Successful Automation in an Agile Project

SQDG May 2011

Janet Gregory, DragonFire Inc.

With material from Lisa Crispin

A Little About Me

My experience comes …
As a tester, working on agile teams
Coaching and training, learning

Programming background
Test automation since 2000

Agile Testing: A Practical Guide for Testers and Agile Teams; Addison Wesley 2009

Takeaways

- Foundation for successful test automation
- "Whole Team" approach
- Where to start
  - Using the automation test pyramid
  - ATDD (Acceptance Test Driven Development)
- Choosing tools

Why Automate

- Manual testing takes too long
- Manual tests are error prone
- Frees people to do their best work
- Provides ‘living documentation’
- Repeatable
- Saves time

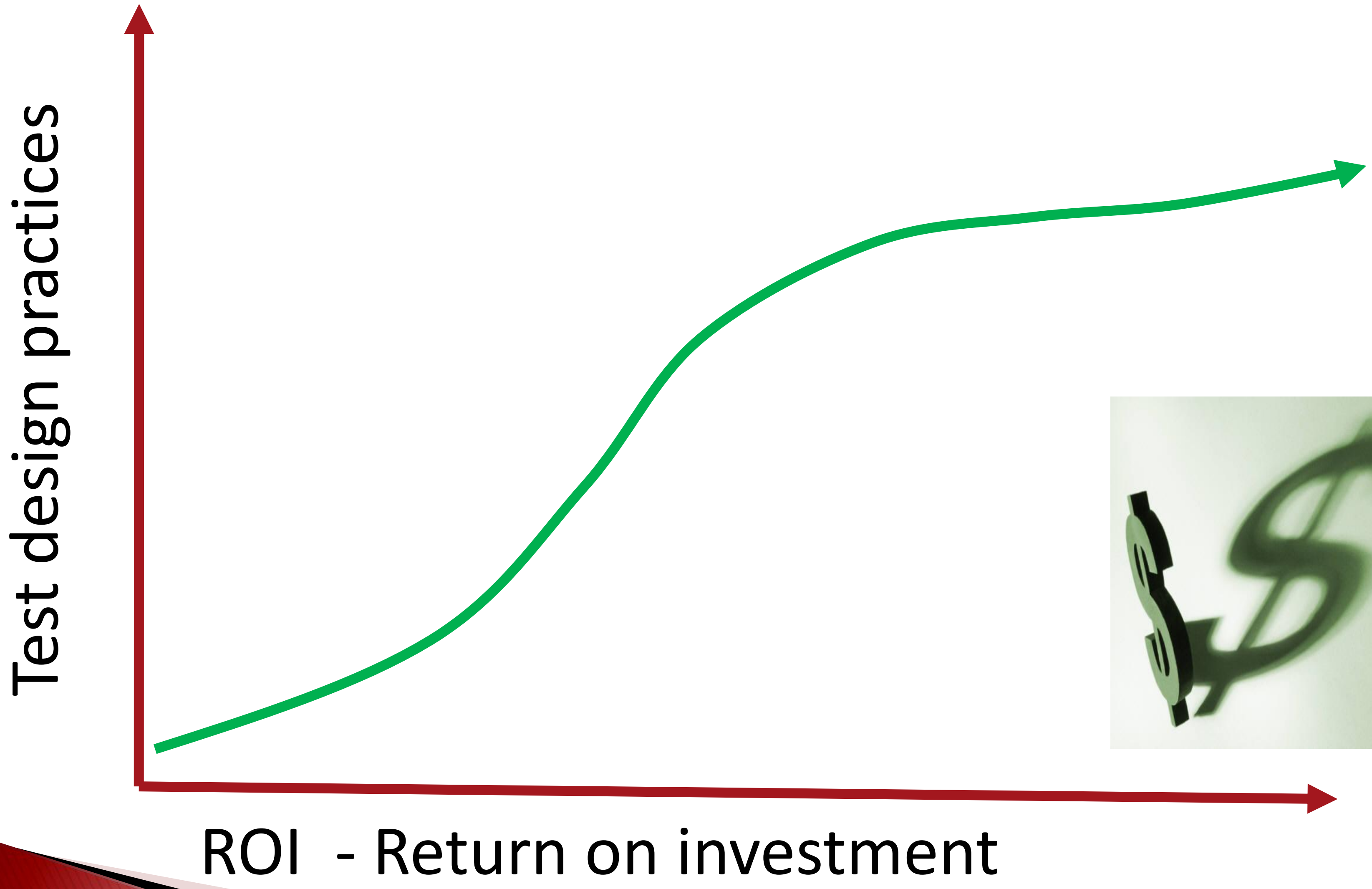
Lets' Start With a Discussion

What concerns you about test automation?

Why aren't you doing it?

The Foundation

Economics of Test Design



Functional Test Design

- Simple – no extra details
- One purpose
- Rerunnable
- Readable by the business
- DRY – Don’t repeat yourself (duplication)
- Follows code design principles
- Easy to refactor
- Easy to maintain when code changes

Example Test – Ruby / Watir

```
def test_validate_password_minimum_length
    # test that password needs at least 7 chars
    # takes 2 parameters (user name, new password)
   puts "... check password minimum length“ [Comment in log]
   set_password "pass$Role1","adcD!1"
   verify_text  "The password must contain at least 7 characters."
end
```

Test Data

- Avoid database access when possible
  - In memory database for unit or API tests
- Setup / tear down test data
- Use canonical data
- Use ‘production-like’ data
  - Get customers to provide example data

Whole Team Approach

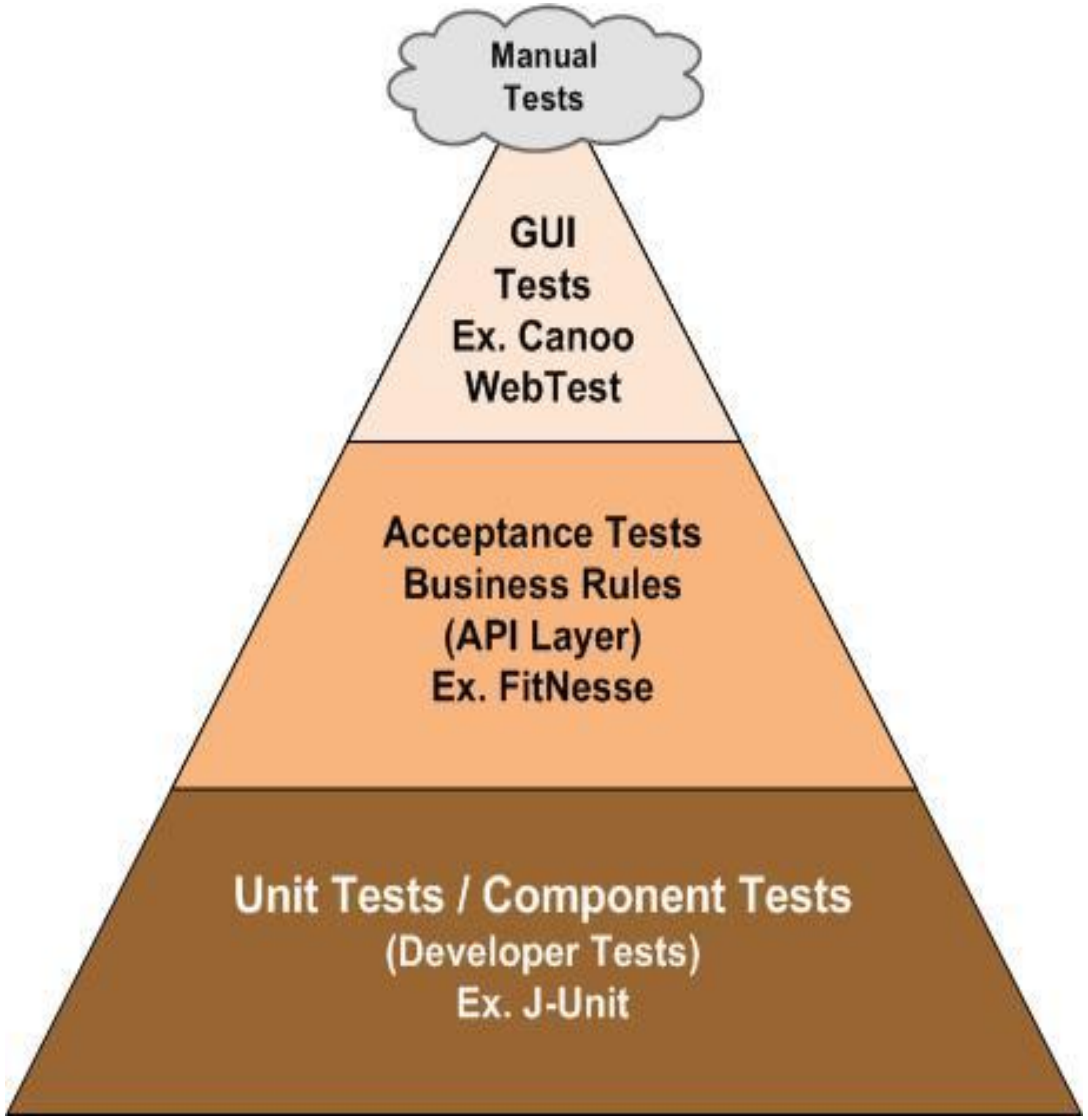
- Whole team = project team
- Team is responsible for testing activities
- Includes automation
- Whole team has all the skills needed
- Team designs for ease of test automation

Where to Start

Simplicity

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

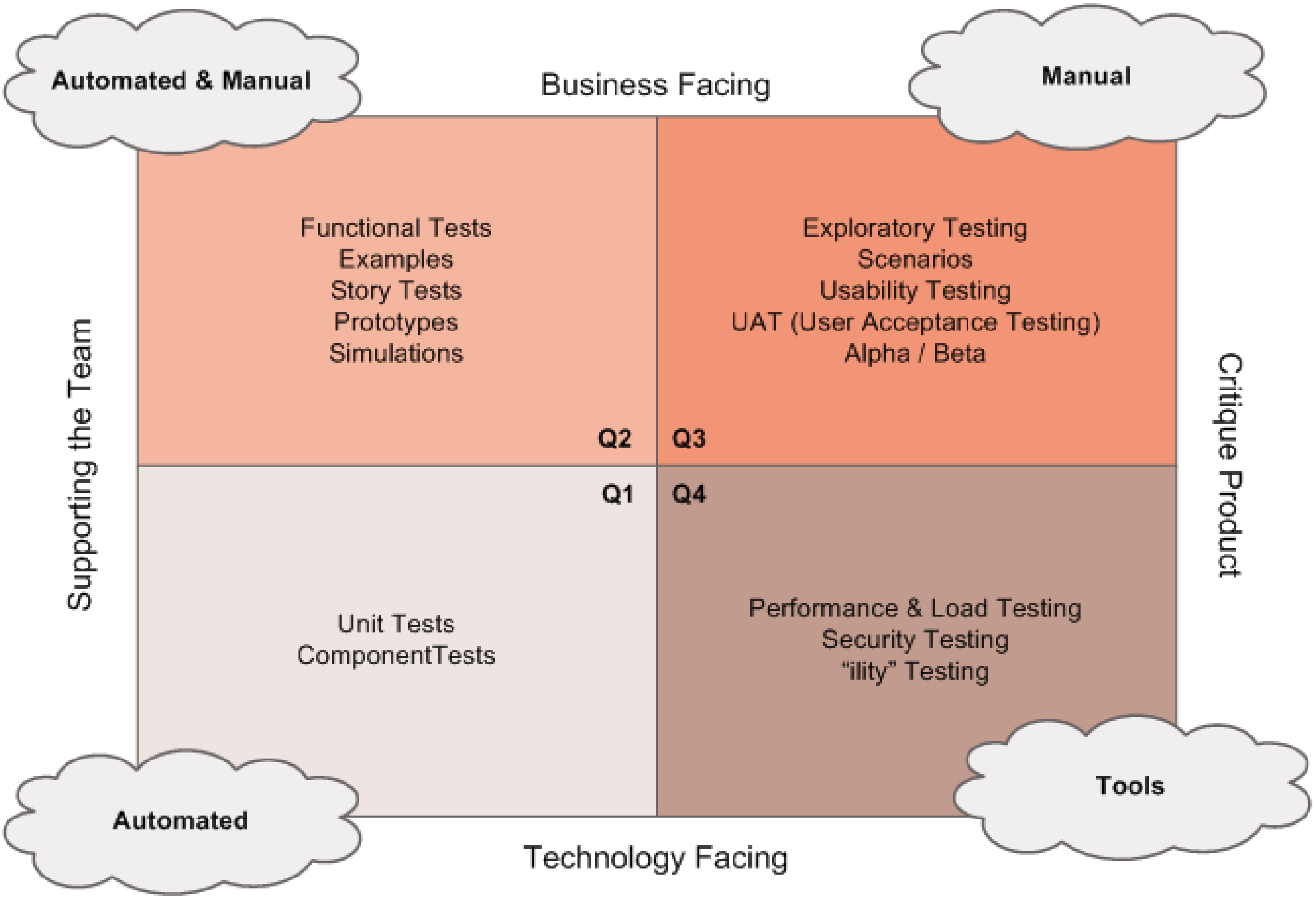
Agile Testing Pyramid (Mike Cohn)


Manual Tests
GUI Tests Ex. Canoo WebTest
Acceptance Tests Business Rules (API Layer) Ex. FitNesse
Unit Tests / Component Tests (Developer Tests) Ex. J-Unit

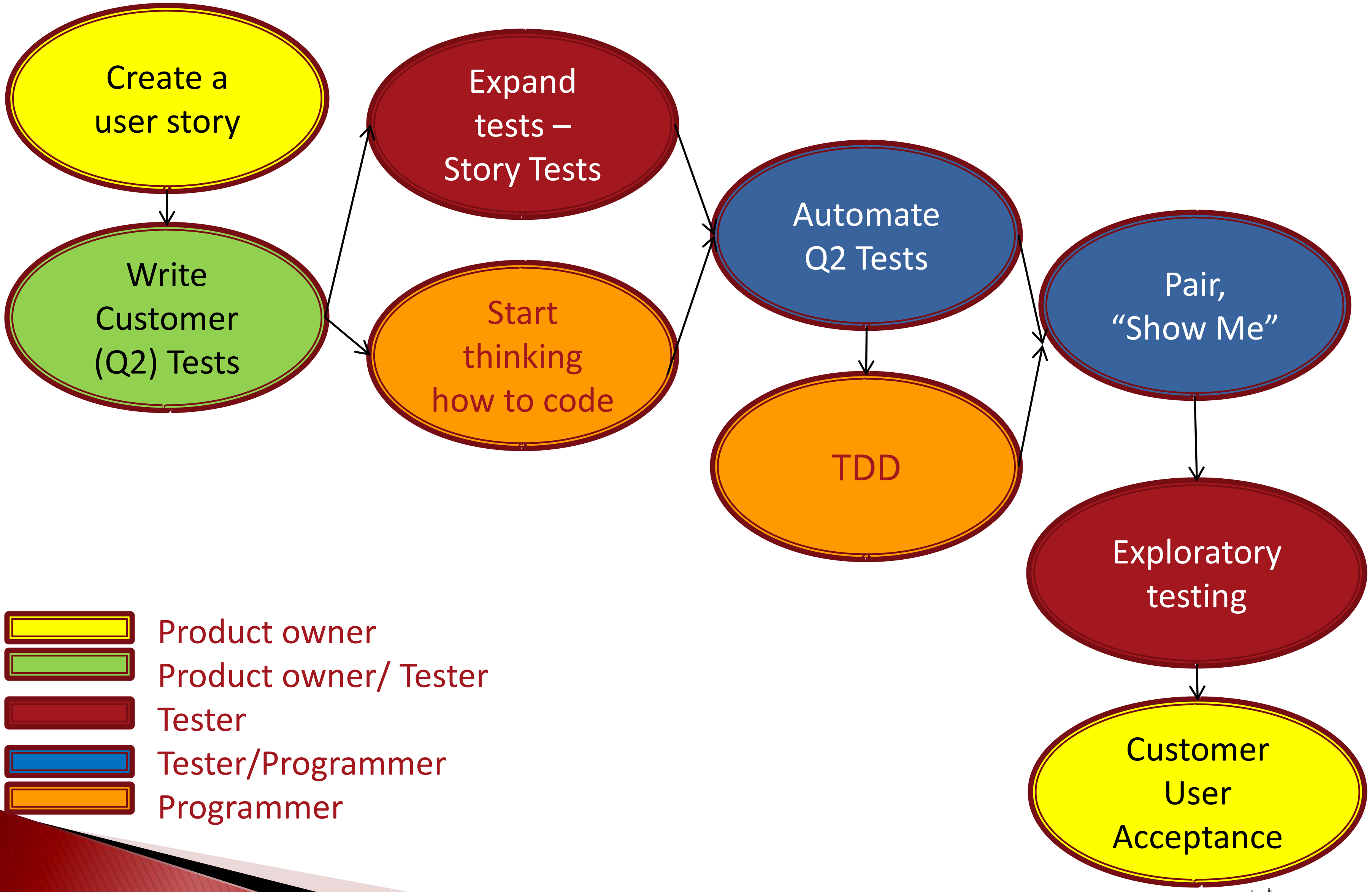
What Else Can We Automate

- Any tedious or repetitive task
- Testing related or otherwise
- Builds – continuous integration
- Parsing
- Comparing output
- Set up for exploratory testing

Agile Testing Quadrants (Brian Marick)



ATDD (Acceptance Test Driven Dev)



🟨 Product owner
🟩 Product owner/ Tester
🟥 Tester
🟦 Tester/Programmer
🟧 Programmer

Example Story

As a new user, I want to create an account with a user name and password so that only I can access my information.

==========================

Acceptance Test – BDD Style

BDD – Behavioural Driven Development

Given the user has no existing account

When she requests to create a new account,

Then she is presented with a screen to enter a valid user name and valid password (rules defined)

And the information is saved upon submitting.

Acceptance Tests – Fit Automation Style

User Name	Password	Expected result	comments
JanetGregory	Password	Login	Valid combo saved
Janet Gregory	Password	Error	Space in user name
JanetGregory	Abc	Error	Invalid password

Expanded Tests Examples – To Automate

User Name	Password	Expected result	comments
JanetGregory	Password	Login	Valid combo saved
Janet Gregory	Password	Error	Space in user name
JanetGregory	Abc	Error	Invalid password
Janet#Gregory	Password	Error	Special char no allowed
	Password	Error	Blank user name
JanetGregory		Error	Blank password
JanetGregory	Password	Error	User already exists

Example API test – Pizza application

# **CreateTopping** method takes 2 parameters: name, price

Add a new topping,,**CreateTopping**,PineApple,22

Verify it was saved,22,**GetTopping**,PineApple,price

==

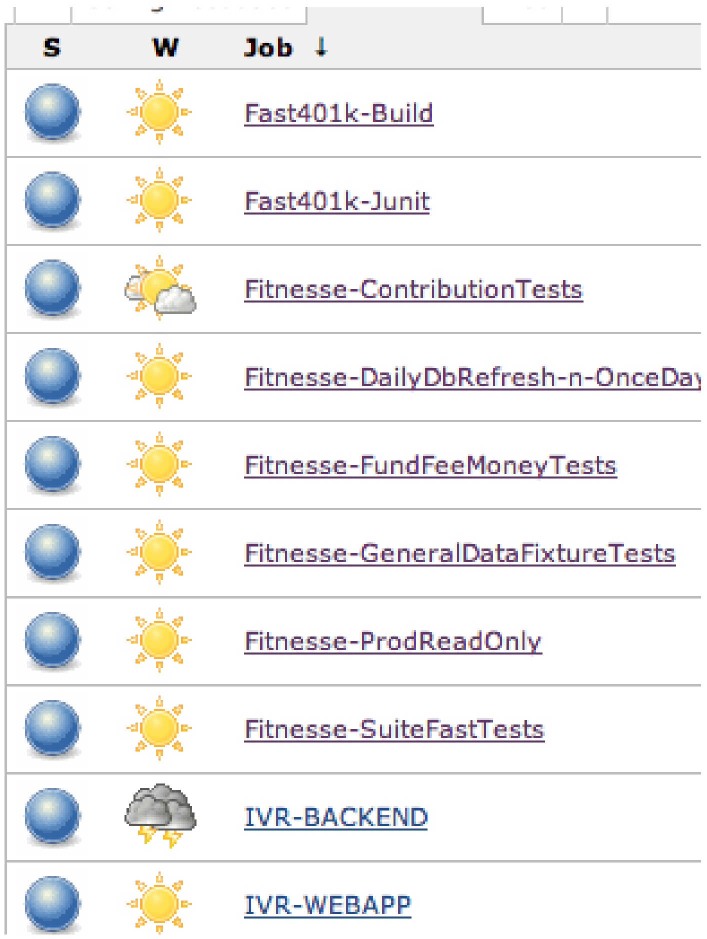
# **EditTopping** method takes 3 parameters: the topping to edit, the attribute to edit,the new value

Edit a Topping,,**EditTopping**,PineApple,price,23

Verify it was saved,23,**GetTopping**,PineApple,price

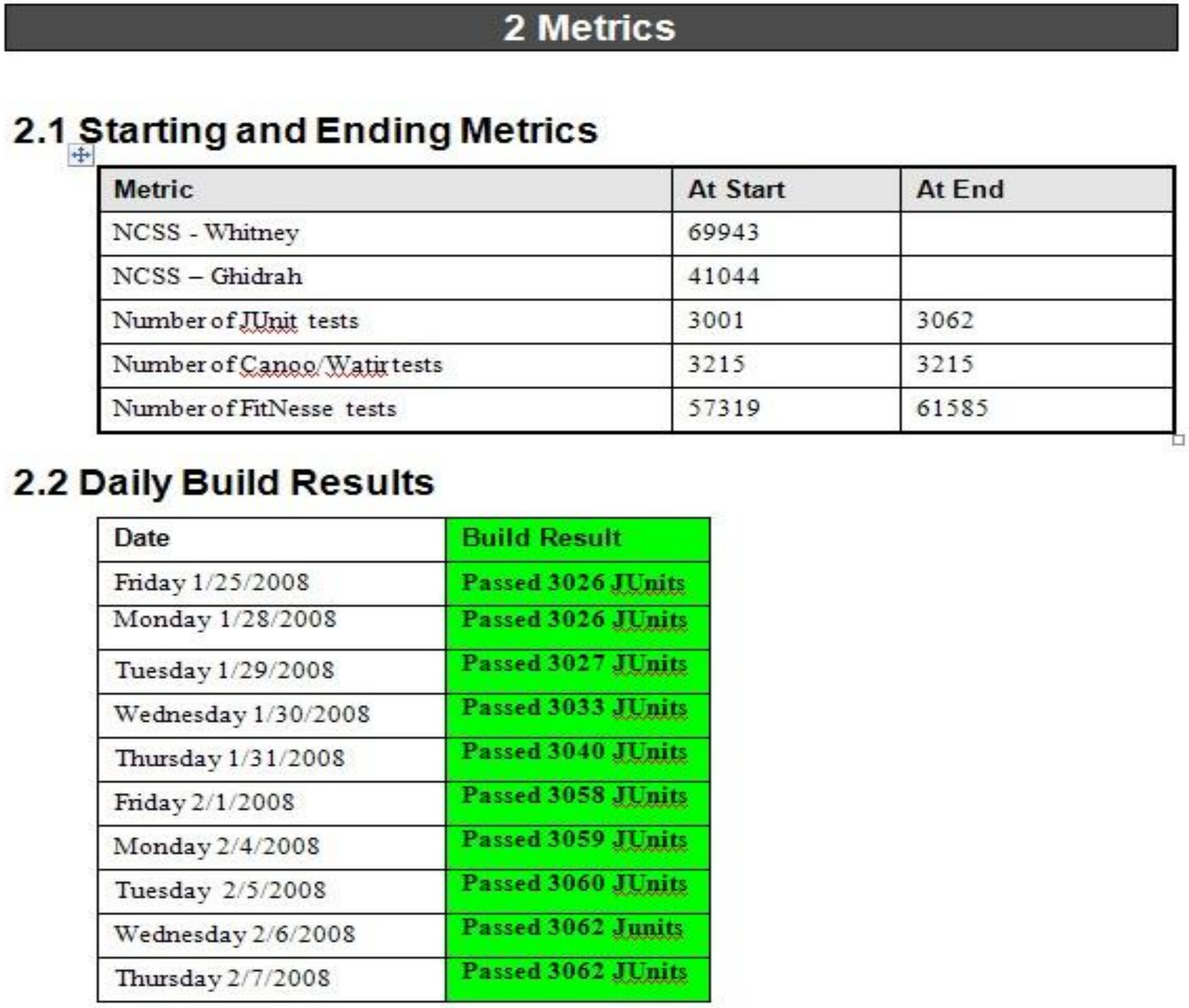
Managing Automated Tests

- Use source control
- Continuous integration
  - To run tests
  - To report results
  - Stable builds for testing
- Keep them passing
  - Analyze failures



S	W	Job ↓
●	☀	Fast401k-Build
●	☀	Fast401k-Junit
●	⛅	Fitnesse-ContributionTests
●	☀	Fitnesse-DailyDbRefresh-n-OnceDay
●	☀	Fitnesse-FundFeeMoneyTests
●	☀	Fitnesse-GeneralDataFixtureTests
●	☀	Fitnesse-ProdReadOnly
●	☀	Fitnesse-SuiteFastTests
●	⛈	IVR-BACKEND
●	☀	IVR-WEBAPP

Build Results



2 Metrics

2.1 Starting and Ending Metrics

Metric	At Start	At End
NCSS - Whitney	69943	
NCSS – Ghidrah	41044	
Number of JUnit tests	3001	3062
Number of Canoo/Watir tests	3215	3215
Number of FitNesse tests	57319	61585

2.2 Daily Build Results

Date	Build Result
Friday 1/25/2008	**Passed 3026 JUnits**
Monday 1/28/2008	**Passed 3026 JUnits**
Tuesday 1/29/2008	**Passed 3027 JUnits**
Wednesday 1/30/2008	**Passed 3033 JUnits**
Thursday 1/31/2008	**Passed 3040 JUnits**
Friday 2/1/2008	**Passed 3058 JUnits**
Monday 2/4/2008	**Passed 3059 JUnits**
Tuesday 2/5/2008	**Passed 3060 JUnits**
Wednesday 2/6/2008	**Passed 3062 Junits**
Thursday 2/7/2008	**Passed 3062 JUnits**

Choosing Tools

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

Vendor Tools - Cons

- Tend to be heavyweight
- Updates tend to be slower
- Tend to be programmer-unfriendly
- Scripts may be brittle, high-maintenance
  - Capture-playback problematic
  - Not always designed for long-term maintainability
- 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

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
- 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

Putting it all Together

- Understand your context
- Understand the purpose
- Consider ROI (return on investment)
- Push the tests lower
- Automate the repetitive and boring tests
- Plan, but document simply
- Plan, but plan for the appropriate level

Some Agile Testing Tool Resources

- http://bit.ly/AgileTestTools   aa-ftt tool spreadsheet
- awta.wikispaces.com/2009ToolsList
- softwareqatest.com/qattls1.html
- opensourcetesting.org
- testingfaqs.org
- junit.org
- nunit.org/index.php
- watir.com
- fit.c2.com
- gojko.net/fitnesse/dbfit
- fitnesse.org
- selenium.openqa.org
- code.google.com/p/robotframework

Now Available

Agile Testing: A Practical Guide for Testers and Agile Teams

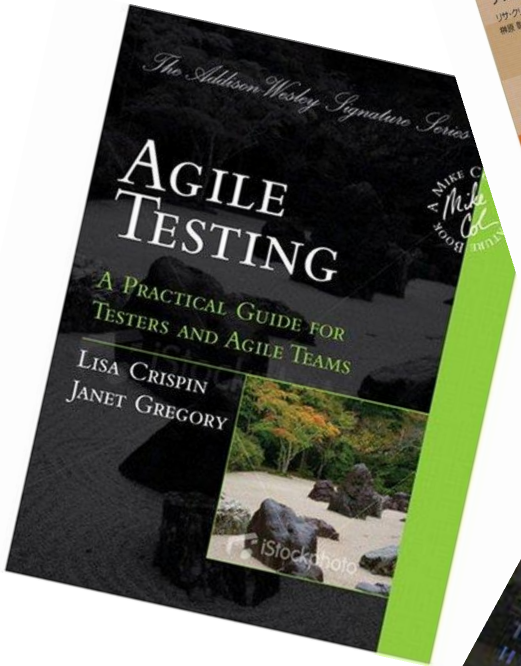
By Lisa Crispin and Janet Gregory

www.agiletester.ca

My contact info

www.janetgregory.ca
http://janetgregory.blogspot.com/
Email: janet@agiletester.ca

www.lisacrispin.com
http://lisacrispin.com
Email: lisa@agiletester.ca



Let’s talk about your concerns?

Are there still unanswered questions?