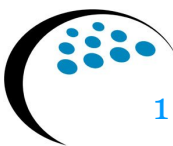


# One Week Iterations! You are joking right?

---

Martin Blower  
Black Pepper Software Ltd

[martin.blower@blackpepper.co.uk](mailto:martin.blower@blackpepper.co.uk)



# Agenda

---

Practical tools and techniques that help us to make our iterations shorter...

- ❖ Why shorter iterations?
- ❖ Improving inefficient practices in typical projects :
  - code review vs automated static analysis
  - manual vs automated unit testing
  - by product - test coverage
  - manual vs automated acceptance testing
  - manual vs automated deployment
  - by product - the value of metrics

# Demands on Development

---

More:

- ❖ Complexity
- ❖ Innovation
- ❖ Flexibility
- ❖ Scope
- ❖ Integration
- ❖ Change
- ❖ Predictability

# Demands on Development

---

Less:

- ❖ Time to delivery
- ❖ Development Costs
- ❖ Ownership Costs

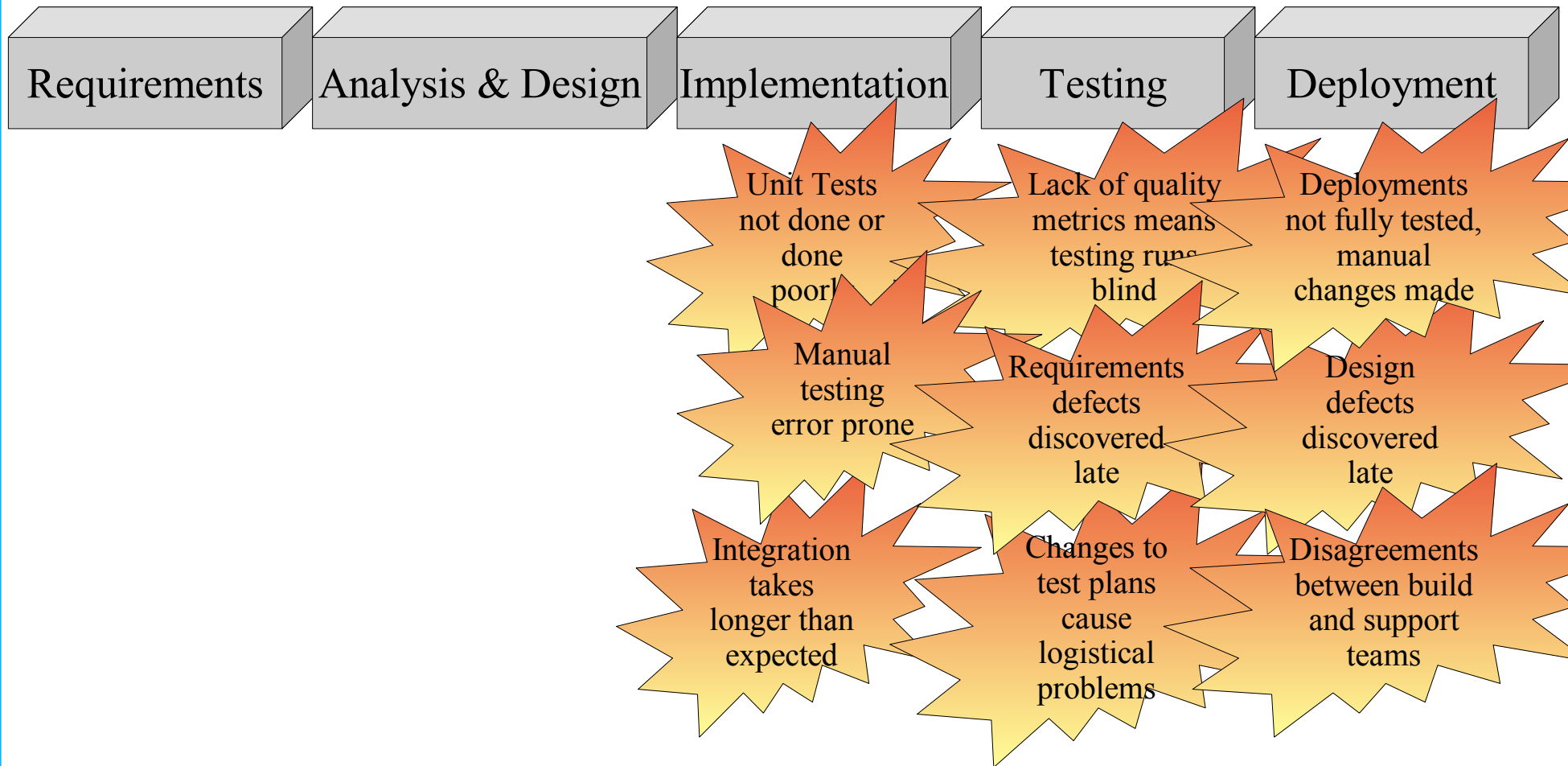
Our Challenge is to find a way to meet these all of these demands

# Waterfall Lifecycles

---

- ❖ Appropriate in some cases
- ❖ But generally has some major problems
  - Not particularly change friendly
  - Early focus is on documentation
  - Risks are back loaded
    - Don't discover problems until late in lifecycle

# Engineering Risks



# Iterative Lifecycles

---

- ❖ Provide improvement using smaller 'chunks'
  - Reduces test, fix re-test cycle time
  - Reduces magnitude of integration problems
- ❖ But still don't solve all the problems
  - When done badly can be worse!
    - Whole use-case
    - Massive iteration plans
    - Long iterations
  - Can be improved by reducing iteration length



# Reducing Iteration Length

---

- ❖ Benefits can be significant
  - Reduction in overheads
  - Faster feedback
  - Identify design issues quicker
  - Adapt to changing business quicker
  - Faster customer buy-in

# Reducing Iteration Length

---

- ❖ Can be achieved by:
  - Increasing team size
  - Doing less in the iteration
  - Automating manual tasks

# Agenda

---

Practical tools and techniques that help us to make our iterations shorter...

- ❖ Why shorter iterations?
- ❖ Improving inefficient practices in typical projects make them better :
  - code review vs automated static analysis
  - manual vs automated unit testing
  - by product - test coverage
  - manual vs automated acceptance testing
  - manual vs automated deployment
  - by product - the value of metrics

# Code Review

---

- ❖ Traditionally slow, expensive and unreliable
  - Time consuming
    - Examining code, review meeting, applying outcome
  - Ties up senior staff
  - Rarely finds anything significant
    - Real issues get hidden by all the noise
    - Reviews get hijacked by other agendas

# Code Review

---

- ❖ Most projects insist on them! Why?
  - Standards Governance
    - Coding Standards
    - Identify bad practice
  - Architecture and design compliance
    - Design Issues

# Improving Code Reviews

---

## ❖ Static Analysis Tools

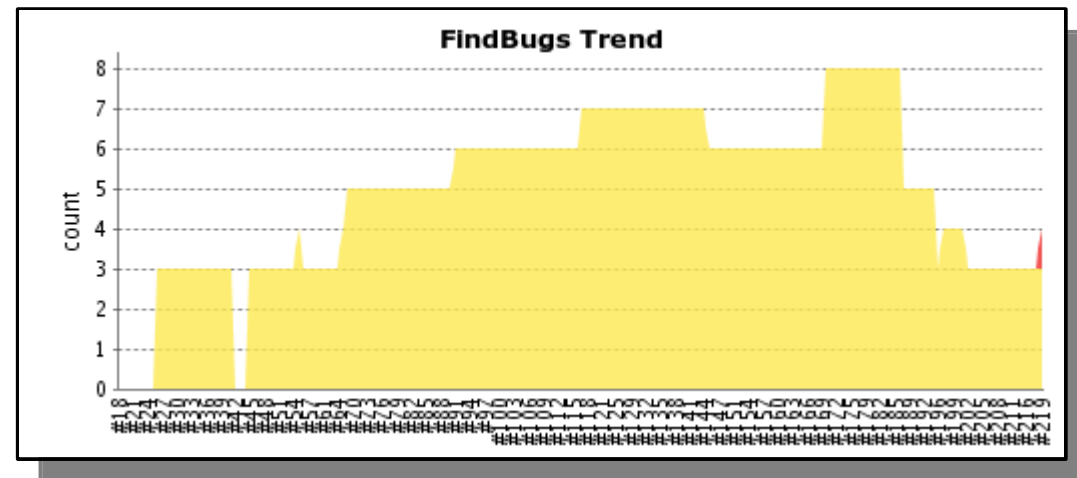
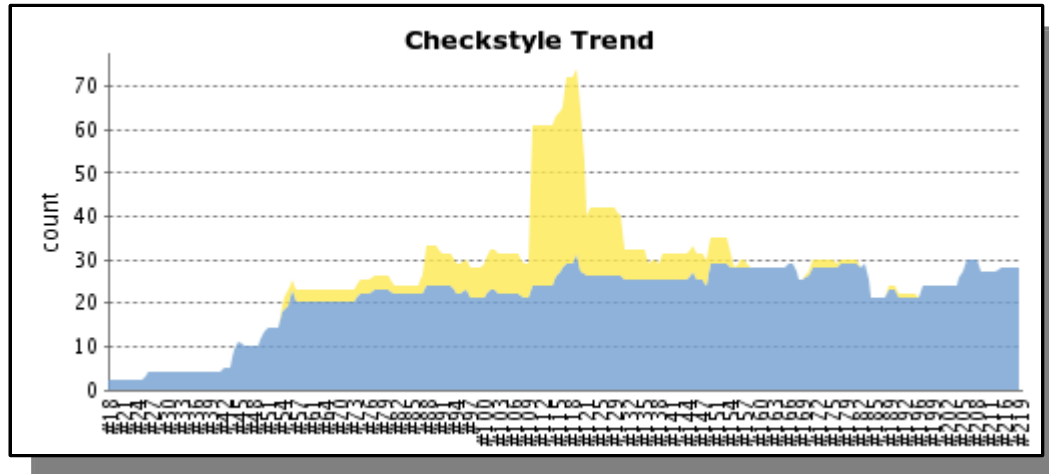
- Some have been around for some time:
  - Lint, McCabe, LDRA
- Others are more recent and open-sourced:
  - Checkstyle, PMD, Findbugs, Crap4J
  - many cloned for other environments such as .net
- Available as plugins for most IDEs
- Can be added to the build pipeline

# Benefits of static analysis

---

- ❖ Can check for standards compliance
  - Tailored to use your own rules
- ❖ Can warn of potential coding problems
- ❖ Can calculate complexity metrics
  - Highlighting areas that might need attention
- ❖ Generate detailed reports
- ❖ Trend analysis using build automation tools

# Example Analysis



# Find the Bug!

## FindBugs Result

### Warnings Trend

All Warnings	New Warnings	Fixed Warnings
4	<a href="#">1</a>	0

### Summary

Total	High Priority	Normal Priority	Low Priority
4	<a href="#">1</a>	<a href="#">3</a>	0

### Details

Packages Files Types Warnings Details New High Normal

Package	Total	Distribution
<a href="#">uk.co.blackpepper.masterdatamanager.gwt.service</a>	3	
<a href="#">uk.co.blackpepper.masterdatamanager.shared.model</a>	1	

## FindBugs Warnings - Package uk.co.blackpepper.masterdatamanager.shared.model

### Summary

Total	High Priority	Normal Priority	Low Priority
1	<a href="#">1</a>	0	0

### Details

#### Details

File: [GwtProperty.java](#), Line: 85, Type: SE\_BAD\_FIELD\_STORE, Priority: High, Category: CORRECTNESS

TEST: Unknown warning SE\_BAD\_FIELD\_STORE in uk.co.blackpepper.masterdatamanager.shared.model.GwtProperty.type; core bug patterns not found

A non-serializable value is stored into a non-transient field of a serializable class.

# Code Reviews Revisited

---

- ❖ Does not replace a good code review
  - Essentially human activity
  - Relies on judgement and experience
- ❖ Does make code reviews easier and faster
  - Removes all coding standard 'noise'
  - Removes common coding issues
  - Highlights areas for examination
  - Lets reviewers concentrate on higher level issues

# Agenda

---

Practical tools and techniques that help us to make our iterations shorter...

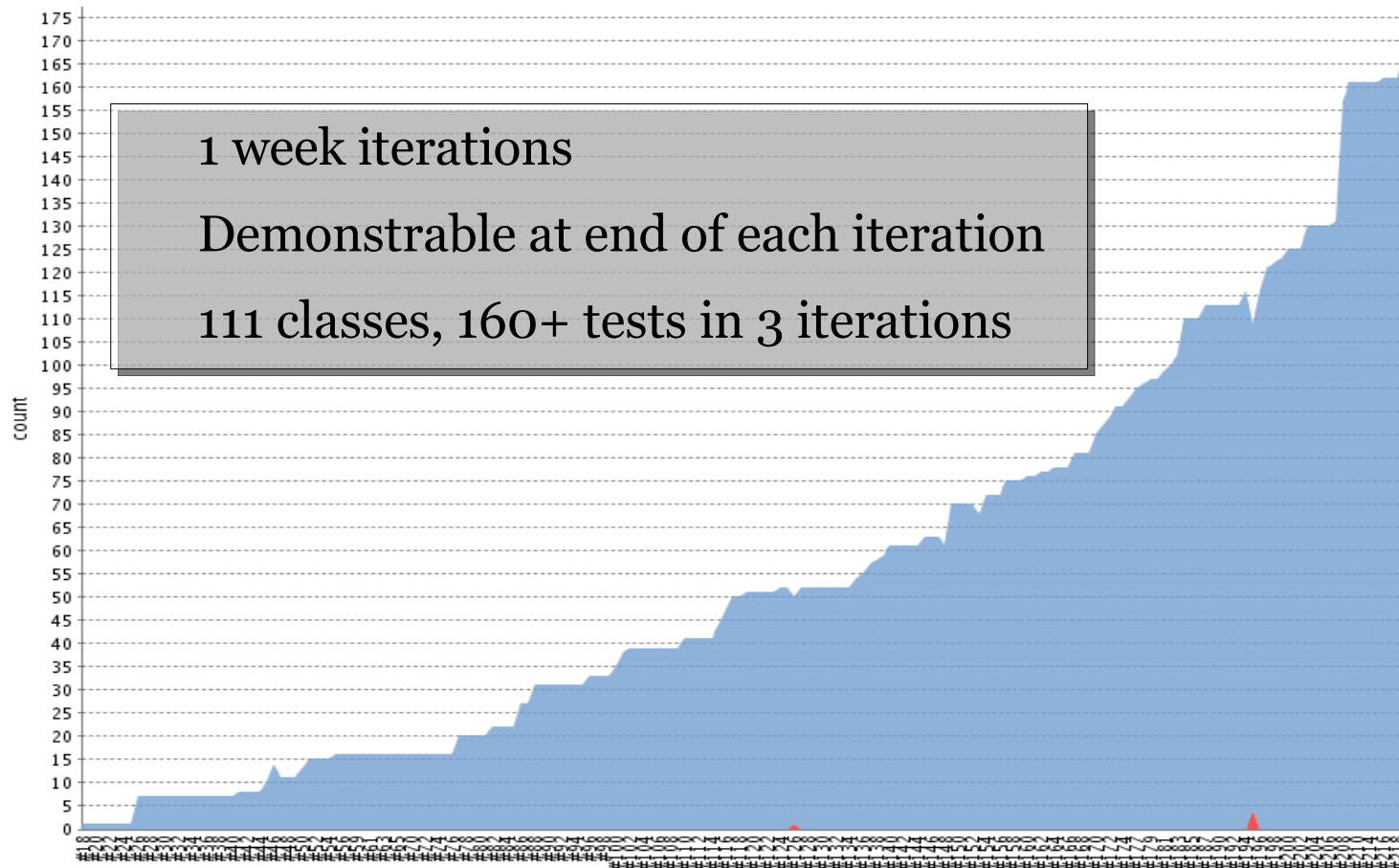
- ❖ Why shorter iterations?
- ❖ Improving inefficient practices in typical projects make them better :
  - code review vs automated static analysis
  - manual vs automated unit testing
  - by product - test coverage
  - manual vs automated acceptance testing
  - manual vs automated deployment
  - by product - the value of metrics

# Unit Testing

---

- ❖ Developers tend to assume their code works
- ❖ Unit Testing tends to be sporadic
  - Time spent building functionality
  - Testing isn't cool
  - Can't unit test this code
  - No metrics produced
- ❖ Leads to late discovery of problems

# Testing Times



# Testing can be cool

---

- ❖ Understand your approach to unit testing
  - Test Driven Development
    - Little or no prior design
    - Tests written before code
    - Use the test development to drive the design
  - Test First Development
    - Design Model available
    - Develop tests first using boundary value analysis

# Design for Testability

---

- ❖ Nothing special – just good OOD
  - Clear responsibilities
  - Loose coupling
  - Design by contract
  - Dependency Injection
- ❖ Use mock objects to create fixtures

# Testing Metrics

Build History		(trend)
#219	<a href="#">17-Nov-2008 17:21:10</a>	
#218	<a href="#">17-Nov-2008 16:25:51</a>	
#217	<a href="#">17-Nov-2008 16:24:11</a>	
#216	<a href="#">17-Nov-2008 15:44:11</a>	
#215	<a href="#">17-Nov-2008 14:36:11</a>	
#214	<a href="#">17-Nov-2008 11:26:11</a>	
#213	<a href="#">17-Nov-2008 11:02:04</a>	
#211	<a href="#">17-Nov-2008 10:36:11</a>	
#210	<a href="#">14-Nov-2008 16:25:51</a>	
#209	<a href="#">14-Nov-2008 16:24:11</a>	
#208	<a href="#">14-Nov-2008 15:44:11</a>	
#207	<a href="#">14-Nov-2008 14:36:11</a>	
#206	<a href="#">13-Nov-2008 17:21:10</a>	

## Test Result

0 failures (±0)

169 tests (+7)

## All Tests

Package	Duration	Fail	(diff)	Total	(diff)
<a href="#">uk.co.blackpepper.masterdatamanager</a>	0 sec	0		1	
<a href="#">uk.co.blackpepper.masterdatamanager.client.framework</a>	10 sec	0		3	
<a href="#">uk.co.blackpepper.masterdatamanager.client.util</a>	3 sec	0		5	
<a href="#">uk.co.blackpepper.masterdatamanager.gwt.service</a>	0 sec	0		21	
<a href="#">uk.co.blackpepper.masterdatamanager.model</a>	0 sec	0		5	
<a href="#">uk.co.blackpepper.masterdatamanager.model.impl</a>	0 sec	0		38	+2
<a href="#">uk.co.blackpepper.masterdatamanager.persistence.ipa</a>	4 sec	0		27	
<a href="#">uk.co.blackpepper.masterdatamanager.service</a>	0 sec	0		38	
<a href="#">uk.co.blackpepper.masterdatamanager.shared.model</a>	0 sec	0		28	+5
<a href="#">uk.co.blackpepper.util.support</a>	0 sec	0		3	

# Agenda

---

Practical tools and techniques that help us to make our iterations shorter...

- ❖ Why shorter iterations?
- ❖ Improving inefficient practices in typical projects make them better :
  - code review vs automated static analysis
  - manual vs automated unit testing
  - **by product - test coverage**
  - manual vs automated acceptance testing
  - manual vs automated deployment
  - by product - the value of metrics

# Test Coverage Analysis

---

- ❖ Probably done less than unit testing
  - Has to be automated
  - Ensures quality of unit testing is measurable
    - Avoids tests that just return success!
    - Ensures that all paths are executed
  - Instruments code before tests are run
  - Tools have been around for some time
    - Commercial: McCabe, LDRA, Purify
    - Open Source: Cobertura

# Test Coverage Analysis

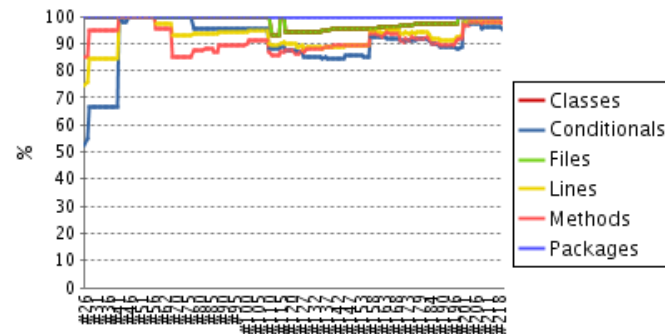
---

## ❖ Benefits

- Understand the quality of unit tests
- Identify code that requires additional testing
- Generates reports
- Can be added to the build pipeline
- Builds can be failed for insufficient coverage
  - Set target class, line and branch coverage

# Coverage Metrics

## Trend



## Project Coverage Summary

Name	Packages	Files	Classes	Methods	Lines	Conditionals
Cobertura Coverage Report	100% (7/7)	100% (45/45)	100% (47/47)	97% (261/268)	97% (927/952)	95% (303/320)

## Coverage Breakdown by Package

Name	Files	Classes	Methods	Lines	Conditionals
<a href="#">uk.co.blackpepper.masterdatamanager.service</a>	100% (3/3)	100% (3/3)	100% (16/16)	99% (134/136)	84% (32/38)
<a href="#">uk.co.blackpepper.masterdatamanager.model.impl</a>	100% (12/12)	100% (13/13)	99% (106/107)	99% (397/399)	98% (210/214)
<a href="#">uk.co.blackpepper.masterdatamanager.shared.service</a>	N/A	N/A	N/A	100% (0/0)	100% (0/0)
<a href="#">uk.co.blackpepper.masterdatamanager.gwt.service</a>	100% (4/4)	100% (4/4)	76% (16/21)	79% (49/62)	80% (8/10)
<a href="#">uk.co.blackpepper.masterdatamanager.persistence</a>	N/A	N/A	N/A	100% (0/0)	100% (0/0)
<a href="#">uk.co.blackpepper.masterdatamanager.model</a>	100% (9/9)	100% (9/9)	100% (18/18)	100% (41/41)	100% (0/0)
<a href="#">uk.co.blackpepper.util.support</a>	100% (1/1)	100% (1/1)	100% (4/4)	100% (5/5)	100% (0/0)
<a href="#">uk.co.blackpepper.util</a>	N/A	N/A	N/A	100% (0/0)	100% (0/0)
<a href="#">uk.co.blackpepper.masterdatamanager.shared.model</a>	100% (9/9)	100% (10/10)	99% (77/78)	98% (204/209)	89% (39/44)
<a href="#">uk.co.blackpepper.masterdatamanager.persistence.ipa</a>	100% (7/7)	100% (7/7)	100% (24/24)	97% (97/100)	100% (14/14)

# Agenda

---

Practical tools and techniques that help us to make our iterations shorter...

- ❖ Why shorter iterations?
- ❖ Improving inefficient practices in typical projects make them better :
  - code review vs automated static analysis
  - manual vs automated unit testing
  - by product - test coverage
  - manual vs automated acceptance testing
  - manual vs automated deployment
  - by product - the value of metrics

# Functional/Acceptance Testing

---

- ❖ Traditionally testers manually run test scripts
  - Error Prone
  - Slow
  - Expensive for large systems
  - Lack of repeatability
  - Low job satisfaction for testers
  - Can cause friction between tester and developer
  - Cannot be added to build pipeline

# Automating Functional Testing

---

- ❖ Tools have been around for some time
  - Commercial
    - Rational Robot/Test Manager
    - Functional Tester
  - Open Source
    - Selenium
- ❖ Not a substitute for manual ad-hoc tests

# Automating Functional Testing

---

- ❖ Test design is still important
  - Build an effective testing model
    - Isolate UI specifics from functionality
  - Tests must be readable by testers
  - Better if testers can write the tests
    - Consider using a DSL for test development

# Automated Functional Testing

---

## ❖ Benefits

- Repeatable
- Maintains the full history of events for analysis
- Can verify the application's basic functionality
  - Leaves testers to focus on more interesting edge cases
- Can be added to the build pipeline
- Gives immediate feedback on application status

# Agenda

---

Practical tools and techniques that help us to make our iterations shorter...

- ❖ Why shorter iterations?
- ❖ Improving inefficient practices in typical projects make them better :
  - code review vs automated static analysis
  - manual vs automated unit testing
  - by product - test coverage
  - manual vs automated acceptance testing
  - manual vs automated deployment
  - by product - the value of metrics

# Deployment

---

- ❖ Short iterations mean frequent deployment
- ❖ Automation is required for consistency
  - Use the artefacts created by the build pipeline
  - Promote builds that pass functional testing
  - Use common scripts for all environments
  - Re-run functional tests
  - Add performance tests for pre-production
- ❖ Well proven by go live

# Agenda

---

Practical tools and techniques that help us to make our iterations shorter...

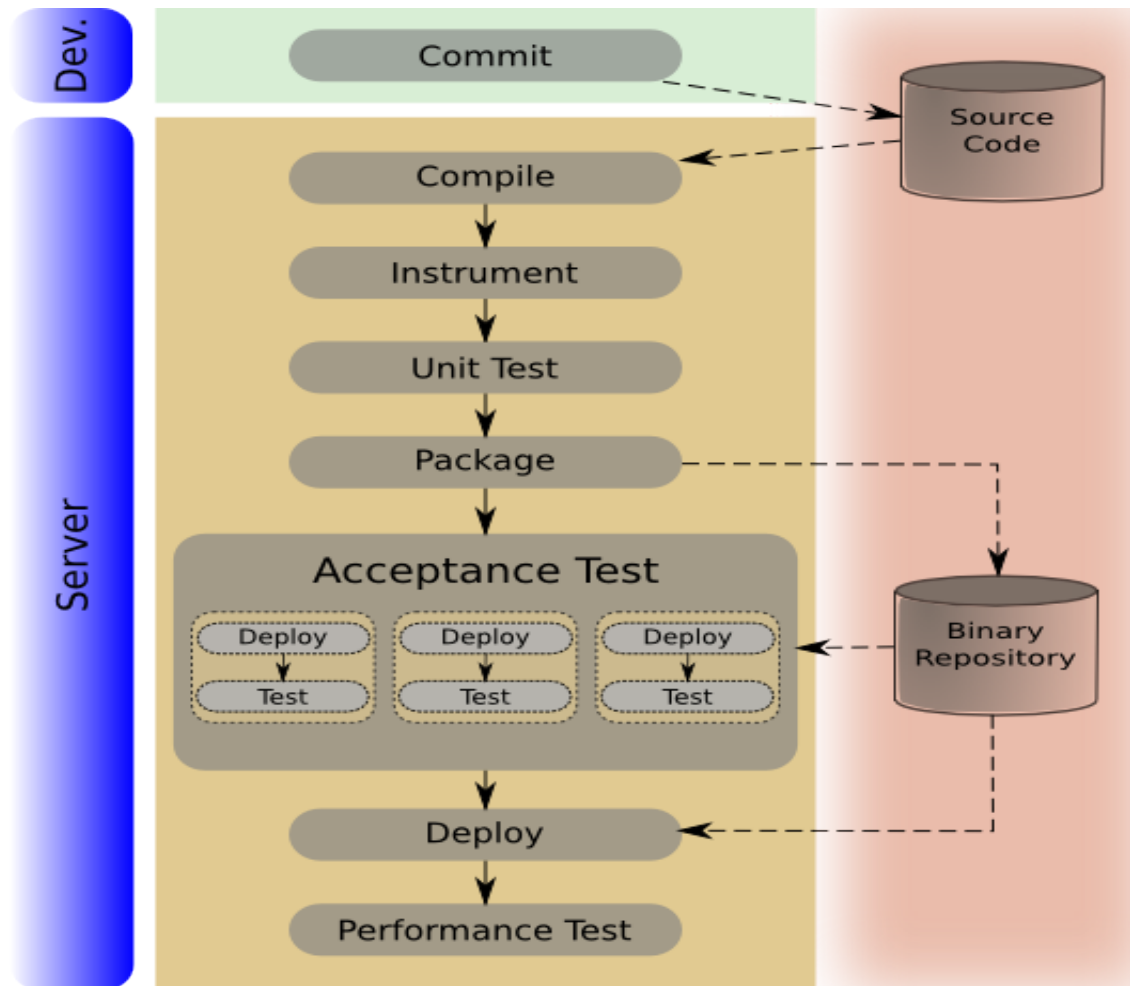
- ❖ Why shorter iterations?
- ❖ Improving inefficient practices in typical projects make them better :
  - code review vs automated static analysis
  - manual vs automated unit testing
  - by product - test coverage
  - manual vs automated acceptance testing
  - manual vs automated deployment
  - by product - the value of metrics

# Automated Build Tools

---

- ❖ Monitor updates to project repositories
  - Push build down a pipeline
    - Successful step moves build to next stage of pipeline
  - Provides notifications of progress and status
  - Provides status of system build via a dashboard
    - Maintains build history
    - Retains build artefacts, logs, test results

# Build Pipelines

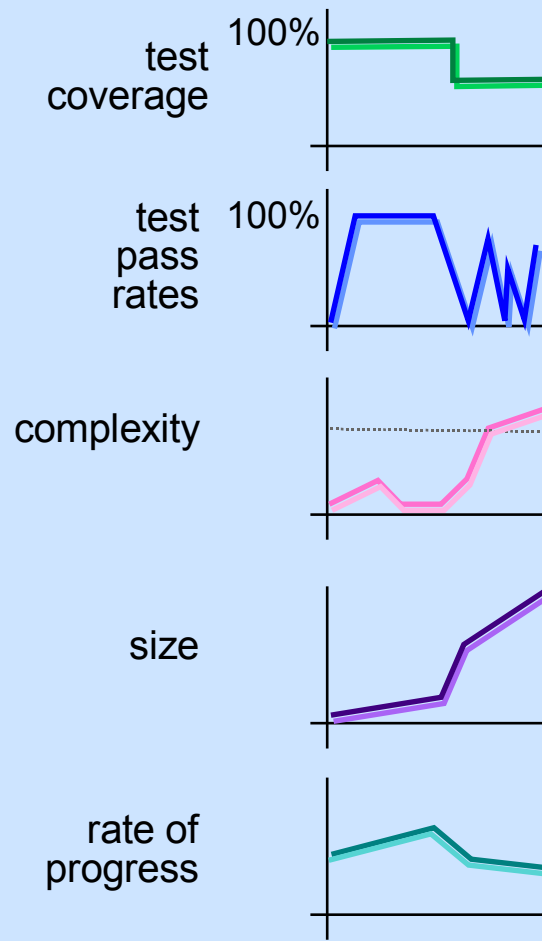


# Analysis of Metrics

## Simple Metrics

test cycle a of b  
% scripts passing  
defect and fix trends  
% duration used

e.g.  
test cycle 3 of 4  
pass rate 70%  
5 defects open, 20  
closed, finding 5 /  
day  
3/4 through  
schedule



## Sophisticated Trends

simple metrics plus trends  
over time of ...

unit test coverage  
unit test pass rates  
acceptance test coverage  
acceptance test rates  
number of check ins  
successful builds  
regression errors  
size  
complexity  
speed of progress  
change in requirements

# Is a one week iteration possible?

---

?

# Thank You and Questions

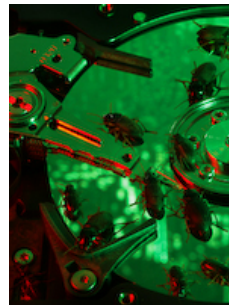
---



**Software  
Architecture**



**Agile  
Coaching**



**Project  
Recovery**



**Application  
Build**



**Troubleshooting,  
Performance  
& Scalability**



**Open Source  
Savings**