Making Test Automation Work in Agile Projects

Agile Testing Days 2010

Lisa Crispin

With Material from Janet Gregory

Introduction - Me

- Programming background
- Test automation from mid-90s
- Agile from 2000
  - Many new automation possibilities!

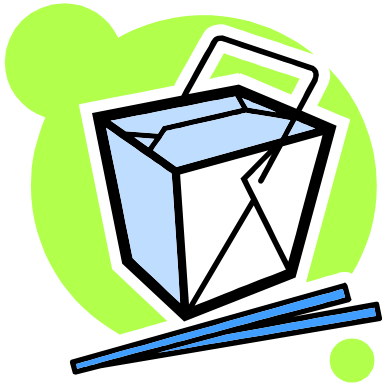
Introduction - You

- Main role on team?
- Programming, automation experience?
- Using agile approach?
- Current level of automation? (Test, CI,deployment, IDEs, SCCS...)

Takeaways

Foundation for successful test automation

- "Whole Team" approach
- When to automate
- Apply agile principles, practices
- Good test design principles
- Identifying, overcoming barriers
- Choosing, implementing tools
- First steps



We won't do any hands-on automation, but will work through some examples together

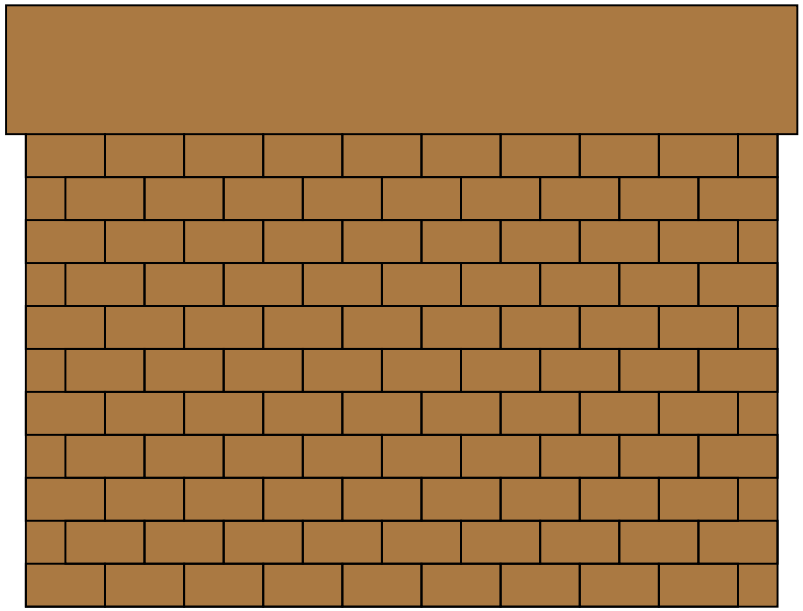
Why Automate?

- Free up time for most important work
- Repeatable
- Safety net
- Quick feedback
- Help drive coding
- Tests provide documentation

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

Barriers to Test Automation

What's holding you back?

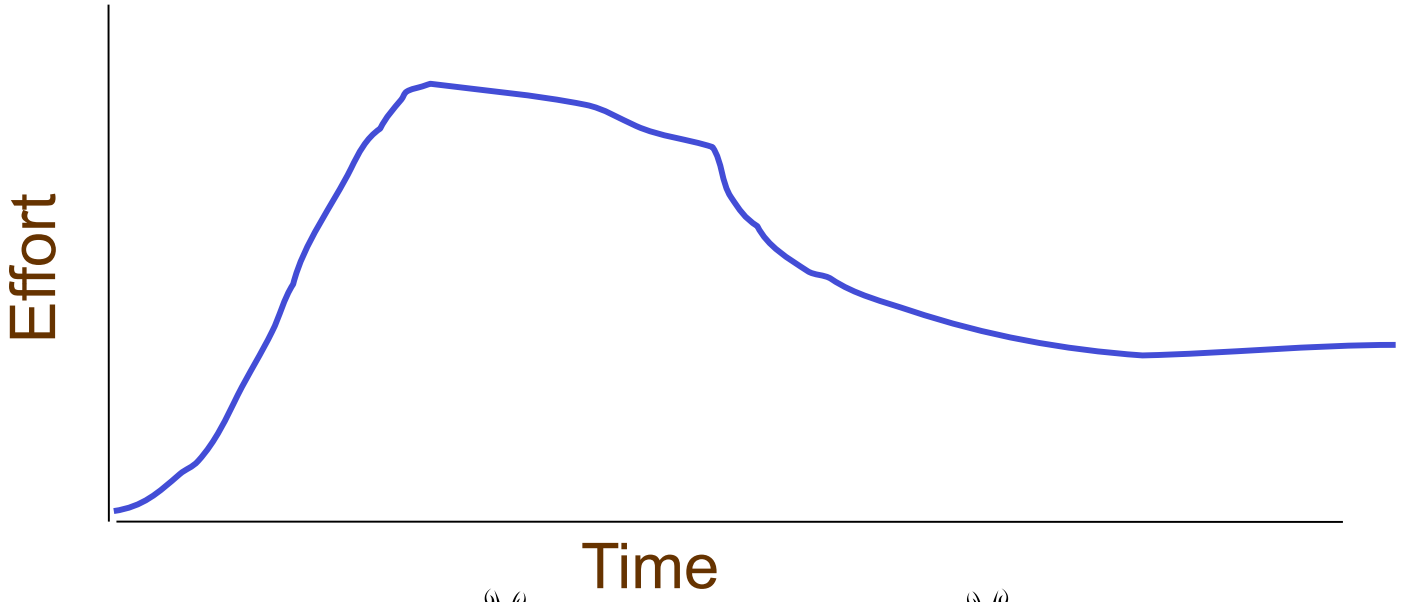


Pain and Fear

- Programmers don't feel manual test pain
- Testers treated as safety net
- Fear
  - Programmers lack testing skills
  - Testers lack programming skills

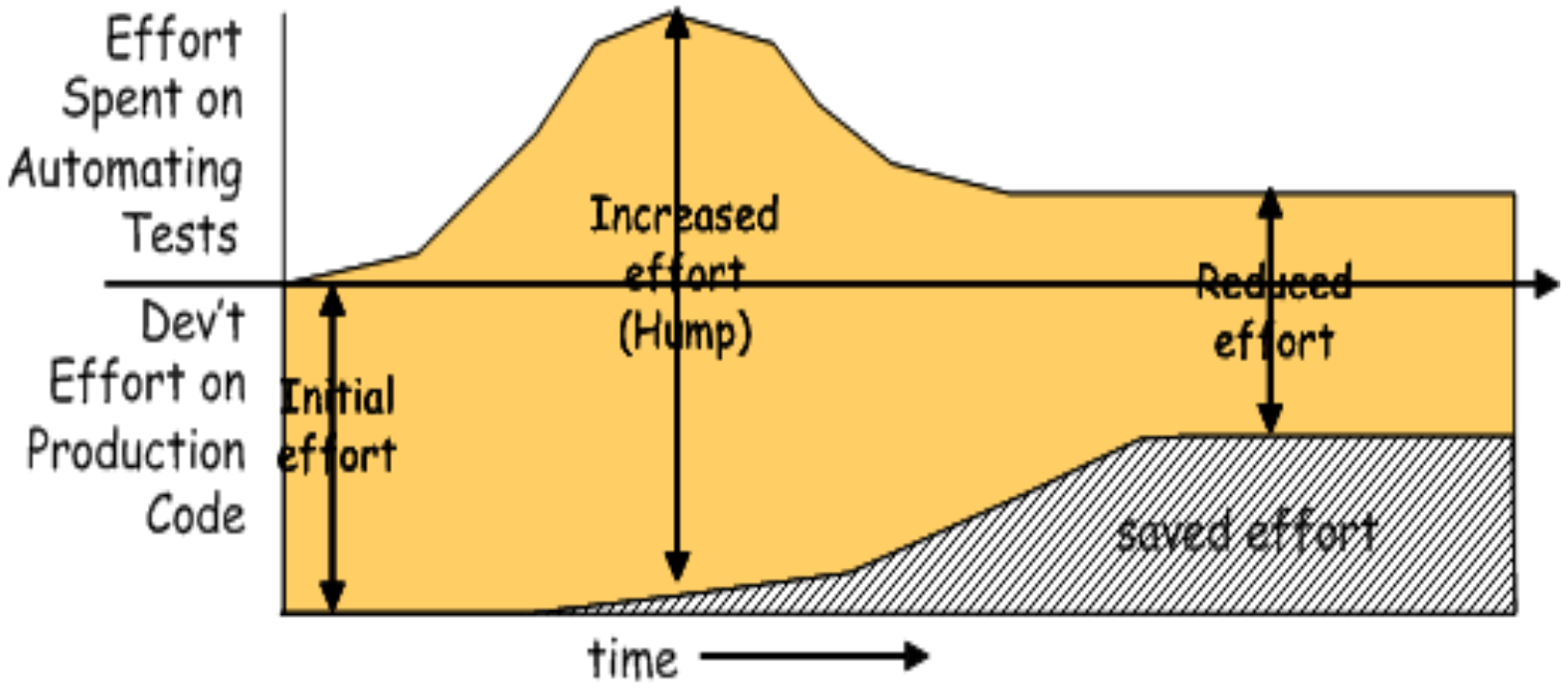
Initial Investment

- Hump of pain
- Legacy code, changing code
- Tools, infrastructure, time



It's Worth It

- ROI – explain to management
- "Present value" of automated tests
- Acknowledge hump of pain



Economics of Test Design

- Poor test practices/design = poor ROI
  - Tests had to understand, maintain
- Good test practices/design = good ROI
  - Simple, well-designed, refactored tests

Exercise – What's holding YOU back?

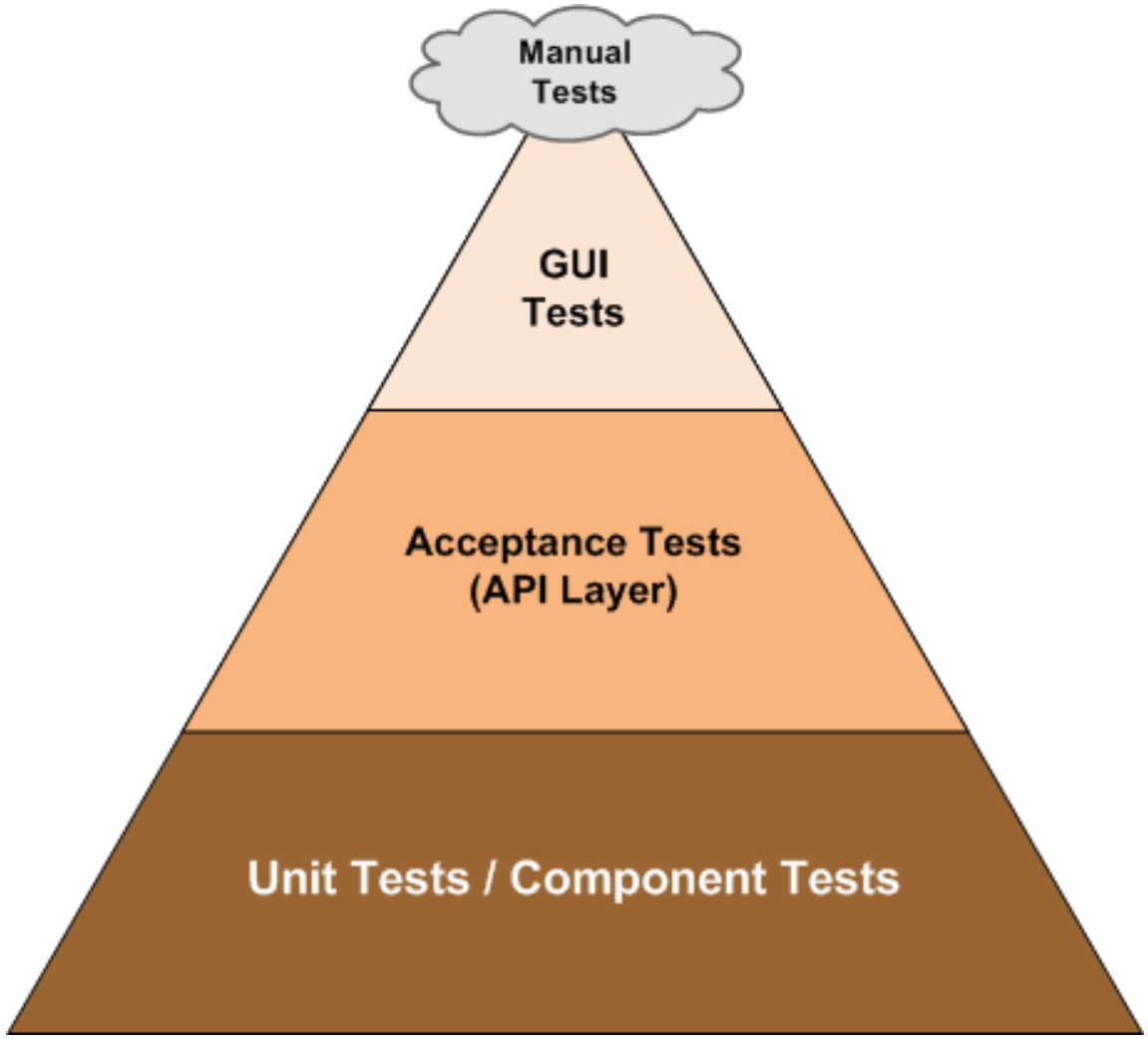
- Write obstacles hindering your team on small paper card.
- If you have the same obstacle as someone else in your group, stick a "dot" on their card.
- Post those on the big flip chart in the appropriate category – or start a new category.
- Are we seeing any patterns?
- Share with the large group

Questions?

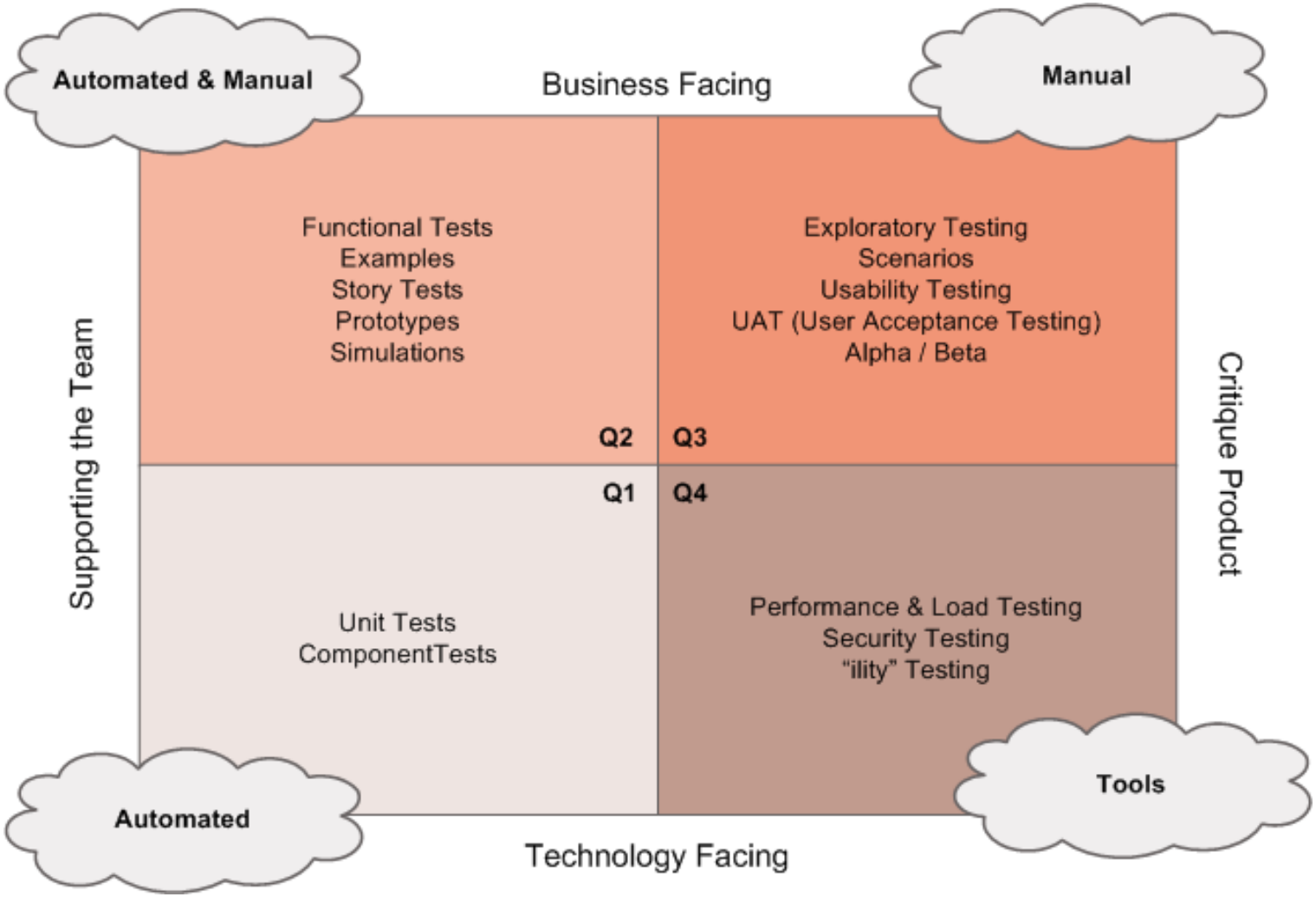
Getting Over the Hump

- The test automation pyramid
- The agile testing quadrants
- What should be automated
- What shouldn't
- Difficult areas
- Test design

Test Automation Pyramid

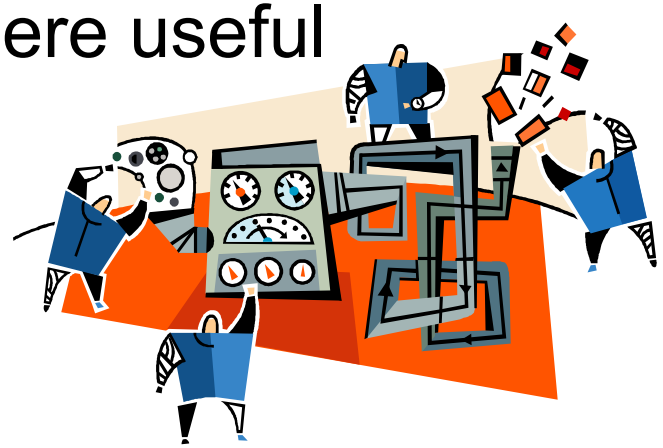


Agile Testing Quadrants



What Should We Automate?

- Quadrant 1 tests
  - Unit, component, TDD
- Quadrant 2 tests
  - Behind GUI, API, web services
- Quadrant 4 tests
  - Load, performance, stress
- Quadrant 3 tests?
  - Leverage automation where useful



What Shouldn't We Automate?

- Quadrant 2 tests
  - Wizard of Oz, prototyping
- Quadrant 3 tests
  - Usability, UAT, ET
- Tests that will never fail?
  - Assess risk
- ROI not enough
  - One-off tests

Where Should We Be Careful?

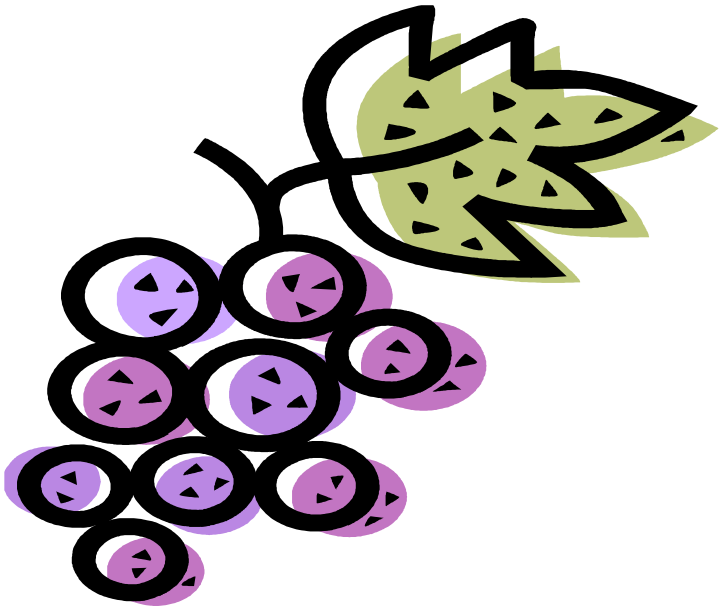
- GUI tests
  - Need to test the GUI
  - Watch ROI
- End-to-End tests
  - Push testing down to lowest level
  - Robust lower-level tests = better ROI
- Remember the Pyramid

Hard to Automate?

- Legacy code
  - Hard to automate, or just lack of skill?
  - “Working Effectively with Legacy Code” – Feathers
  - “Strangling” – Fowler, Thomas

Discussion

- Where should you start?
  - Is there “low-hanging fruit”?
  - What’s the best ROI?
  - Or the most critical area?



Agile Automation Strategy

- What hurts the most
- Layered approach
- Applying agile principles
  - Whole team approach
  - Small chunks/thin slices
  - Smart test design
- Choosing the right tools

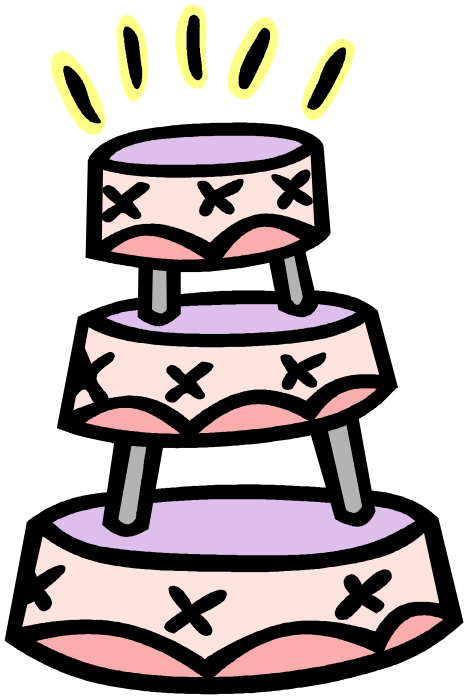
What Hurts the Most

- Keep an impediment backlog
- What's biggest obstacle?
  - Time
  - Tools
  - Code design
  - ...

Multi-Layered Approach

Example:

- Developers address unit tests
- While testers write GUI smoke tests



Whole-Team Approach

- Team responsible for testing, quality
  - Testable architecture
- Team responsible for testing activities
- Whole team has all the skills needed
  - Good code design skills essential
- Team designs for ease of test automation

Discussion

- What skills would help your team overcome automation barriers?
- What skills are missing from your team? Where could you get them?

Agile Values: Simplicity

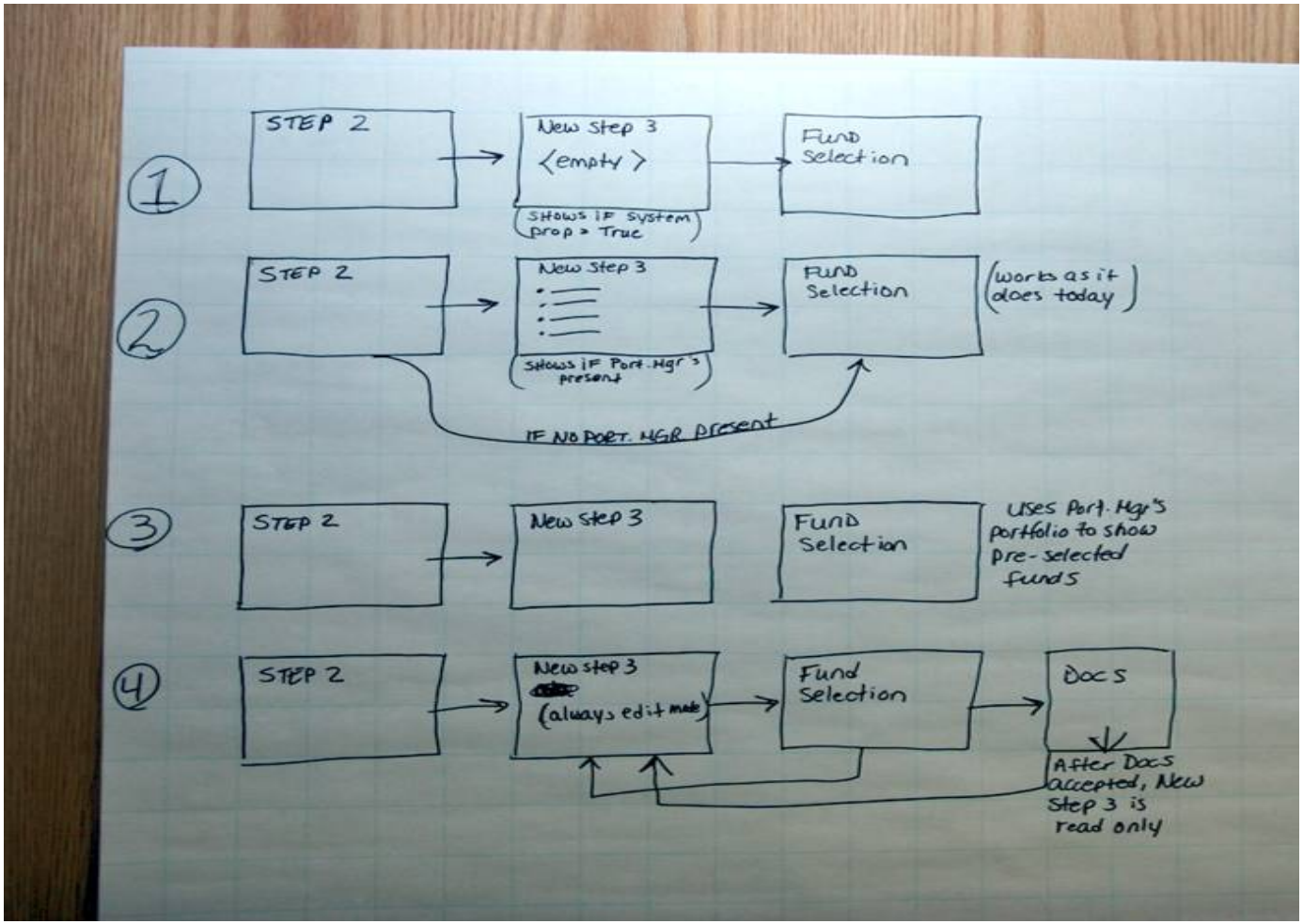
- Address one or two needs at a time
- Understand the problem first
- Try simplest approach first
- Work in small chunks, thin slices
- Incremental & iterative

Automate a Slice at a Time

Example: 4-step UI to validate, upload profit sharing contribution data

- Thread 1: All four pages with navigation
- Thread 2: Select year, enter description on page 1, display on page 2, browse and upload file on page 2
- Thread 3: Validate data in file, display on page 3
- Thread 4: Persist data, display ‘success’ message on page 4

Thin Slice Example



Group Exercise – use flip charts

Given this story:

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 Post, 5-day Ground, 2 day and Overnight. Items > 20 lbs are Ground only. PO Box (Postfach) addresses can only be shipped Post. We have API to cost calculator available, takes postal code and weight

Identify a basic end-to-end slice of functionality that can be coded, tested and automated

(If time, identify additional slices)

Automation Decision Guidelines

- Use risk analysis: probability x impact
- Consider cost of automating, ROI
- High value, high risk – coverage worth it
- Low value, high cost – happy path only, or none

The Importance of Test Design

- Test design skills = production code skills
- Poor design = un-maintainable tests, poor ROI
  - Spend all your time maintaining tests
  - Or abandon them altogether
- Design means the difference between valuable and failed automation

Tests Are Understandable

- Essence of each test is clear
  - Readable by business experts
  - Hide incidental details
  - Tests one thing
- Establish and follow standards
  - eg, FitNesse page template
  - Libraries, building blocks

Tests Are Maintainable

- DRY – don't repeat yourself
  - Extract duplication
  - Reusable components
  - Macros, modules, variables, classes, mixins
- Test design patterns
  - Build/Operate/Check
- Rerunnable
  - Setup/teardown test inputs
  - Randomize values

“But I’m Not a Programmer”

How can we learn correct design?

- Use tools/language that works for whole team
- Pair: tester-programmer
- Peer review
- Use retrospectives, address pain points
- Refactor continually

Enable Good Design

- Frameworks allow non-programmers to specify executable test cases
- Tools allow this: eg., Robot Framework, Cucumber, FitNesse, Twist
- Or team can build custom framework using any driver: Selenium, Watir
- Example: Page Object approach

Let's Work Through Some Examples

- Test design principles illustrated with Robot Framework examples
- Examples from my own team
  - Unit tests
  - FitNesse tests
  - Canoo WebTest scripts
- More examples of DRY from my team
- Some vendor tools promote good design: GUI Dancer Flash demo

Iterative Feedback

- Commit to trying new tool/framework/ technique for N iterations
- Plan automation tasks for each iteration
- Use retrospective to evaluate

Learn by Doing

- Courage – don't be afraid to fail
- Use agile coding practices for automation
  - Simple design
  - Pairing
  - Refactoring
  - Object-oriented, libraries, modules
  - Test-first if scripts have logic
- Small Chunks
- Experiment

Questions About Automation Strategy?

Choosing Tools

- Must be team decision
- Find time for evaluating
  - Story for tool evaluation
  - Iteration for development team
- Determine requirements
- Do some basic research, compile list
- Try each tool

Understand the Purpose

- What's being automated?
  - eg. Ajax, SSL support; load; embedded s/w
  - Speeding up exploratory testing, test data
- Existing tools, environment
  - eg., integration with build process
  - Reporting needs
- Who's writing, maintaining the tests?
- Who's using the tests? What for?

What Fits Your Situation

- Existing skills on team
- Language of application under test
- Collaboration needs
- Utilities (automating tedious tasks) vs. Testing
- Life span, future use of tests

Vendor Tools - Pros

- Existing expertise
- Some built on open-source libraries
- Fast ramp-up for non-programmers
- Perceived as safe choice
- Training, support
- Part of existing tool set
- May have robust features
- New generation of “agile” tools

Vendor Tools - Cons

- Tended to be heavyweight in past
- But this has changed
- Tend to be programmer-unfriendly
  - Also changing
- Scripts may be brittle, high-maintenance
  - Capture-playback problematic
  - Not designed for long-term maintainability
  - Newer tools have overcome this
- Can be pricey

Open-Source Tools - Pros

- Designed by test-infected programmers
- Designed for agile environments
- Designed for maintainability
- Programmer-friendly
- May have excellent support, tutorials, doc
- Easily customized
- Low up-front cost
- Plug in to CI

Open-Source Tools - Cons

- May be difficult for non-programmers
  - Depends on the tool/framework
- Future enhancements may be uncertain
- Training, support can be an issue
- Be sure to look for active development community

Home-Brewed - Pros

- Programmer-friendly
  - Integration with app, IDEs, CI
- Development framework may support
  - Rails, Ruby, Groovy
- Can build on top of existing framework
  - Fit, Slim, Watir, RSpec
- Specifically tailored to needs
- Someone's there to address problems

Home-Brewed - Cons

- Team needs enough bandwidth, expertise
  - Reporting framework
  - Allow test specification by non-programmers
- Could be harder sell to management

Where To Find Tools

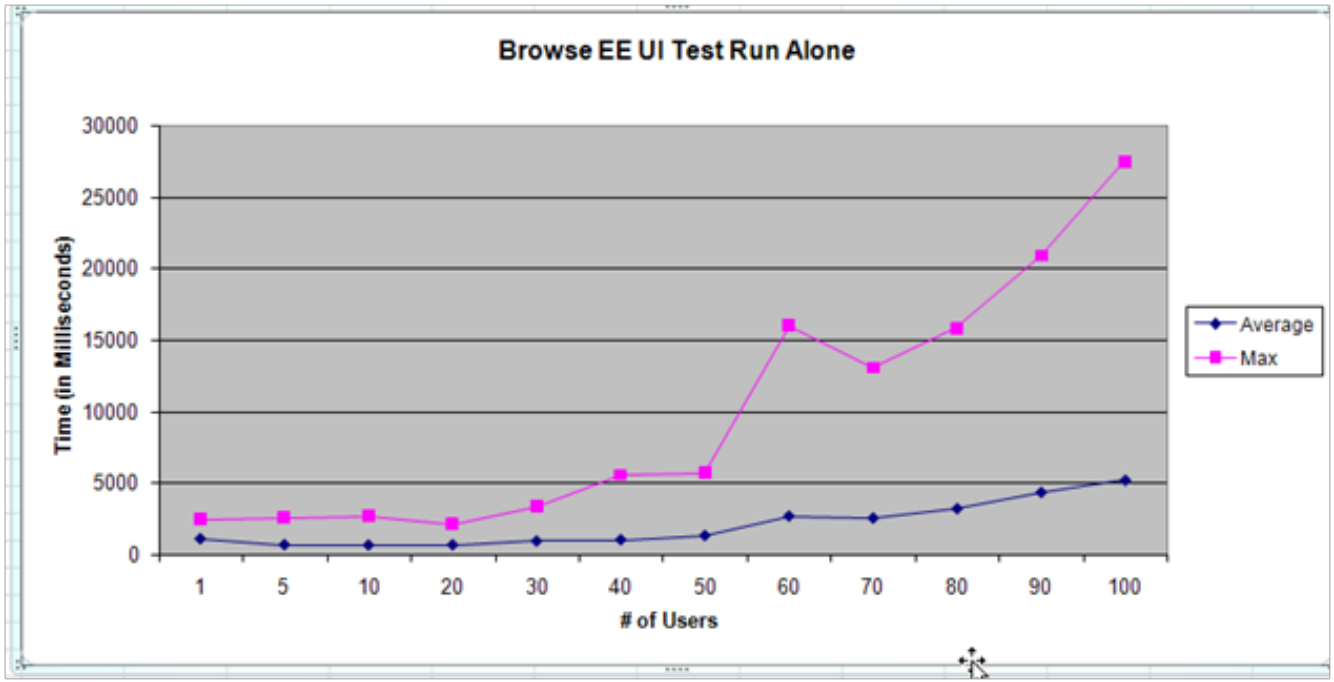
- www.softwareqatest.com/qattls1.html
- www.testingfaqs.org
- www.opensourcetesting.org
- awta.wikispaces.com/2009ToolsList
- groups.yahoo.com/group/agile-testing
- http://bit.ly/AgileTestTools - aa-ftt spreadsheet

Example: My Team's Tool Choices

- IntelliJ Idea
- Hudson for CI
- JUnit for TDD, unit, component
- FitNesse for functional, behind GUI
- Canoo WebTest for GUI regression smoke tests
- Watir to aid exploratory testing
- JMeter for load, performance testing
- Perl scripts for comparing files, making files human-readable
- Java programs for concatenating forms, cover letters

Discussion

Do you have tool success (or failure) stories to share?



Making Test Automation Work

- Time to do it right
- Learning culture
- Testable architecture
- Test data
- Managing tests

Time To Do It Right

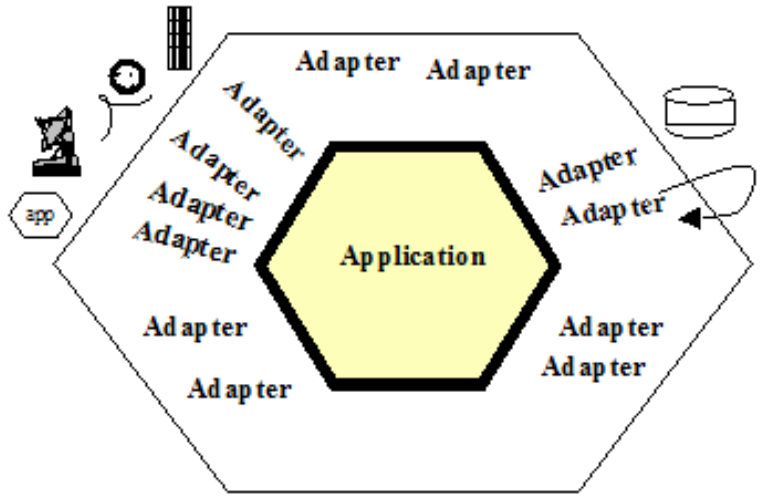
- Limit scope, don't over-commit
- Write automation task cards
- Quality must be goal
- Long-term, will let you go faster

Learning Culture

- OK to make mistakes
- Lots of small experiments
- Slack
- Evolve right design

Testable Architecture

- Layered architecture
  - eg. UI, business logic, data access
- Ports and Adapters pattern
  - App can work without UI or database
  - Ports accept outside events
  - Adapters convert for human or automated users



Test Data

- Avoid database access when possible
- Setup/Teardown
  - Independent, rerunnable tests
- Canonical data
  - Refresh before each test run
- Customizable data for ET
- Production-like data
  - Get customers to provide example data

Managing Automated Tests

- Documenting tests
  - Tests as documentation
  - Finding what you need
- Running tests – Continuous Integration
- Reporting results
  - Analyze failures
- Test coverage

Tests as Documentation

- Automated tests can be readable by everyone
  - Given/When/Then BDD style is one way
  - Automation can be hidden or built in
  - "Do" or "Scenario" fixture is another way
- Tests automated in CI must pass = documentation is up to date!

Given/Then/When Example

Scenario: Valid name search returns results

GIVEN that Kant is a supervisor with employees
AND Kant has an employee named Smith
WHEN Kant navigates to the employee name search page
AND enters the value “S”
THEN Kant will see a search result that includes Smith

FitNesse “Do” Fixture

1. Take out a loan
2. Check the calculated loan payment
3. Post the payment, then receive it
4. Settle and confirm the payment
5. 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							

Test Management Tools

- Tool may include management framework
  - FitNesse
    - Wiki – documentation + executable tests
    - Hierarchy
  - Rasta – spreadsheets to organize tests
  - Twist – uses Eclipse IDE
- What problem are you trying to solve?
- Simplest approach
- Check tests into same sccs as production code

A Look At One Team's Approach

Demo

- Wiki
- FitNesse
- CI

Questions on test management?

Key Success Factors

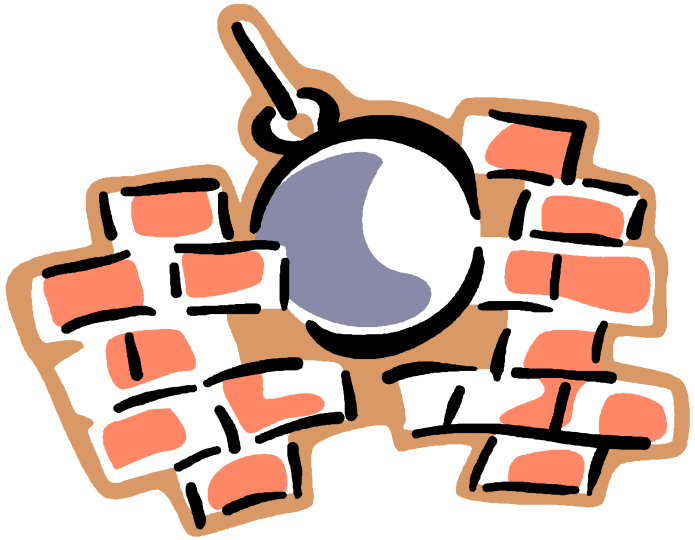
- Whole team approach
- Simple approach
- Iterative feedback
- Good test design
- Agile principles, practices
- Learning culture

Succeeding with Test Automation

- Don't overcommit – budget time to automate tests
  - Well-designed tests mean speed later
- Need testable architecture, design
  - But don't get stuck, find a way
- Keep feedback loop short – use CI
- Team designs how much coverage is enough
  - Focus on feature coverage, not % of code

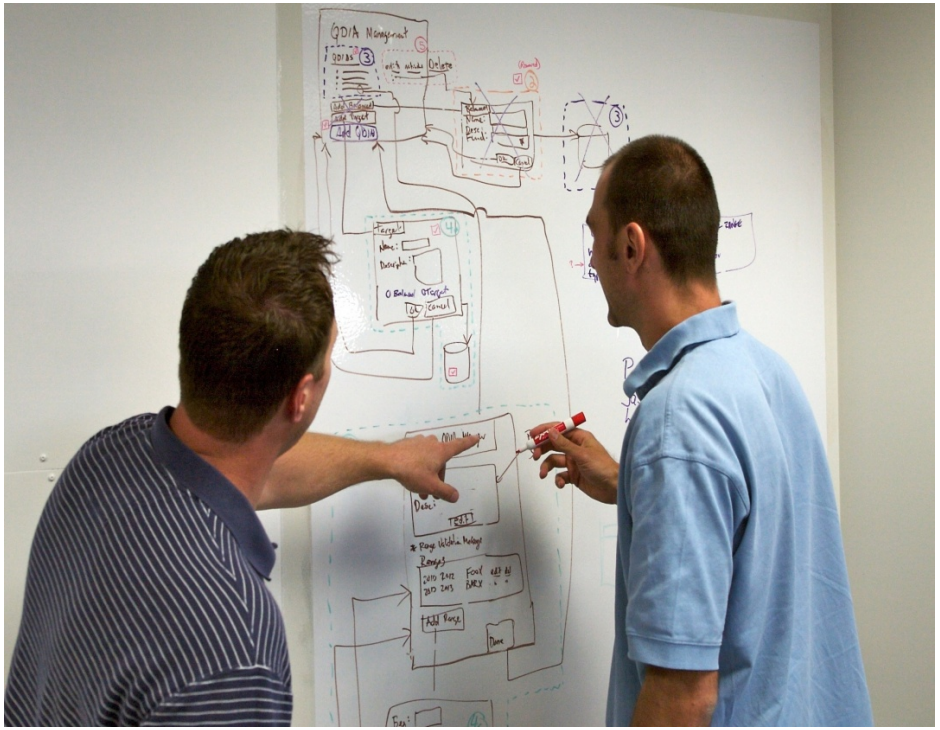
Exercise

- Look at the barriers from the first exercise.
- What's the first thing you will do when you get back to the office to overcome a barrier to automation? Write it on a post-it and put it next to the barrier on the flip chart paper.



Remember

- It's a team problem!
- Build foundation of core agile practices
- Design investment, refactoring pay off
- Experiment
- Baby steps



Questions? “Aha” Moments?

Some Agile Testing Tool Resources

- http://bit.ly/AgileTestTools tool spreadsheet aa-ftt
- awta.wikispaces.com/2009ToolsList
- softwareqatest.com/qattls1.html
- opensourcetesting.org
- nunit.org/index.php
- webtest.canoo.com
- testingfaqs.org
- junit.org
- watir.com
- gojko.net/fitnesse/dbfit
- fitnesse.org
- seleniumhq.org
- code.google.com/p/robotframework

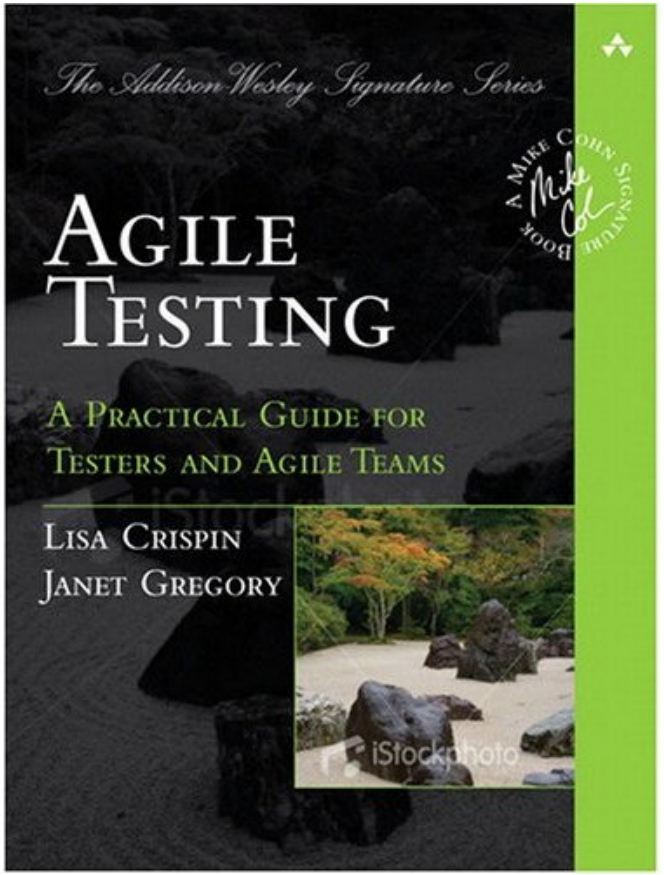
Honing Your Craft

- www.weekendtesters.com
- www.softwaretestingclub.com

Agile Testing: A Practical Guide for Testers and Agile Teams

By Lisa Crispin and Janet Gregory

www.agiletester.ca

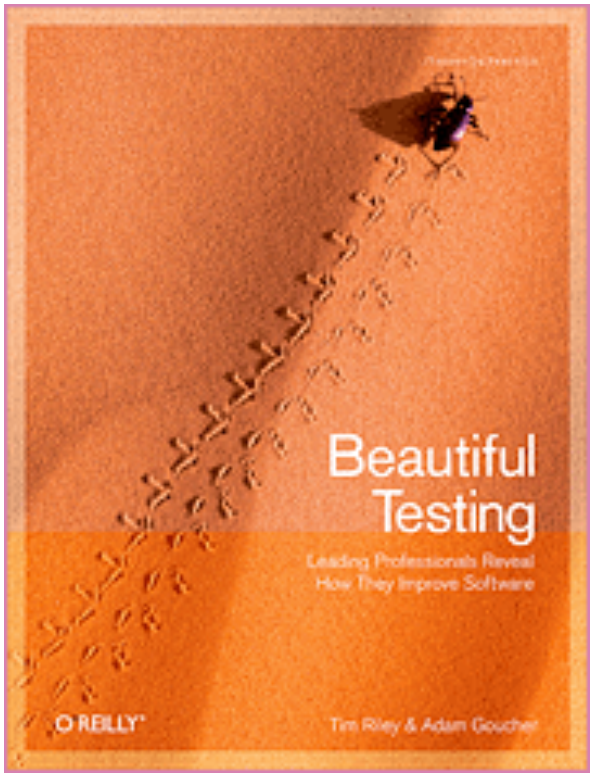


Now Available

Beautiful Testing: Leading Professionals Reveal How They Improve Software

Edited by Tim Riley, Adam Goucher

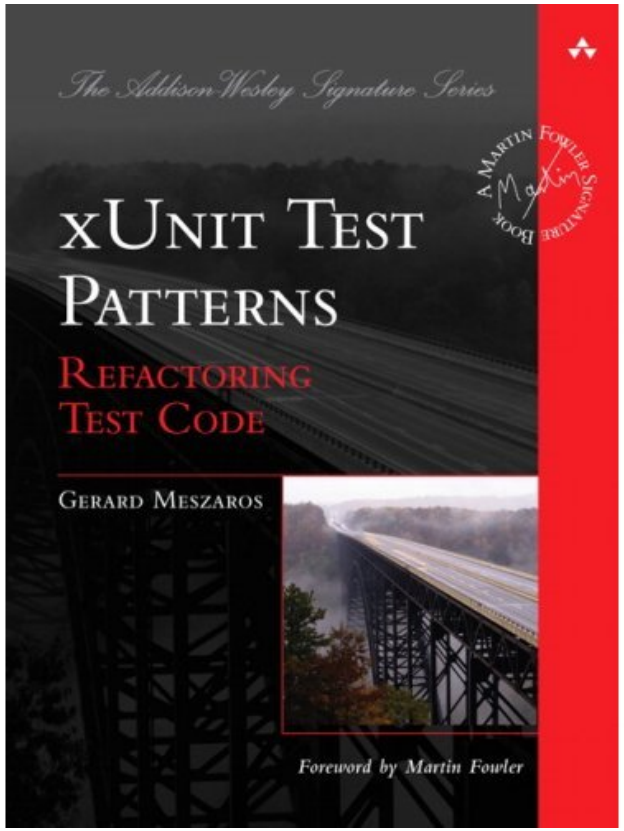
Includes chapter by yours truly



Test Patterns

Xunit Test Patterns: Refactoring Test Code

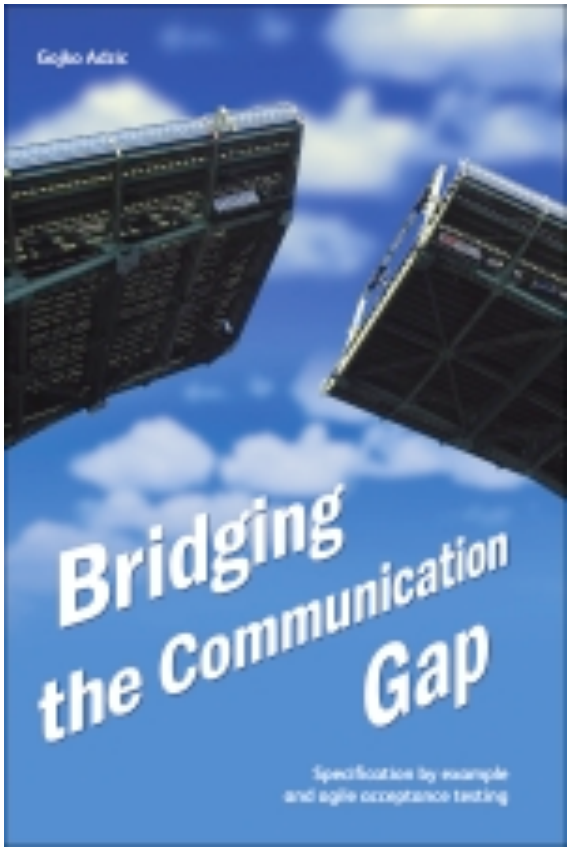
By Gerard Meszaros



Bridging the Communication Gap

Specification By Example and Acceptance Testing

Gojko Adzic



Agile Test Automation Resources

- dhemery.com/pdf/writing_maintainable_automated_acceptance_tests.pdf
- agile-testing@yahoogroups.com
- lisacrispin.com
- janetgregory.ca
- agilealliance.org
- exampler.com
- testobsessed.com
- testingreflections.com
- pairwith.us