAs, analysts, ...
- Start of iteration
 - Business-facing tests drive development
- Throughout iteration
 - No story done until tested



Toolkit – Eliciting Requirements

- Checklists
- Mind maps
 - brainstorming
 - words, ideas, tasks
- Mockups / paper prototypes
 - User-centered design
- Flow diagrams
- Whiteboards (physical and virtual)
- Thin slice/steel thread exercise



Mock-Up Example

sum of contributions and earnings (14466.07)

Roth 1099R Tax Information*

Roth 5yr Holding Period Start Date:	01-04-2007
Qualified Roth Distribution:	No
Roth Deferral Contributions:	\$13,298.00
Roth Deferral Earnings:	\$1,168.07
Fee Amount:	\$50.96
Gross Distribution (Box 1):	\$14,415.11
Taxable Amount (Box 2b):	\$1,168.07
Tax Withheld (Box 4):	\$233.61
Roth Contributions (Box 5):	\$13,298.00
Form 1099R Year:	2008



Toolkit – Turning Examples into Tests

- **Fit/FitNesse**
 - collaboration in software development
 - Takes place of regular UI

Build Employees Fixture												
userid	dob	doh	doe	dot	directOwnerPct	lookbackTotalOwnerPct	lookbackAnnualComp	annualComp	deferral	eligibleComp	match	addl
1001	01-01-1950	01-01-1993	01-01-1994	null	0	0	101500.00	102500.00	16000.00	102500.00	16000.00	true
1002	01-01-1960	01-01-1993	01-01-1994	null	4	3	102500.00	102500.00	13000.00	102500.00	13000.00	true
1003	01-01-1960	01-01-1993	01-01-1994	null	5.01	5.01	30000.00	30000.00	7500.00	30000.00	7500.00	true
1004	01-01-1960	01-01-1993	01-01-1994	null	10	10	20000.00	30000.00	3000.00	30000.00	3000.00	true
1005	01-01-1960	01-01-1993	01-01-1994	null	8	0	40000.00	40000.00	8000.00	40000.00	8000.00	true
1006	01-01-1960	01-01-1993	01-01-1994	null	5.01	0	150000.00	150000.00	13000.00	150000.00	13000.00	true
1007	01-01-1960	01-01-1993	01-01-1994	null	0	0	100000.00	100000.00	0	100000.00	0	true
1008	01-01-1960	01-01-1993	01-01-1994	null	0	0	40000.00	50000.00	3000.00	50000.00	3000.00	true

OPERATE ON INPUT BY RUNNING ADP TEST

Operate Adp Test Fixture
operatel
true

MAKE ASSERTIONS ABOUT ADP TEST RESULTS

Check Employee Fixture				
userid	isHce?	isEligible?	adr?	acr?
1001	true	true	12.682927	15.61
1002	true	true	12.682927	12.68
1003	true	true	25.00	25
1004	true	true	10.00	10
1005	true	true	20.00	20
1006	true	true	8.666667	8.67
1007	true	true	0	0
1008	false	true	6	6



More Tools to Turn Examples into Tests

- **behavior-driven development tools**
 - Another approach to TDD
 - Focus on examples, “should”
 - easyB, jBehave
- **GUI test tools**
 - Test UI behavior, system test, legacy systems
 - Some examples:
 - Selenium
 - Watir/Watin/Watij
 - WebTest
 - QTP



Automation Approach

- **Incremental**
 - Start simple
 - Small chunks
 - Build on success
- **Patterns**
 - Build/Operate/Check
 - Event-based
- **Setup/Teardown**
 - Repeatable tests
 - Related to build/operate/check



Example Event-Based Test

1. Check the calculated loan payment after taking the loan.
2. Post the payment, then receive it.
3. Settle and confirm the payment
4. Check the interest, principal, loan balance and default state.

Loan Processing Fixture									
take loan in the amount of	1000	with interest rate	6.0	frequency	Monthly	and term	1	year with loan origination date	12-31-2005
check	periodic payment is	86.07							
post payment	1	of	86.07	on	01-30-2006				
receive payment	1	of	86.07	on	01-31-2006				
settle and confirm payment	1								
check	interest applied for	1	is	5.10					
check	principal applied for	1	is	80.97					
check	loan balance is	919.03							
as of	02-01-2006								
check	default state is	Not in Default							



Questions?



Exercise

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

The first “slice” is: After entering the shipping address, the user goes to a page to choose the shipping option, with ground shipping as the only choice. When the user submits, display the cost returned from an API cost calculator that takes postal code and weight.

Pair with another person. One of you plays tester, the other customer.

Draw paper prototypes for this thread. Testers will ask customers for details and examples.

Write some tests that would help drive development.



Quadrant Three

Business-facing Tests that Critique the Product

- Demos and testing with customers
- Exploratory testing
- Usability testing
- Testing behind the GUI
- Feedback to Quadrants One and Two
- Toolkit



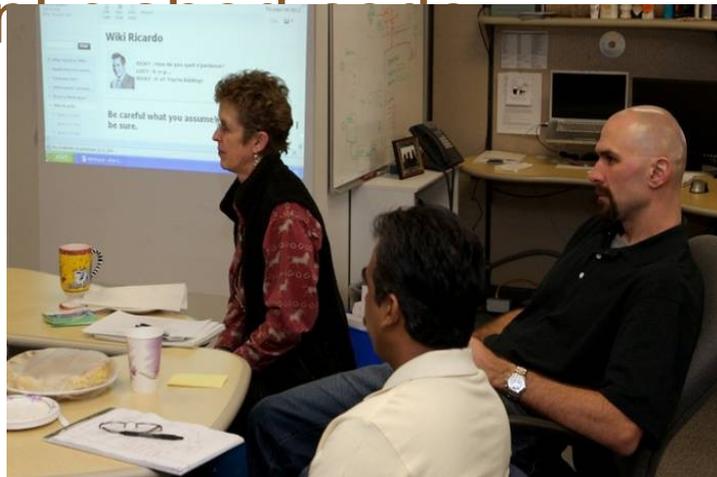
Evaluating the Product

- Recreate actual user experiences
- Realistic use
- Learn as you test
- Context
 - What works for your situation
 - “It depends”
 - A tool, not a rule
- Constructive



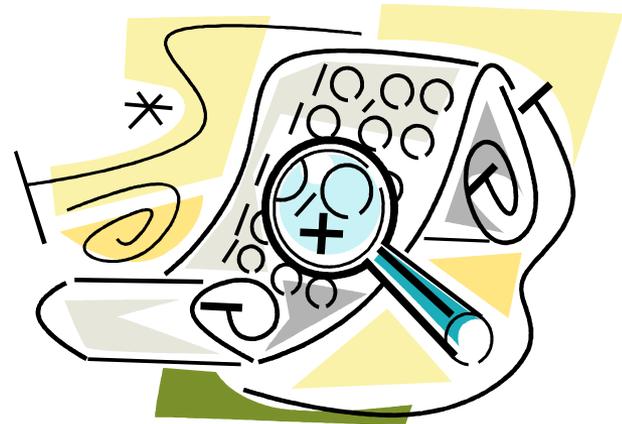
Demos with Customers

- Iteration reviews
 - Builds confidence
 - Quick feedback loop
- Informal demos
 - Pair exploratory testing with customer
 - Even on unfinished work



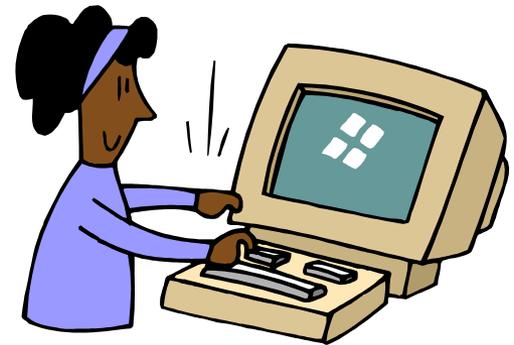
Exploratory Testing

- Simultaneous learning, test design, test execution [source: James Bach]
 - “Doing” reveals more than “thinking”
- Careful observation
- Critical thinking
- Diverse ideas
- Rich resources
 - Tools, data, people [source: Jon Hagar]



Other Types of Testing

- **Scenario testing**
 - Process flows
 - Realistic data
 - Soap opera testing (Hans Buwalda)
- **Usability testing**
 - Personas
 - Navigation
 - Observing users
- **Don't forget documents, reports, help text**



Behind the GUI

- **API testing**
 - Inputs and outputs
 - Sequence of API calls
 - Checking log files
 - Example: Test parsing of upload file
 - Example: Test shipping cost calculation
 - States and transitions
- **Web Services**
 - External customers
 - Levels of service
 - Validate definitions against profiles
 - Validate requests and responses



Feedback to Tests that Support Team

- Discuss with technical, customer team
- Turn what you learn into tests that drive new features
- Change process as needed



Who Does Quadrant 3 Tests, When

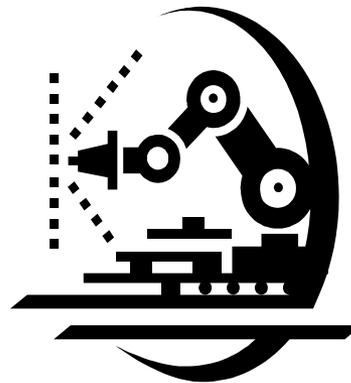
- Requires good skills, experience, intuition, critical thinking
- Involve the customers
- Programmers help with tools to facilitate
- Do as early as possible



Quadrant Three Toolkit

Tool Strategy

- Consider who uses tests, who writes and maintains tests
- Quadrant two tools may apply
- Take time to research, experiment



Tools to Critique the GUI

- Record/playback
 - Traditional vendor tools
 - Some open source tools have recorders
- Open source libraries for web browsers
 - Watir is Ruby library for testing thru browser
- Specifying tests
 - Canoo WebTests specified in XML

```
<setInputField description="set query" name="q" value="Lisa Crispin"/>  
<clickButton description="submit query" label="Google Search"/>  
<verifyText description="check for result" text="Agile Tester" />
```



Tools for Functional Testing

- **Keyword and data driven tools**
 - Share test automation with manual testers
 - Can reduce maintenance
 - Test spreadsheets with action words and data
 - Feed into automated frameworks
- **Record/playback tools**
 - Tests tend to be brittle
 - Record tool can help get started



Example Action Word Test

ScriptID	Logging	Environment	Site	Lang	Email
8	ON	STAGING	Global	English	ON

Test ID	Description	Class Name	Action	Inp ut 1	Inp ut 2
# Signup Cus tomer					
123	Cdn customer	Member	Signup	Janet	Gregory
123	Reg complete	Member	signoff		TRUE
123	Log out	Member	Log_out		TRUE

Perform CC Deposit

Setup	Description	Class Name	Action	Inp ut 1	Inp ut 2
234	Log in mbr	Member	Login	get.AcctId	get.ID_greg
234	Submit CC Txn	Member	CCDe posit	VISA	4444333322
234	Member Logout	Member	log_out		

END



Tools for Exploratory Testing

- Test scenario setup
 - eg. Watir/Watij scripts
- Generate test data
 - eg. PerlClip, Ruby script
- Simulators
 - Simulate data, feed to app over time
- Monitors
 - Watch log files
- Emulators
 - Duplicate system behavior
 - eg. mobile devices



Tools for Testing Web Services

- CrossCheck
- Ruby test::unit
- SoapUI
 - Open source version available
 - Developer-friendly
 - Expandable with Groovy



Tools for Testing API

- **Fit/FitNesse**
 - Take the place of UI
 - Use to set up data
 - Test different inputs faster
- **xUnit frameworks**
 - Works well for developers

Data Fixture	all_fund			
ticker	name	share_price	class_id	addRow?
BOGUS	Fund ABC	15.786	6	true
BOGEY	Fund DEF	2.413	2	true
BENNY	Fund BEN	1	2	true
WHAAT	Fund What	27.4	3	true
WHYYY	Fund Why	1	6	true
WHERE	Fund Where	1.431	1	true



FitNesse Example

GENERATE STATEMENTS

Generate one month at a time.

Statement Fixture		
planId	rangeld	operate?
33029	1068	true

Statement Fixture		
planId	rangeld	operate?
33029	1069	true

Statement Fixture		
planId	rangeld	operate?
33029	1070	true

Generate an account statement:

Statement Generation Fixture					
productId	planId	userId	startRangeld	endRangeld	generate?
2	33029	3756	1068	1070	true

Generate a trust statement:

Statement Generation Fixture				
productId	planId	startRangeld	endRangeld	generateTrustStatement?
2	33029	1068	1070	true



Questions?



Exercise

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

Team up with the other people at your table.

Identify some personas to use to do exploratory testing on this story. What scenarios might you try as the different personas?

What types of tools might assist with exploratory testing on this story?



Quadrant Four

Technology-facing tests that critique the product

- “ility” testing
- Performance, scalability, stress, load
- Memory management
- Data migration
- Recovery
- Who does them, when
- Toolkit



“ility” testing

- **Security**
 - Authentication, integrity, confidentiality
- **Maintainability**
 - Code reviews, inspections
 - Standards
 - Shared code ownership
- **Interoperability**
 - Diverse systems work together
 - Integration testing



More “ilities”

- **Compatibility**
 - Browsers, operating systems
 - Third-party apps
- **Reliability**
 - Mean time to failure
 - Mean time between failures
- **Installability**
 - Repeatability
 - Test early and often



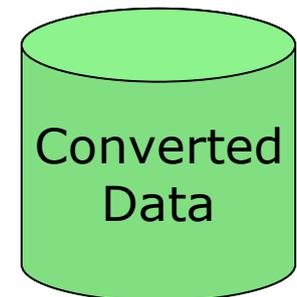
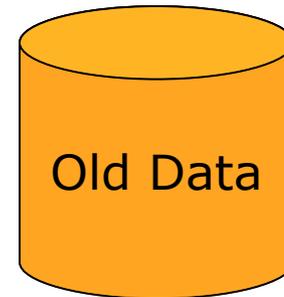
Quadrant Four Tests

- Performance
 - How fast? Identify bottlenecks
- Stability
 - How long?
- Reliability
 - How often?
- Scalability
 - How much?



More Quadrant Four Tests

- Memory management
 - Issues such as leaks
- Data migration
 - Conversion scripts
- Recovery
 - Failover testing
- Test environments
 - Independent, production-like
- Baselines
- Write stories for these types of tests



Who Does Quadrant 4 Tests, When

- Depends on priorities
- May need from start
- May need to test scalability early
- It pays to get a baseline
- Programmers can write multiple-thread harnesses at unit level
- Plan for specialists as needed
- Team responsibility



Quadrant Four Automation

- Write stories to evaluate tools
- Native database tools
 - SQL, data import tools
- Shell scripting
- Monitoring tools
 - jConsole
 - Application bottlenecks, memory leaks
 - jProfiler
 - Database and bean usage



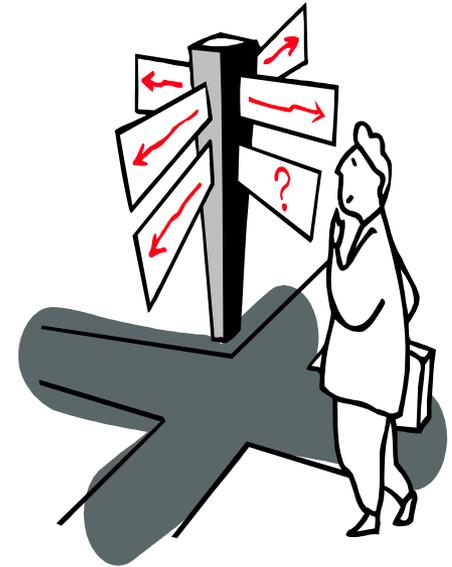
More Quadrant Four Tools

- Commercial load test tools
 - Loadrunner
 - Silk Performer
- Open source test tools
 - jMeter
 - The Grinder
 - jUnitPerf
- Performance test providers
 - Multiple sites



Planning Your Test Strategy

- Scope
- Priorities, Risks
- Tools that solve the problem
- Customers
- Document only what is useful
- Consider all four quadrants
- Use lessons learned to improve



Questions?



Group Exercise

Team up with others at your table to discuss the following.

In which quadrant is your team “back home” lacking the most?

What will you do to address this, when you go back?



Some Agile Testing Resources

- lisa.crispin.home.att.net
- www.agilealliance.org
- www.testing.com
- agile-testing@yahooogroups.com
- www.fitnessse.org
- webtest.canoo.com
- fit.c2.com

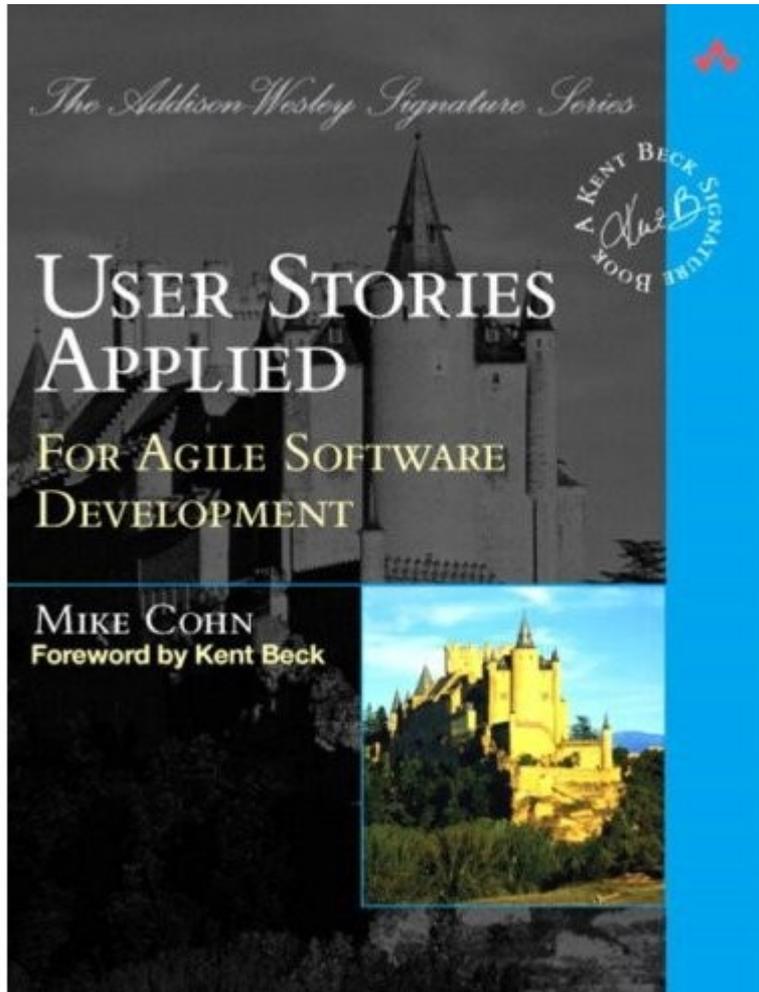


Exploratory Testing Resources

- 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>
- <http://www.satisfice.com/articles/sbtm.pdf>



Agile Resources

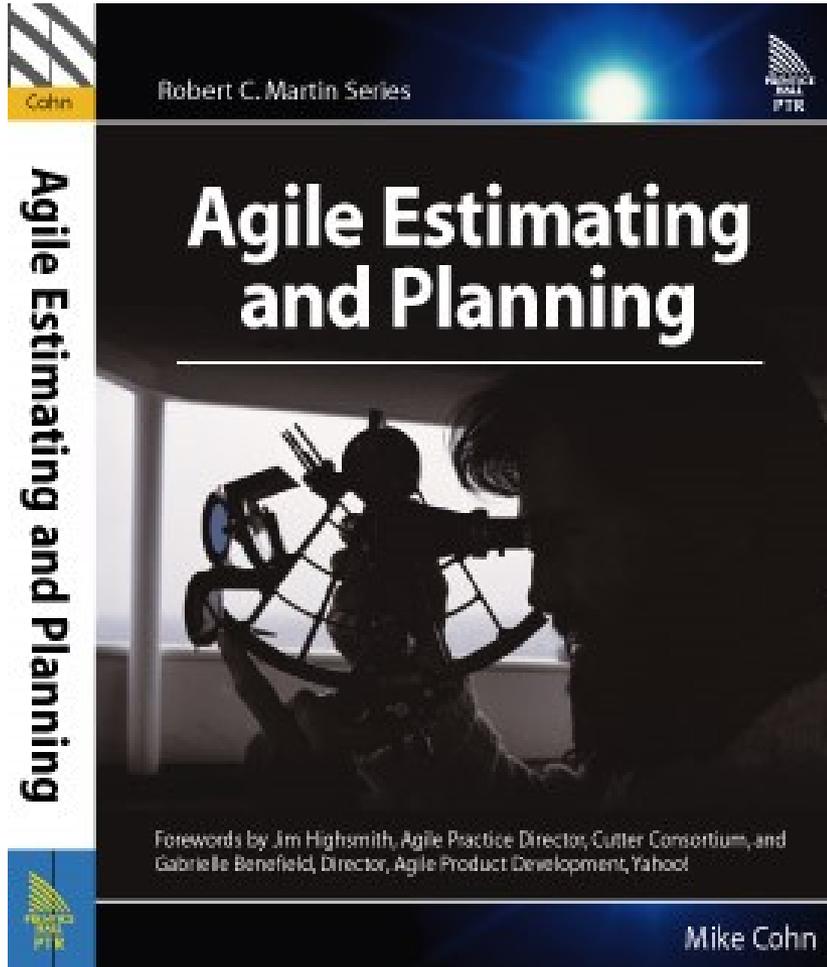


User Stories Applied

by Mike Cohn



Agile Resources



Agile Estimating and Planning

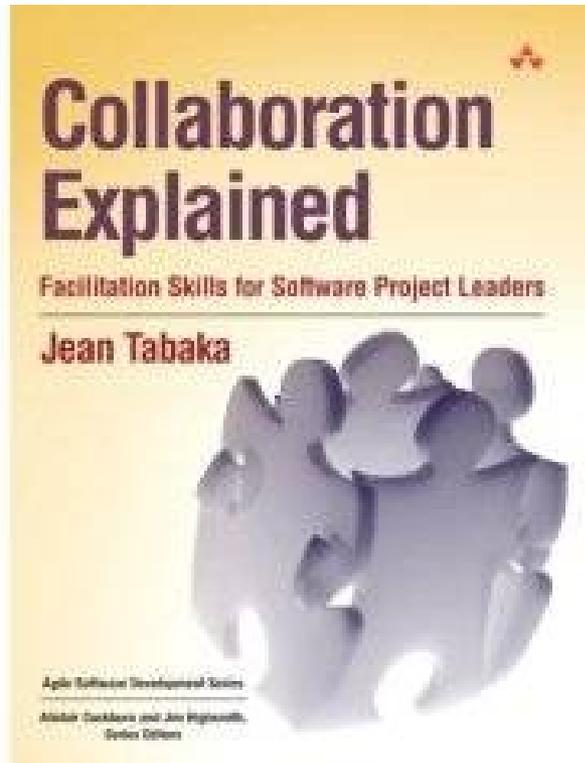
By Mike Cohn



Copyright 2008: Lisa Crispin



Collaboration



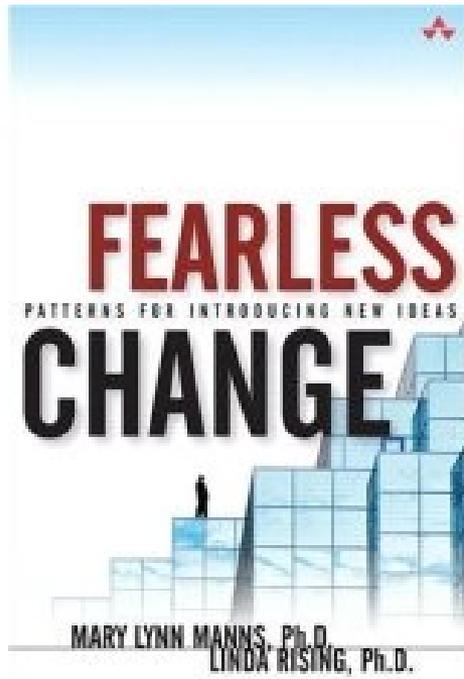
*Collaboration Explained :
Facilitation Skills for
Software Project Leaders*

By Jean Tabaka

Available on Amazon



Implementing Change



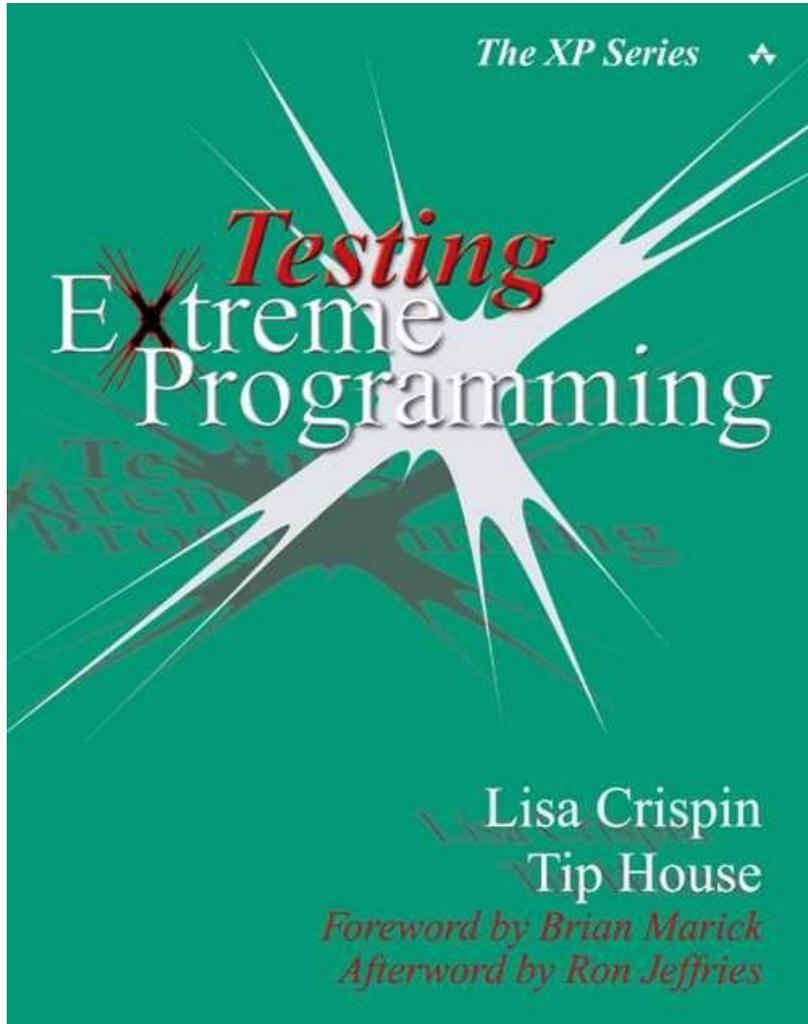
Fearless Change: Patterns for introducing new ideas

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



Copyright 2008: Lisa Crispin



Goal

Have fun, whatever you do!

lisa.crispin@gmail.com

lisacrispin.blogspot.com



Copyright 2008: Lisa Crispin



Questions?

