


# Focus Your Testing

## Using the Agile Testing Matrix



Agile Edmonton  
February 4, 2009

Janet Gregory  
DragonFire Inc.

# Agenda

---

- A Bit About Testing
- Introduction to the Agile Testing Quadrants
  - Quadrant 1
  - Quadrant 2
  - Quadrant 3
  - Quadrant 4
- Planning your Testing Strategy
- References



# Why Do We Test?

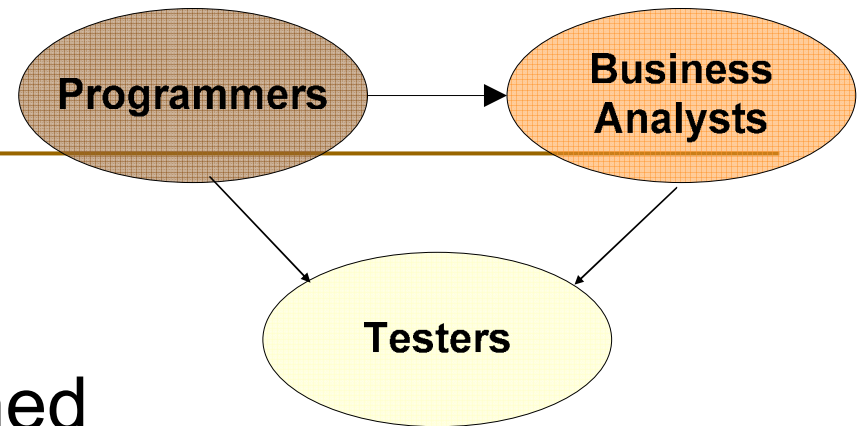
---

- To find bugs
- Make sure system is reliable
- To learn about the application
- See if UI is usable
- Feedback to future stories
- Check for doneness
- Manage technical debt
  - deferred work
  - hacks, untidy work
  - Slows the team down



# Traditional Testing

---



## Phased / Gated Projects

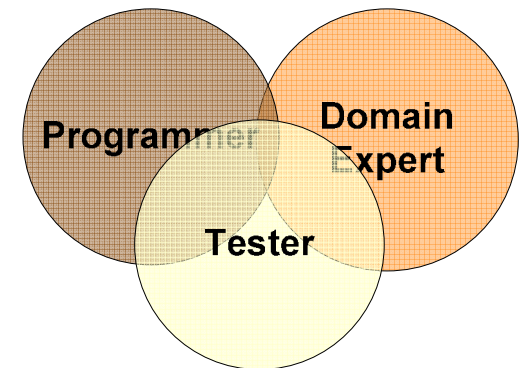
- Done after coding is finished
- Works from requirements documents
- Can be a fight to be involved up front
- Organizational culture – silos
- Communicates through DTS
- Few face to face conversations
- Different language than developers



# Agile Testing

---

- Principles / Values
  - Communication, collaboration, interactions
- Teams are test infected
- Testers are part of the team
- Whole team responsible for quality
- Tester's role
  - Help uncover hidden assumptions
  - Provide feedback
  - Elicit and clarify requirements
  - Drive development with examples



# Agile Means a Change in Mindset

---

Focus on the 'why' we test

Change the way we think

- not about the when
- not about the how (ex. white box, black box)

Instead of

- We're here to break the software!

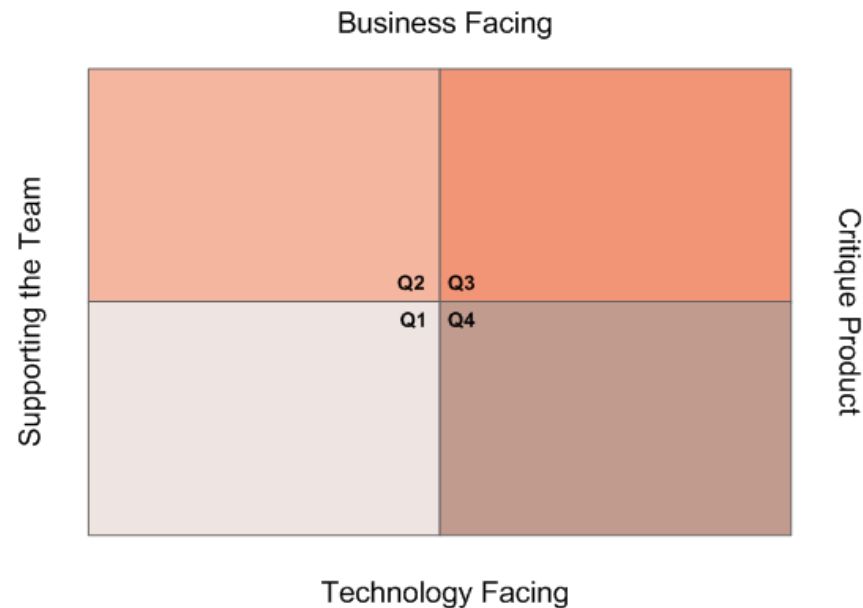
Think

- What can we do to help deliver the software successfully?



# The Agile Testing Quadrants

- Can be used to ensure we accomplish all goals
  - Q1 - Technology-facing tests that support the team
  - Q2 - Business-facing tests that support the team
  - Q3 - Business-facing tests that critique the product
  - Q4 - Technology-facing tests that critique the product



# The Quadrants .....

---

- Can be used as a communication tool
  - To the project team
  - To management
  - To explain testing in a common language
- Emphasize whole-team responsibility
  - Focus on collaboration
  - Whole team participation



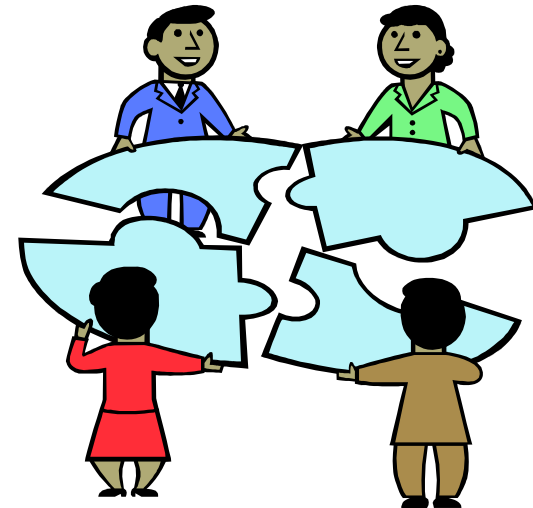


# Use to define 'Doneness'

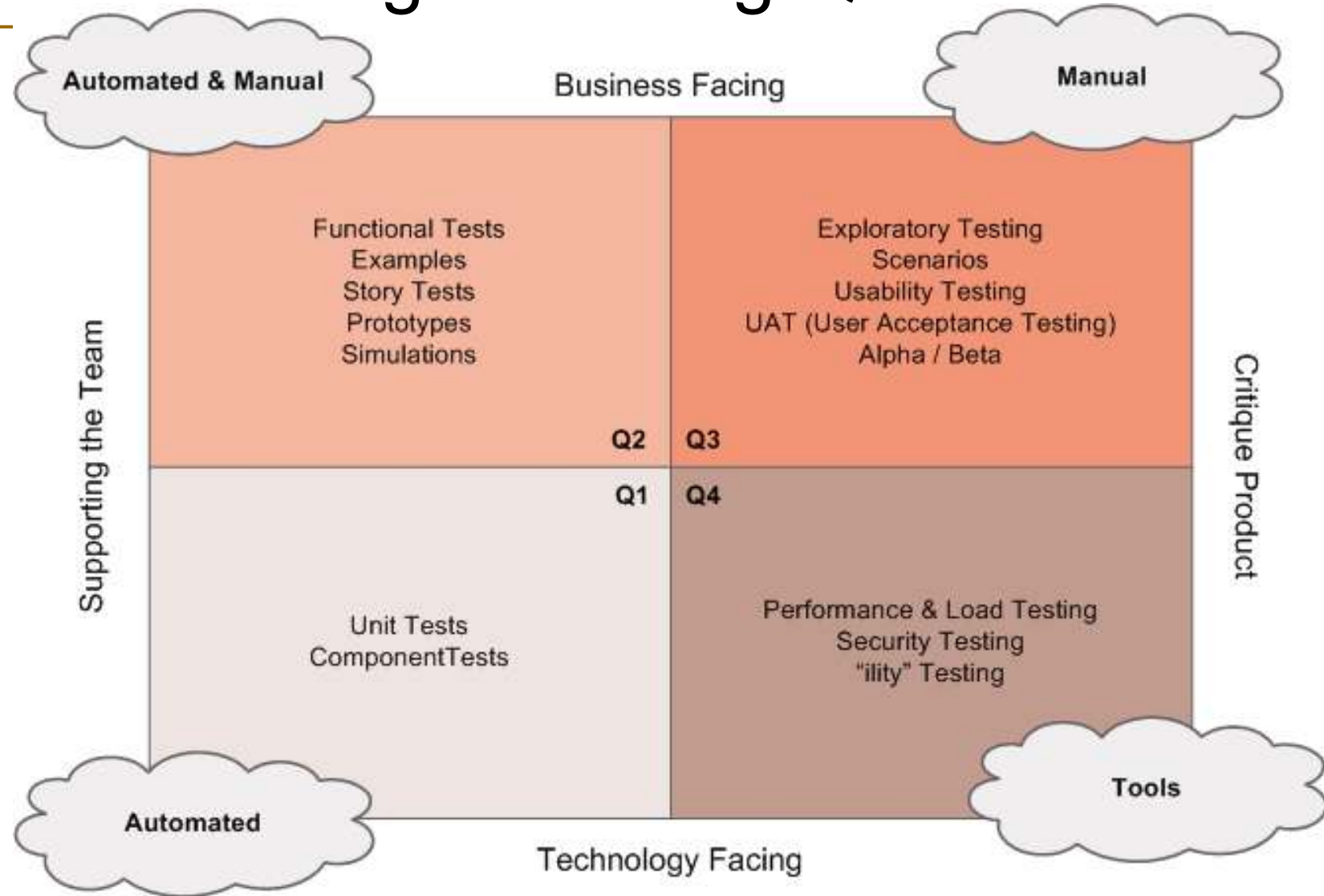
---

## Defining “doneness” for release readiness

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



# The Agile Testing Quadrants



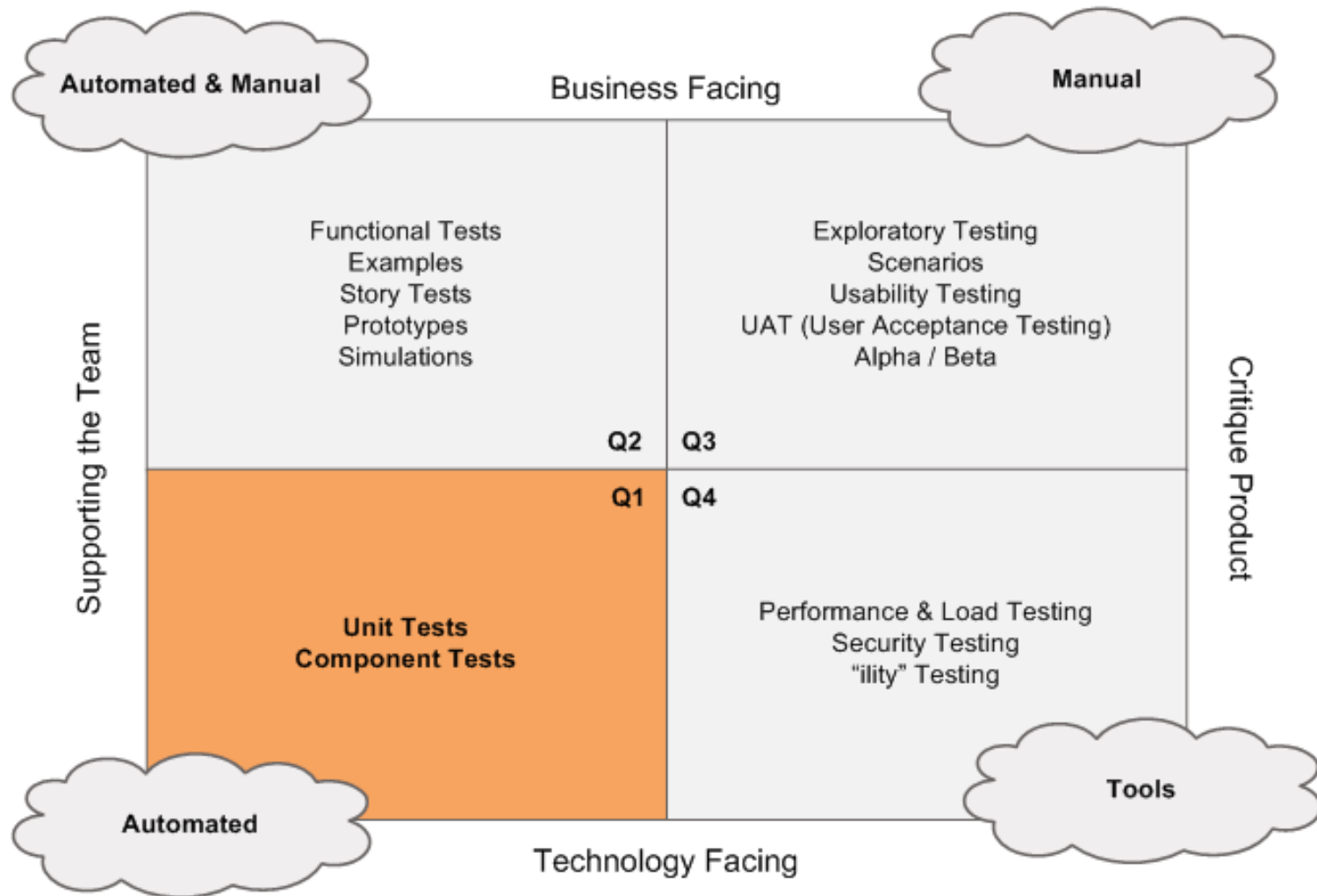
Original idea by Brian Marick, [www.exampler.com](http://www.exampler.com)

Copyright 2009 Janet Gregory, DragonFire



# Quadrant 1

## Agile Testing Quadrants



# Q1 Kinds of Tests

---

- Unit Tests

- Tests developer intent - program design
- Tests a small piece of code
- Makes sure it does what it should

- Component Tests

- Tests architect intent – system design
- Tests that components work together correctly

Technology-facing Tests that Support the Team



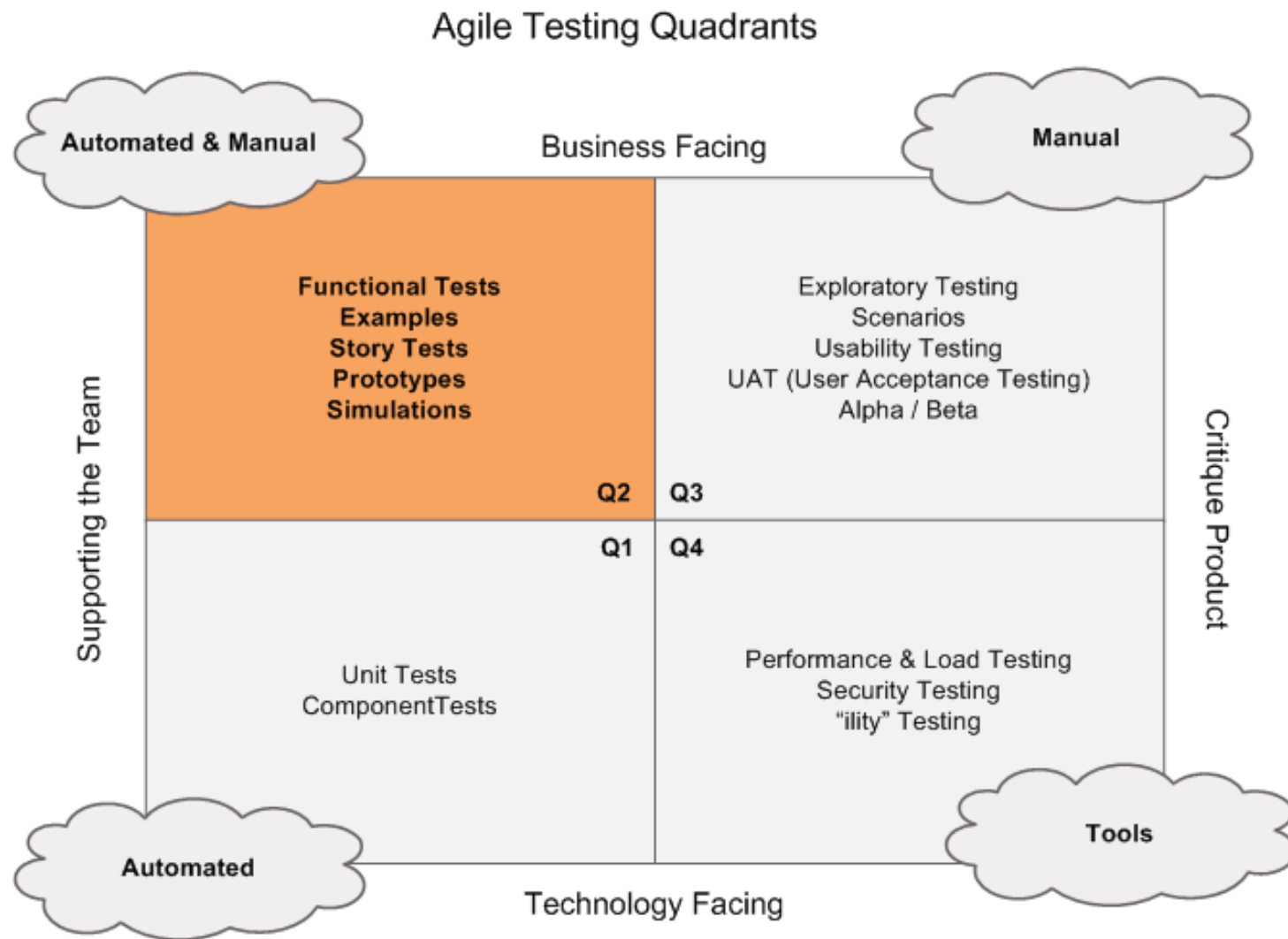
# Reasons / Benefits for Q1 Testing

---

- Focus on internal code quality
- Go faster, do more
  - Unit tests provides safety net and refactoring support
- Builds quality in
- Provides instant feedback
- TDD increases confidence in design
- Forms the foundation of automation suite
- Builds testability into code



# Quadrant 2



## Q2: Kinds of Tests

---

- Elicit requirements
- SDD or ATDD (Acceptance Test Driven Dev)
  - allows developers to code until the tests pass
  - Fit / Fittesse
- Examples
- User Experience
  - wire frames
  - Mock-ups / prototypes
- Pair Testing

Business-facing Tests that Support the Team



# Reasons / Benefits for Q2 Testing

---

- Drive development with business-facing tests
- Obtain enough requirements to start coding
- Help customers achieve advance clarity
- Capture examples, express as executable tests
- Focus is external quality
- Know when we're done
- Customer – developer – tester collaboration

Business-facing Tests that Support the Team





# Toolkit – Turning Examples into Tests

Set-up: Create a variety of users that start with the first initial, have the same name, similar email address, and unique ids.

<b>create user with login id</b>	frankb	<b>first name</b>	Frank	<b>last name</b>	Billian	<b>email address</b>	frankb@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Quality Assurance
<b>create user with login id</b>	junebug	<b>first name</b>	Juno	<b>last name</b>	Williams	<b>email address</b>	junebug@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Quality Assurance
<b>create user with login id</b>	fillpot	<b>first name</b>	Darien	<b>last name</b>	Fillpot	<b>email address</b>	fillpot_darien@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Quality Assurance
<b>create user with login id</b>	joneero	<b>first name</b>	Jone	<b>last name</b>	Roberts	<b>email address</b>	Jone.Roberts@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Operations
<b>create user with login id</b>	billboa	<b>first name</b>	Bill	<b>last name</b>	Bia	<b>email address</b>	bill_bia@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Employee
<b>create user with login id</b>	june	<b>first name</b>	Juno	<b>last name</b>	Will	<b>email address</b>	junebug1@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Employee

Query users - return all: no criteria set (sort by login id)

<b>query users with login id</b>	<b>first name</b>		<b>last name</b>	<b>email address</b>	<b>phone number</b>	<b>roles</b>
<i>first name</i>	<i>last name</i>	<i>login id</i>	<i>email address</i>	<i>phone number</i>	<i>roles</i>	
Idi	Administrator	administrator	idi_administrator@ngx.com	403-974 4957	Employee,Operations,Production Support,Quality Assurance	
Bill	Bia	billboa	bill_bia@dragonfire.xx		Employee	
Darien	Fillpot	fillpot	fillpot_darien@dragonfire.xx		Quality Assurance	
Frank	Billian	frankb	frankb@dragonfire.xx		Quality Assurance	
Jone	Roberts	joneero	Jone.Roberts@dragonfire.xx		Operations	
Juno	Will	june	junebug1@dragonfire.xx		Employee	
Juno	Williams	junebug	junebug@dragonfire.xx		Quality Assurance	

Query users based on first name

<b>query users with login id</b>	<b>first name</b>		<b>last name</b>	<b>email address</b>	<b>phone number</b>	<b>roles</b>
<i>first name</i>	<i>last name</i>	<i>login id</i>	<i>email address</i>	<i>phone number</i>	<i>roles</i>	
Jone	Roberts	joneero	Jone.Roberts@dragonfire.xx		Operations	
Juno	Will	june	junebug1@dragonfire.xx		Quality Assurance <i>expected</i>	
Juno	Will	june	junebug1@dragonfire.xx		Employee <i>actual</i>	
Juno	Williams	junebug	junebug@dragonfire.xx		Quality Assurance	

Query users based on last name

<b>query users with login id</b>	<b>first name</b>		<b>last name</b>	<b>email address</b>	<b>phone number</b>	<b>roles</b>
<i>first name</i>	<i>last name</i>	<i>login id</i>	<i>email address</i>	<i>phone number</i>	<i>roles</i>	
Bill	Bia	billboa	bill_bia@dragonfire.xx		Employee	
Frank	Billian	frankb	frankb@dragonfire.xx		Quality Assurance	

Query users based on email

<b>query users with login id</b>	<b>first name</b>		<b>last name</b>	<b>email address</b>	<b>phone number</b>	<b>roles</b>
<i>first name</i>	<i>last name</i>	<i>login id</i>	<i>email address</i>	<i>phone number</i>	<i>roles</i>	
Darien	Fillpot	fillpot	fillpot_darien@dragonfire.xx		Quality Assurance	



Set-up: Create a variety of users that start with the first initial, have the same name, similar email address, and unique ids.

<b>create user with login id</b>	frankb	<b>first name</b>	Frank	<b>last name</b>	Billian	<b>email address</b>	frankb@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Quality Assurance
<b>create user with login id</b>	junebug	<b>first name</b>	Juno	<b>last name</b>	Williams	<b>email address</b>	junebug@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Quality Assurance
<b>create user with login id</b>	fillpot	<b>first name</b>	Darien	<b>last name</b>	Fillpot	<b>email address</b>	fillpot_darien@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Quality Assurance
<b>create user with login id</b>	jonero	<b>first name</b>	Jone	<b>last name</b>	Roberts	<b>email address</b>	Jone.Roberts@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Operations
<b>create user with login id</b>	billboa	<b>first name</b>	Bill	<b>last name</b>	Bia	<b>email address</b>	bill_bia@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Employee
<b>create user with login id</b>	june	<b>first name</b>	Juno	<b>last name</b>	Will	<b>email address</b>	junebug1@dragonfire.xx	<b>phone number</b>		<b>roles</b>	Employee

Query users - return all: no criteria set (sort by login id)

<b>query users with login id</b>		<b>first name</b>		<b>last name</b>		<b>email address</b>		<b>phone number</b>		<b>roles</b>
<i>first name</i>	<i>last name</i>	<i>login id</i>		<i>email address</i>		<i>phone number</i>		<i>roles</i>		
Idi	Administrator	administrator		idi_administrator@ngx.com		403-974 4957		Employee,Operations,Production Support,Quality Assurance		
Bill	Bia	billboa		bill_bia@dragonfire.xx				Employee		
Darien	Fillpot	fillpot		fillpot_darien@dragonfire.xx				Quality Assurance		
Frank	Billian	frankb		frankb@dragonfire.xx				Quality Assurance		
Jone	Roberts	jonero		Jone.Roberts@dragonfire.xx				Operations		
Juno	Will	june		junebug1@dragonfire.xx				Employee		
Juno	Williams	junebug		junebug@dragonfire.xx				Quality Assurance		

Query users based on first name

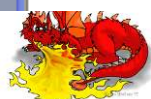
<b>query users with login id</b>		<b>first name</b>	J	<b>last name</b>		<b>email address</b>		<b>phone number</b>		<b>roles</b>
<i>first name</i>	<i>last name</i>	<i>login id</i>		<i>email address</i>		<i>phone number</i>		<i>roles</i>		
Jone	Roberts	jonero		Jone.Roberts@dragonfire.xx				Operations		
Juno	Will	june		junebug1@dragonfire.xx				Employee		
Juno	Williams	junebug		junebug@dragonfire.xx				Quality Assurance		

Query users based on last name

<b>query users with login id</b>		<b>first name</b>		<b>last name</b>	Bi	<b>email address</b>		<b>phone number</b>		<b>roles</b>
<i>first name</i>	<i>last name</i>	<i>login id</i>		<i>email address</i>		<i>phone number</i>		<i>roles</i>		
Bill	Bia	billboa		bill_bia@dragonfire.xx				Employee		
Frank	Billian	frankb		frankb@dragonfire.xx				Quality Assurance		

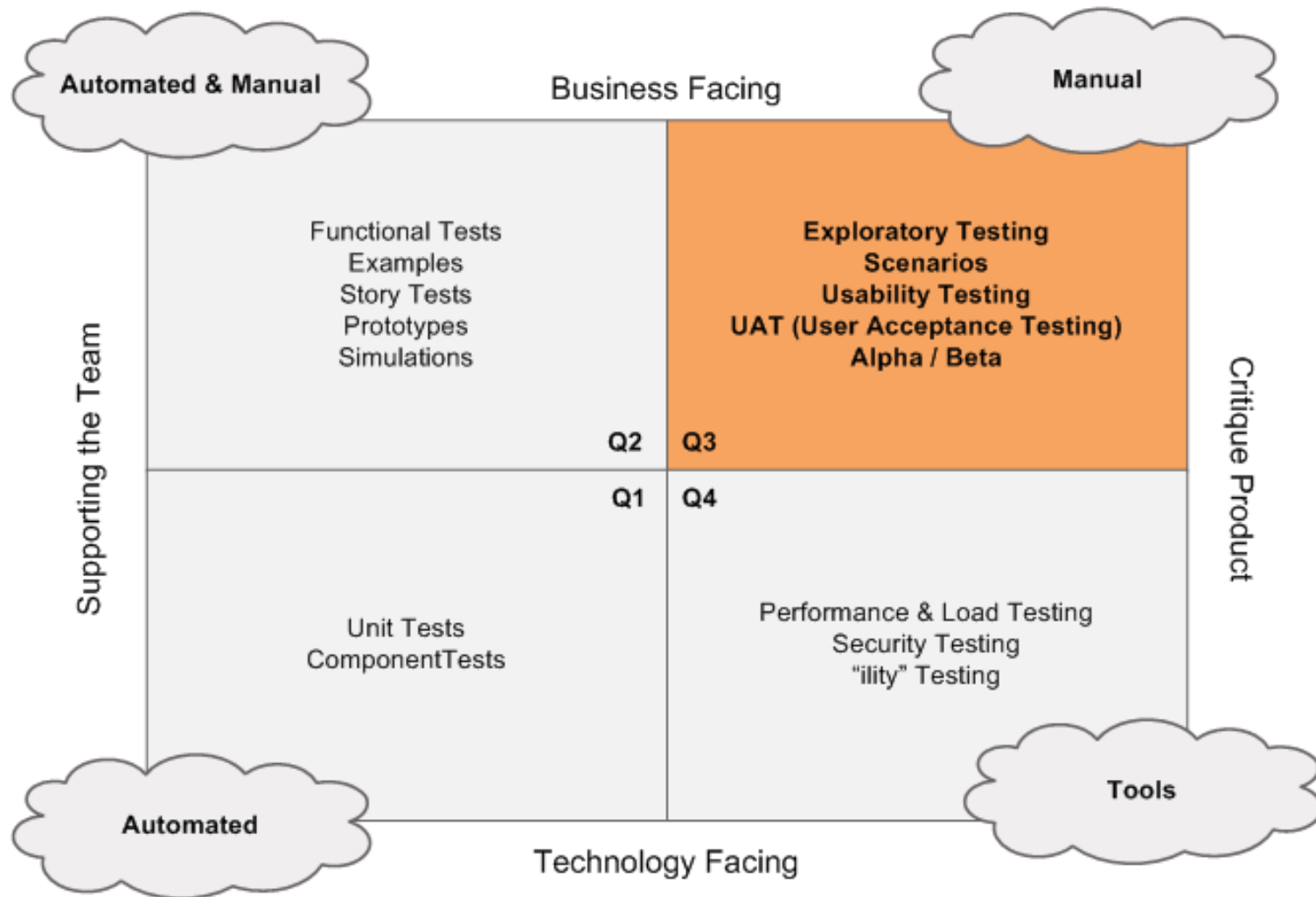
Query users based on email

<b>query users with login id</b>		<b>first name</b>		<b>last name</b>		<b>email address</b>	fillpot_	<b>phone number</b>		<b>roles</b>
<i>first name</i>	<i>last name</i>	<i>login id</i>		<i>email address</i>		<i>phone number</i>		<i>roles</i>		
Darien	Fillpot	fillpot		fillpot_darien@dragonfire.xx				Quality Assurance		



# Quadrant 3

Agile Testing Quadrants



# Q3: Kinds of Testing

---

- Exploratory Testing
  - feedback into stories
  - work with customers to understand what you think
  - work with developers if you have questions
- Test for Usability
  - understand end users - personas
  - who will be using the system
- Pair test with customers
- User Acceptance Testing

Business-facing Tests that Critique the Product



# Q3: Collaborative Testing

---

- Provide feedback ....
  - Discuss with technical, customer team
  - Turn what you learn into tests that drive new features
  - Change process as needed
- Iteration reviews
  - Builds confidence
  - Quick feedback loop
- Informal demos
  - Pair exploratory testing with customer
  - Even on unfinished code

Business-facing Tests that Critique the Product



# Reasons / Benefits for Q3 Testing

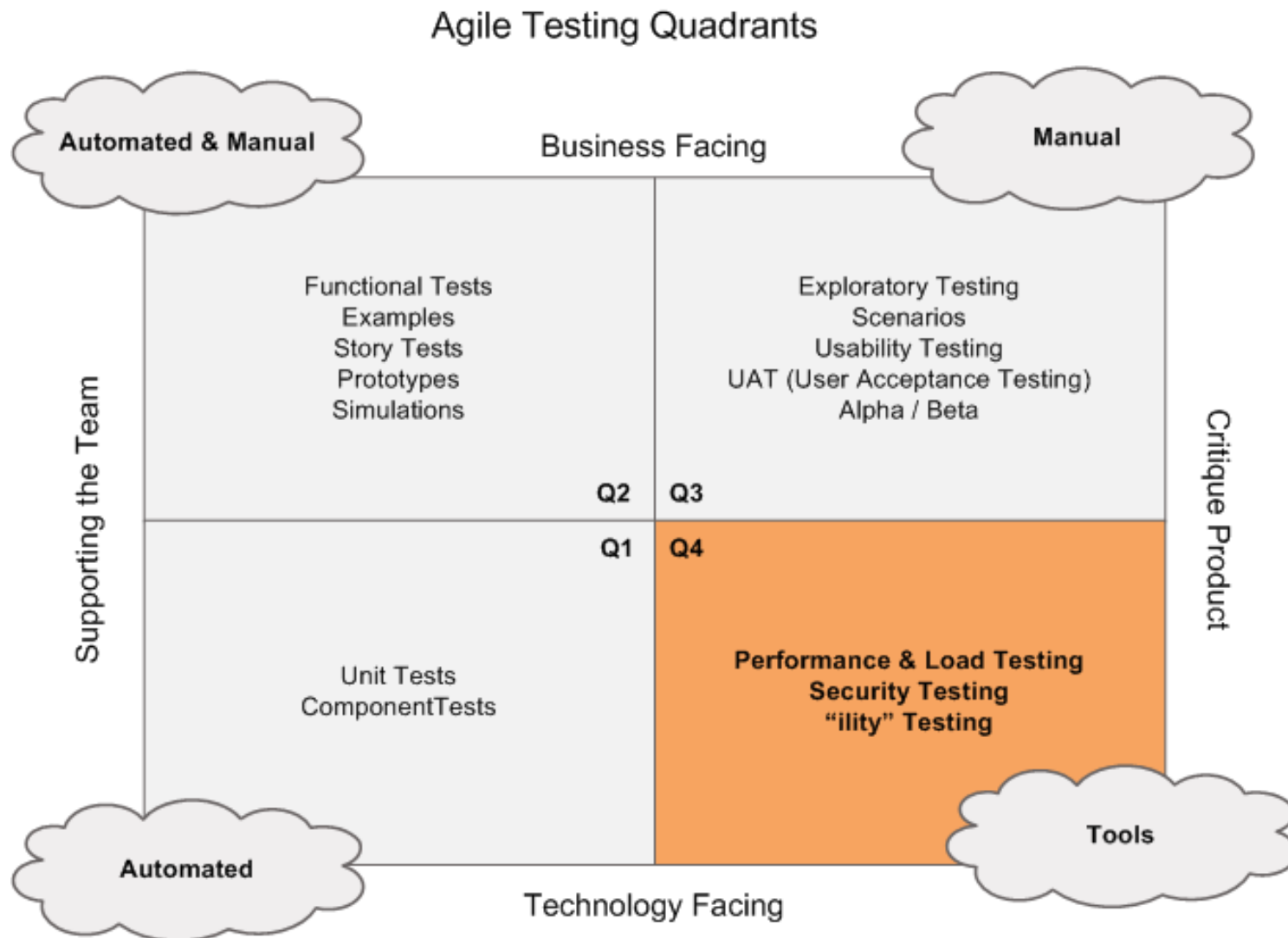
---

- Feedback to Quadrants One and Two
- Evaluate 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

Business-facing Tests that Critique the Product



# Quadrant 4



## Q4: Kinds of Tests

---

- Nonfunctional or parafunctional tests
- “ility” testing
- Performance, scalability, stress, load
- Memory management
- Security testing
  - Roles & permissions
  - System ‘hacking’
- Data migration
- Infrastructure Testing
- Recovery

Technology-facing Tests that Critique the Product





# Reasons / Benefits for Q4 Testing

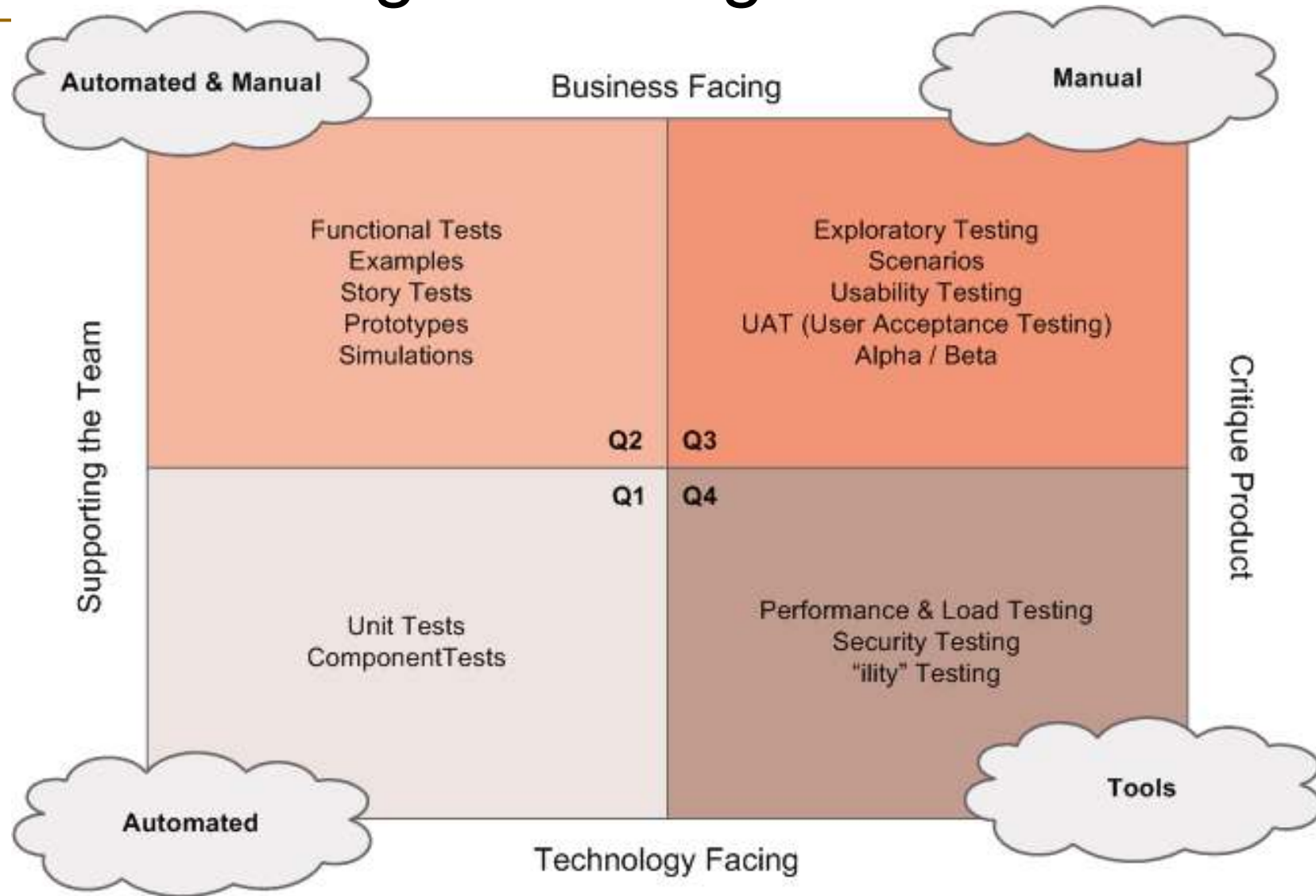
---

- “Non-functional” requirements may be higher priority than “functional”
  - Performance
  - Stability
  - Security
- Specialized expertise might be needed
  - Collaborate to transfer skills
- Makes the ‘finished’ product
- Does your application deliver the ‘right’ value

Technology-facing Tests that Critique the Product



# The Agile Testing Quadrants



Original idea by Brian Marick, [www.exampler.com](http://www.exampler.com)

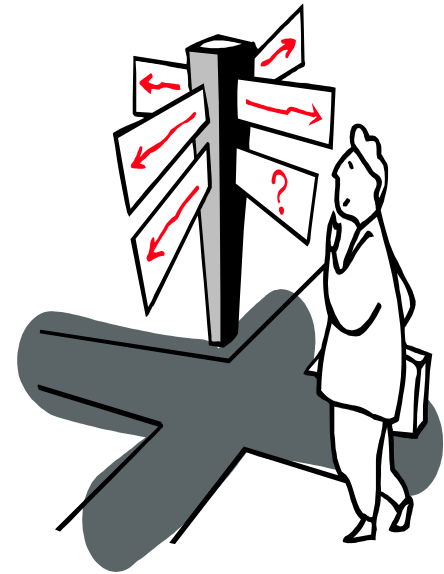
Copyright 2009 Janet Gregory, DragonFire



# Planning Your Test Strategy

---

- Consider scope, priorities, risks
- Tools that solve the problem
- Involve customers
- Collaborate with programmers
- Consider all four quadrants
- Test matrix - big picture
  - allows whole team to understand
- Document only what is useful
- Use lessons learned to improve



# Questions?

---



# Available Now!

---

## *Agile Testing: A Practical Guide for Testers and Agile Teams*

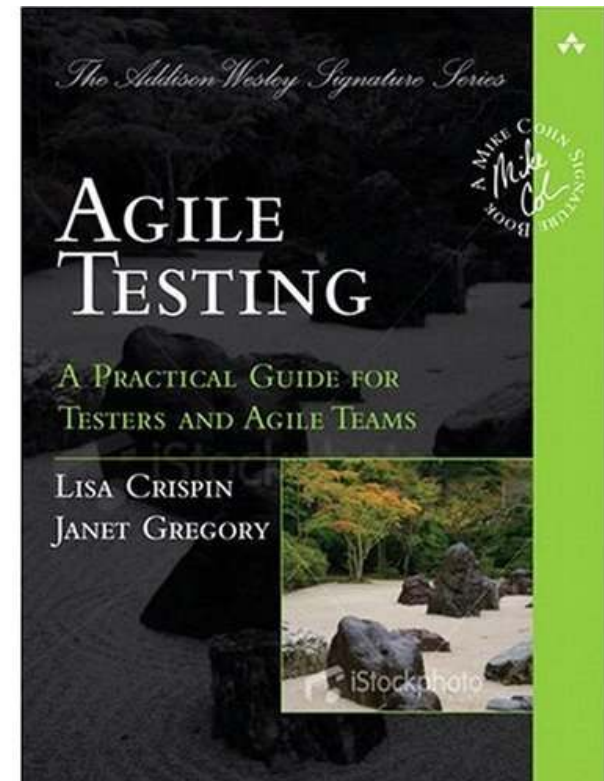
By Janet Gregory and Lisa Crispin

Available on

- Amazon.com
- Amazon.ca

[www.agiletester.ca](http://www.agiletester.ca)

[www.janetgregory.ca](http://www.janetgregory.ca)



Copyright 2008 Janet Gregory, DragonFire

# Resources

---

- [www.exampler.com](http://www.exampler.com)
- *Collaboration Explained*: Jean Tabaka
- *Testing Extreme Programming*, By Lisa Crispin and Tip House
- *Fearless Change: Patterns for introducing new ideas*, Linda Rising and Mary Lynn Manns
- [agile-testing@yahoogroups.com](mailto:agile-testing@yahoogroups.com)

