CM Crossroads Webcast Series

Lean Release Management

Moderator

Megan O'Meara
Editor – CM Crossroads

Speakers

Martin Fowler - Chief Scientist, ThoughtWorks

Jez Humble - Cruise Product Manager, ThoughtWorks

cm//crossroads™
The configuration management community

Martin Fowler

Martin Fowler is the Chief Scientist at ThoughtWorks. A part of ThoughtWorks since 1999, Martin Fowler is a renowned international speaker on software architecture, specializing in object-oriented analysis and design, UML, patterns, and agile software development methodologies. Martin Fowler started working with software in the early 80's and has written five popular books on the topic of software development. He has also served on program committees for OOPSLA, Software Development, UML World, XP 2001, and TOOLS.

Jez Humble

Jez is the Product Manager for Cruise, ThoughtWorks's new Continuous Integration and Release Management software. As the founding leader of ThoughtWorks' build and deployment community, he has worked hard to capture and propagate best practices in the build, test and deployment space. His goal is to make Cruise the number one commercial continuous integration and release management server.

Jez has over eight years of professional experience in IT as a developer, system administrator, trainer and manager. He has worked with a variety of platforms and technologies, consulting for non-profits, telecoms, financial service and insurance service organizations. Since 2004 he has worked for ThoughtWorks in London and Bangalore and currently lives in Beijing. He holds a BA in Physics and Philosophy from Oxford University and an MMus in Ethnomusicology from the School of Oriental and African Studies, University of London.

Lean Release Management

Martin Fowler

Jez Humble

28 July 2008

thoughtworks.com/cruise

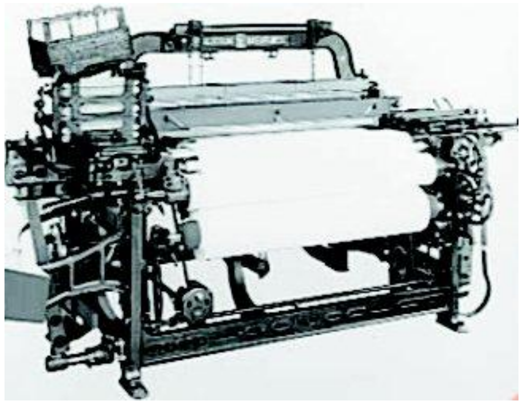
What will we talk about?

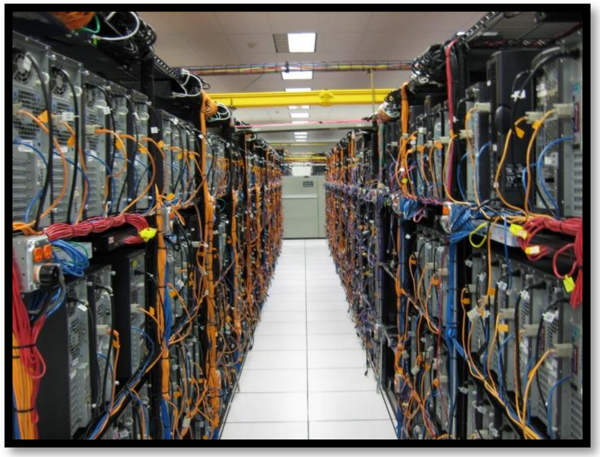
What is release management? What is lean?

Principles of lean applied to release management.

Practices derived from the principles.

Cruise and how it helps.





What is release management?

Getting software in a state where you are making money from it

As opposed to functionally complete and tested

Problem:

You’re not done and you don’t know when you will be

Your software isn’t earning you money

Your developers aren’t working on something new

What is lean?

Toyota Production System originally developed by Toyota:

"All we are doing is looking at the time line, from the point the customer gives us an order to the point when we collect the cash. And we are reducing the time line by reducing the non-value-added wastes."

– Taiichi Ohno

Invented to manage complex processes;

Toyota Product Development System.

Why should you care?

Lean product development is designed to optimise for:

- Frequent releases
- Short development time
- Adaptability in design, schedule and cost targets
- Continuous improvement in quality, process, productivity and development time

Release management often *appears* to be the highest risk activity

Seven principles of lean software development

Eliminate waste

Build quality in

Amplify knowledge

Defer commitment

Deliver fast

Respect people

Optimise the whole

Taken from *Lean Software Development: An Agile Toolkit*, Poppendieck (Addison Wesley, 2003)

Eliminate waste

If it isn’t delivering value to your users, it isn’t done

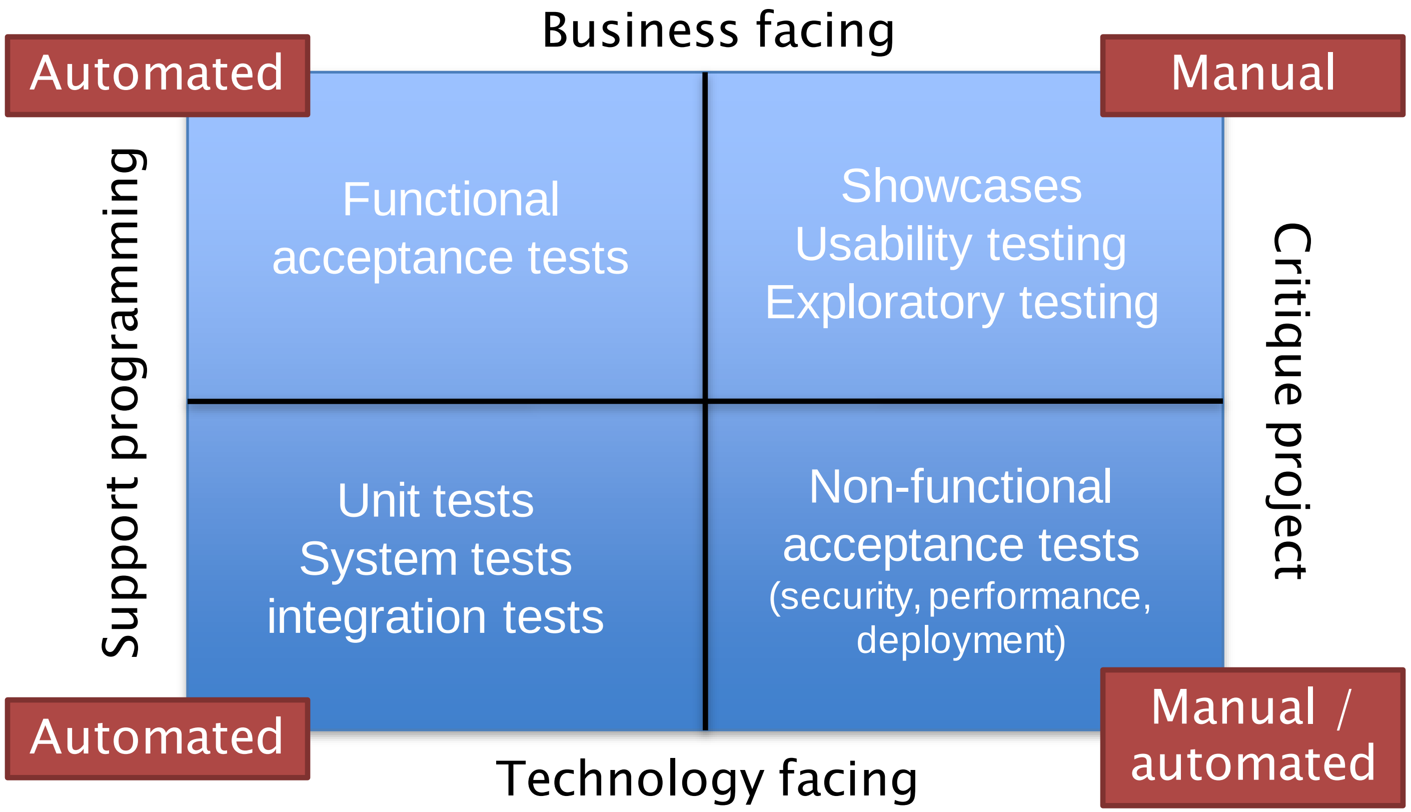
First find waste: map your value stream

Look for delays and queues

Anything that gets in the way of *deployed* code

How long is your cycle time?

Build quality in



From Brian Marick http://www.exampler.com/old-blog/2003/08/21/#agile-testing-project-1, via Poppendieck, *Implementing Lean Software Development* (Addison Wesley, 2006), ch. 8.

Amplify knowledge

Get feedback as soon as possible

Involve as many people as is reasonable

The cost of change is less when done sooner

– But be careful to avoid churn

Defer commitment

Make irreversible decisions at the last responsible moment

Make as many of your decisions easy to reverse as possible

Requires **information** and **confidence**

Deliver fast

Maximise NPV on your investment

Get feedback quickly and adjust

Businesses change fast

Respect people (over process)

There is no process that can’t be improved

Everybody is responsible for getting software released

Involve all stakeholders from the start

Trust your team

Provide entrepreneurial leadership

Optimise the whole

Only measure globals: **cycle time, margin, user satisfaction**

The only goal that matters is **delivered business value**

Everybody needs to have a stake

Eliminate waste: practices

Automate your deployment process pragmatically

Deploy frequently – if it hurts do it more

Check in no less than once a day and ensure a successful build

Only build your binaries once – separate out configuration

Anti-patterns

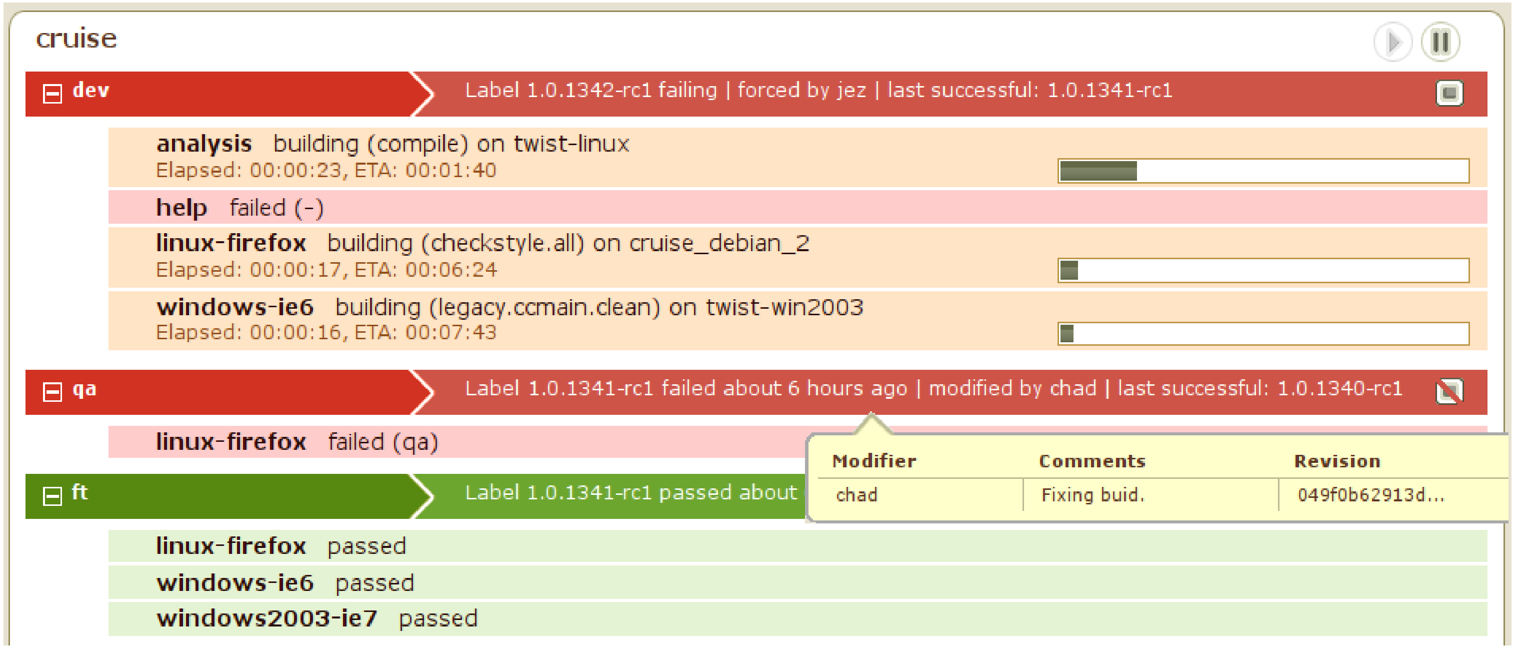
Trying to automate past the point of diminishing returns

There are exceptions – C++ compilers, embedded hardware with limited resources, dynamic languages

Eliminate waste

See the moment a defect occurs

See who did it



cruise

dev — Label 1.0.1342-rc1 failing | forced by jez | last successful: 1.0.1341-rc1

analysis building (compile) on twist-linux
Elapsed: 00:00:23, ETA: 00:01:40

help failed (-)

linux-firefox building (checkstyle.all) on cruise_debian_2
Elapsed: 00:00:17, ETA: 00:06:24

windows-ie6 building (legacy.ccmain.clean) on twist-win2003
Elapsed: 00:00:16, ETA: 00:07:43

qa — Label 1.0.1341-rc1 failed about 6 hours ago | modified by chad | last successful: 1.0.1340-rc1

linux-firefox failed (qa)

Modifier	Comments	Revision
chad	Fixing buid.	049f0b62913d...

ft — Label 1.0.1341-rc1 passed about

linux-firefox passed

windows-ie6 passed

windows2003-ie7 passed

thoughtworks.com/cruise

Build quality in: practices

Do continuous integration

Smoke-test your deployments

Make it easy to get binaries

Automate build, test and deployment

Make it easy to see what's happening

Standardise the process of making changes

Anti-patterns

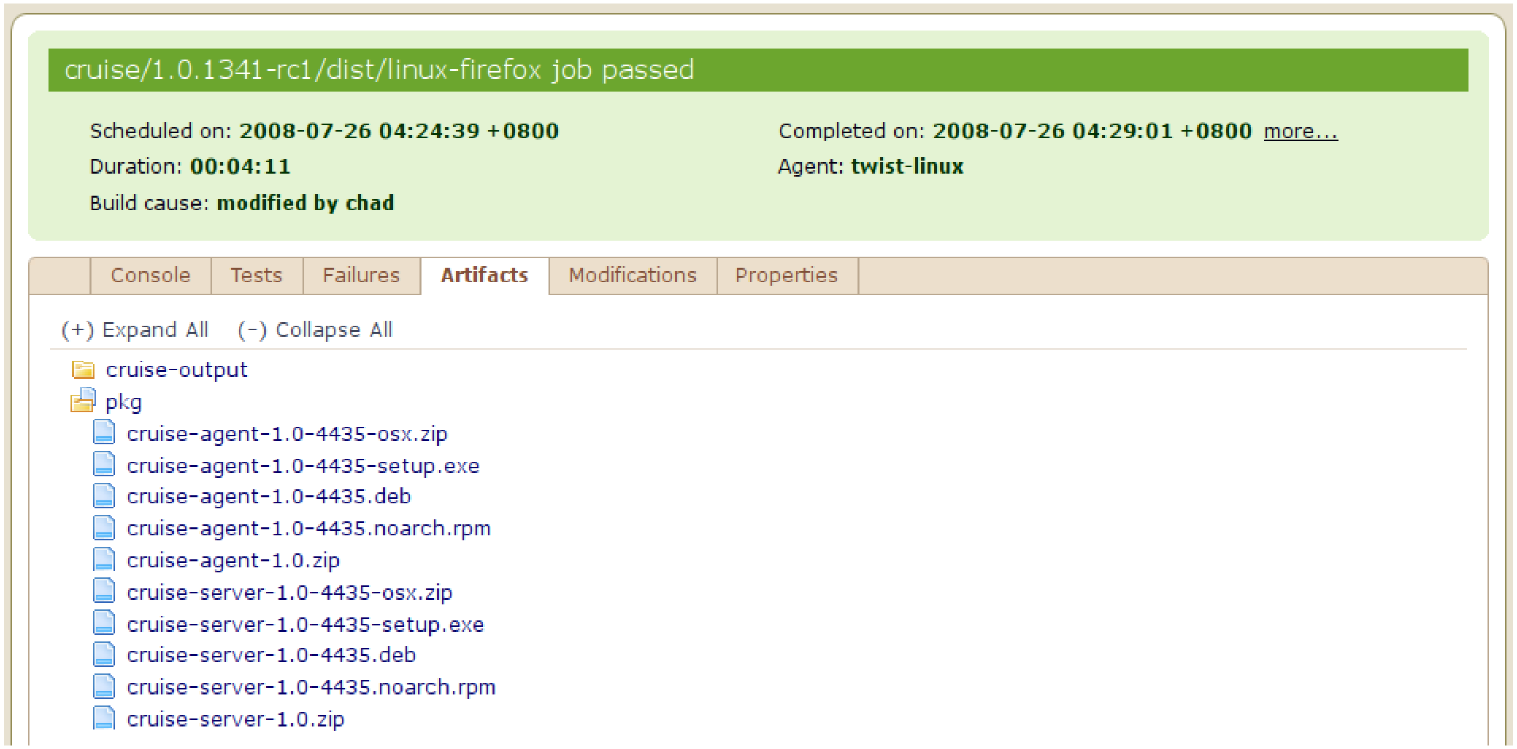
Fragile tests

Mock overload (testing behaviour, not intent)

Missing part of the quadrant

Build quality in

Testers can find and get good installers



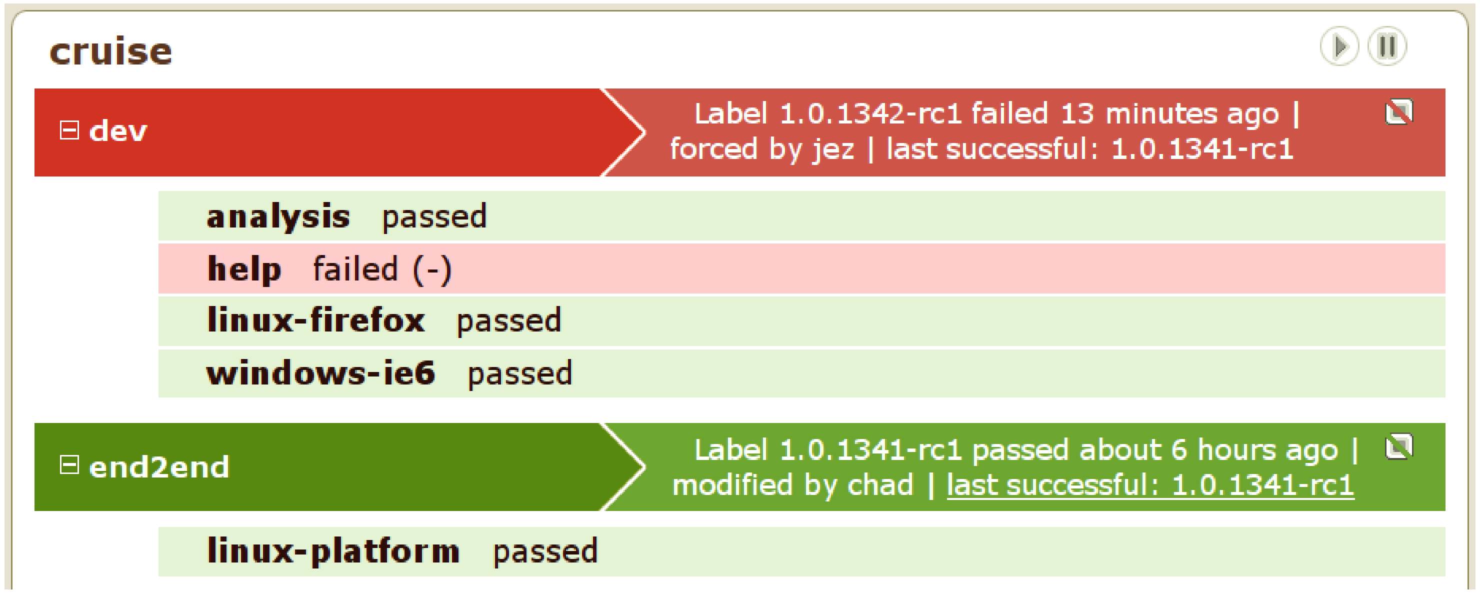
cruise/1.0.1341-rc1/dist/linux-firefox job passed

Scheduled on: **2008-07-26 04:24:39 +0800**
Completed on: **2008-07-26 04:29:01 +0800** more...
Duration: **00:04:11**
Agent: **twist-linux**
Build cause: **modified by chad**

- cruise-output
- pkg
  - cruise-agent-1.0-4435-osx.zip
  - cruise-agent-1.0-4435-setup.exe
  - cruise-agent-1.0-4435.deb
  - cruise-agent-1.0-4435.noarch.rpm
  - cruise-agent-1.0.zip
  - cruise-server-1.0-4435-osx.zip
  - cruise-server-1.0-4435-setup.exe
  - cruise-server-1.0-4435.deb
  - cruise-server-1.0-4435.noarch.rpm
  - cruise-server-1.0.zip

Build quality in

Cruise as *andon* (information radiator)



cruise

⊟ dev
Label 1.0.1342-rc1 failed 13 minutes ago | forced by jez | last successful: 1.0.1341-rc1

analysis passed
help failed (-)
linux-firefox passed
windows-ie6 passed

⊟ end2end
Label 1.0.1341-rc1 passed about 6 hours ago | modified by chad | last successful: 1.0.1341-rc1

linux-platform passed

Amplify knowledge: practices

Maintain a single source repository

Start deploying in a production-like environment from day one

Have a support and an operations person on your team

Get useful metrics: cyclomatic complexity + code coverage, npath complexity, duplicates, efferent and afferent couplings

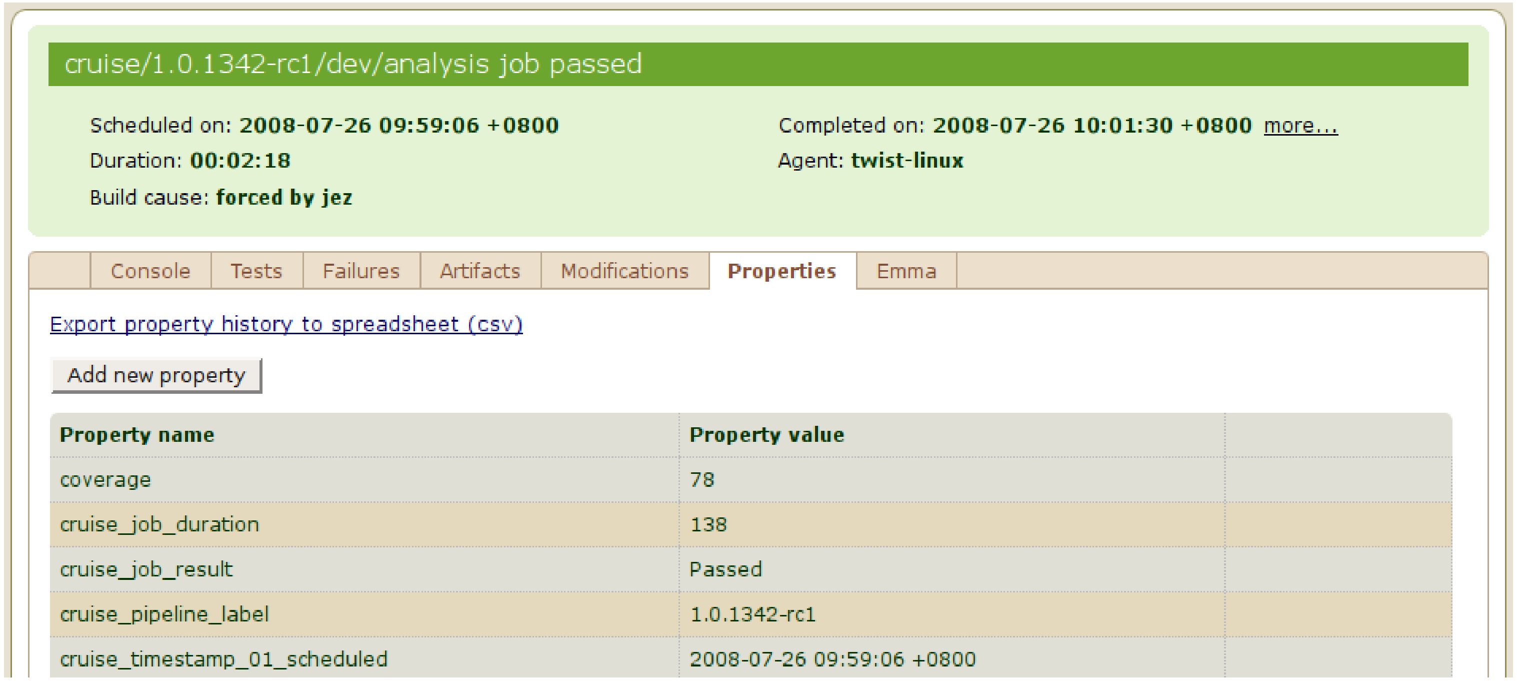
Fail the build or warn when metrics get worse

Anti-patterns

Useless metrics: lines of code, number of classes

Amplify knowledge

See metrics



cruise/1.0.1342-rc1/dev/analysis job passed

Scheduled on: **2008-07-26 09:59:06 +0800**
Duration: **00:02:18**
Build cause: **forced by jez**
Completed on: **2008-07-26 10:01:30 +0800** more...
Agent: **twist-linux**

Export property history to spreadsheet (csv)

Property name	Property value	
coverage	78	
cruise_job_duration	138	
cruise_job_result	Passed	
cruise_pipeline_label	1.0.1342-rc1	
cruise_timestamp_01_scheduled	2008-07-26 09:59:06 +0800	

thoughtworks.com/cruise

Defer commitment: practices

It should be possible to easily back out every change

Use blue/green deployments or virtual machines

I should know exactly what's in every environment:

Universal configuration management

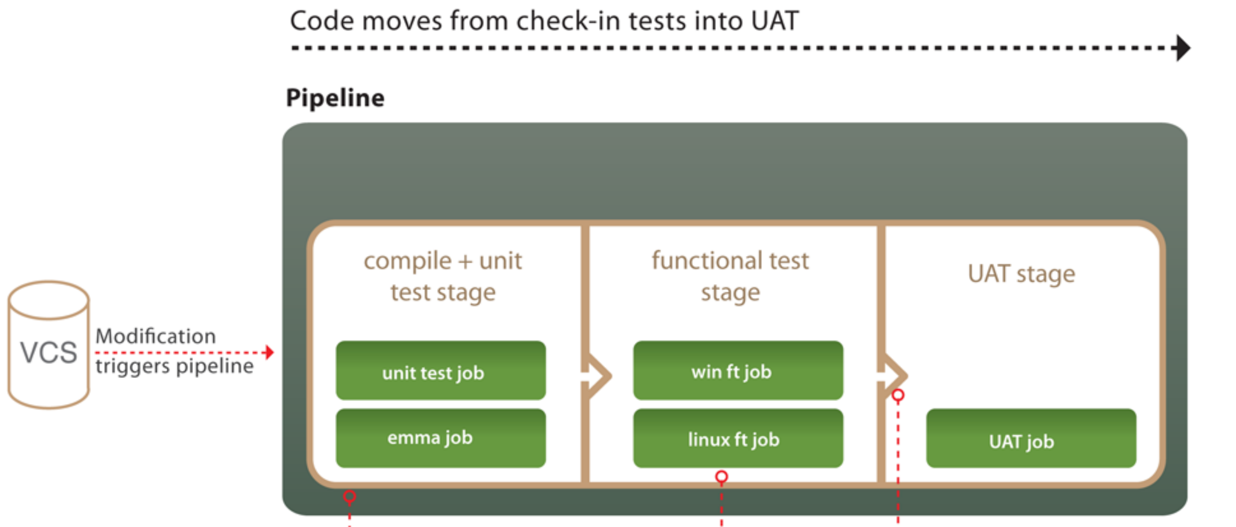
Anti-patterns

Ultimate configurability

Locking down configuration

Undocumented configuration

Deployment pipelines


Code moves from check-in tests into UAT
Pipeline
VCS
Modification triggers pipeline
compile + unit test stage
unit test job
emma job
functional test stage
win ft job
linux ft job
UAT stage
UAT job

Approval
- Can be automatic or manual

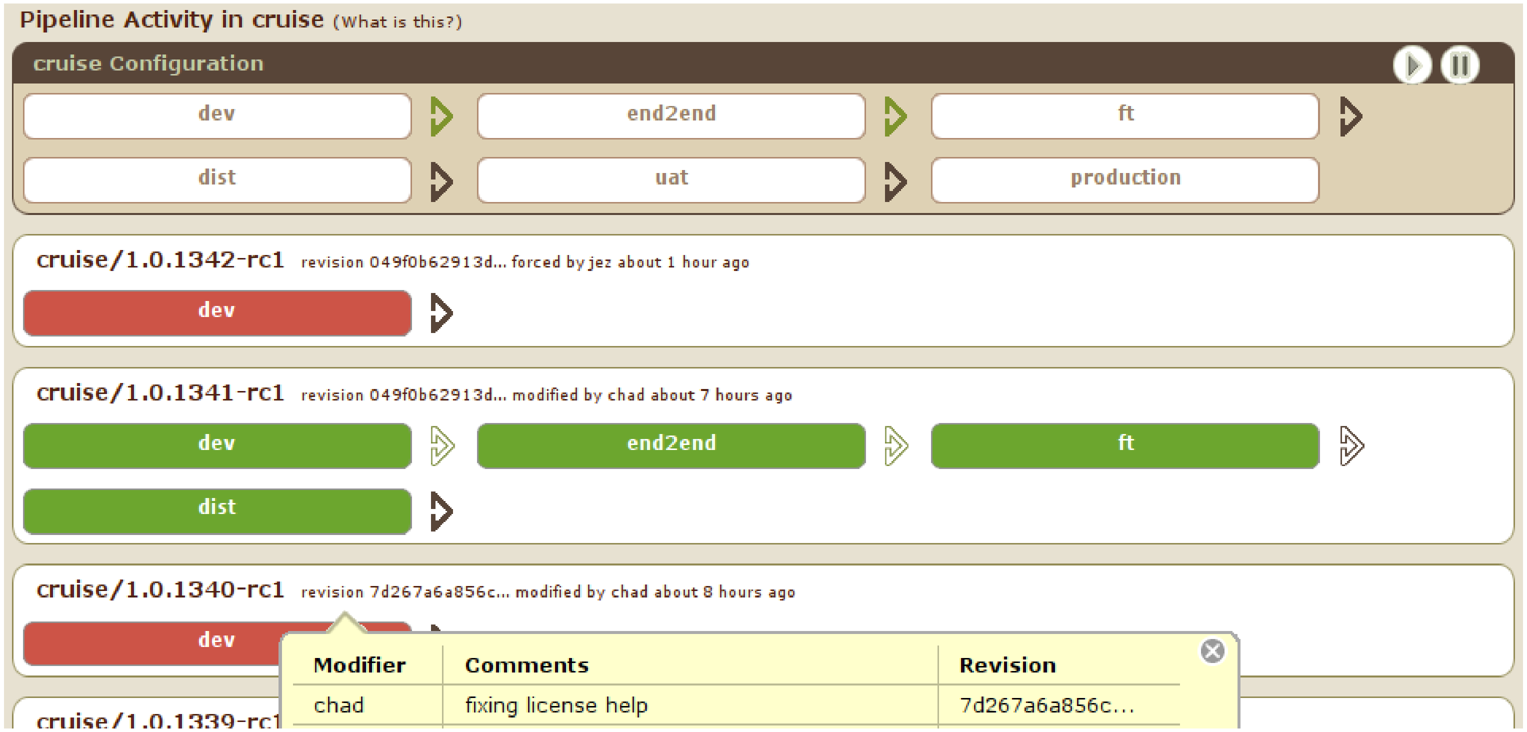
Job
- Jobs run concurrently within a stage
- If a job fails, the stage it is in fails
- Each job plan runs one or more targets (ant, nant, or ecec)
- Jobs can run in Parallel within a stage if you have multiple agents

Stage
- Stages run consecutively
- Each stage is triggered automatically by the successful completion of the previous stage
- Can also be triggered manually

Deployment pipelines

See code go through UAT, performance testing, staging...

...and even into production



Pipeline Activity in cruise (What is this?)

cruise Configuration

dev ▷ end2end ▷ ft ▷

dist ▷ uat ▷ production

cruise/1.0.1342-rc1 revision 049f0b62913d... forced by jez about 1 hour ago

dev ▷

cruise/1.0.1341-rc1 revision 049f0b62913d... modified by chad about 7 hours ago

dev ▷ end2end ▷ ft ▷

dist ▷

cruise/1.0.1340-rc1 revision 7d267a6a856c... modified by chad about 8 hours ago

dev

Modifier	Comments	Revision
chad	fixing license help	7d267a6a856c...

cruise/1.0.1339-rc1

thoughtworks.com/cruise

Deliver fast: practices

Constant, dependable heartbeat of rapid, regular releases

The more often you release, the fewer changes between releases

Keep your feedback cycles fast: parallelise tests, use VMs, grids

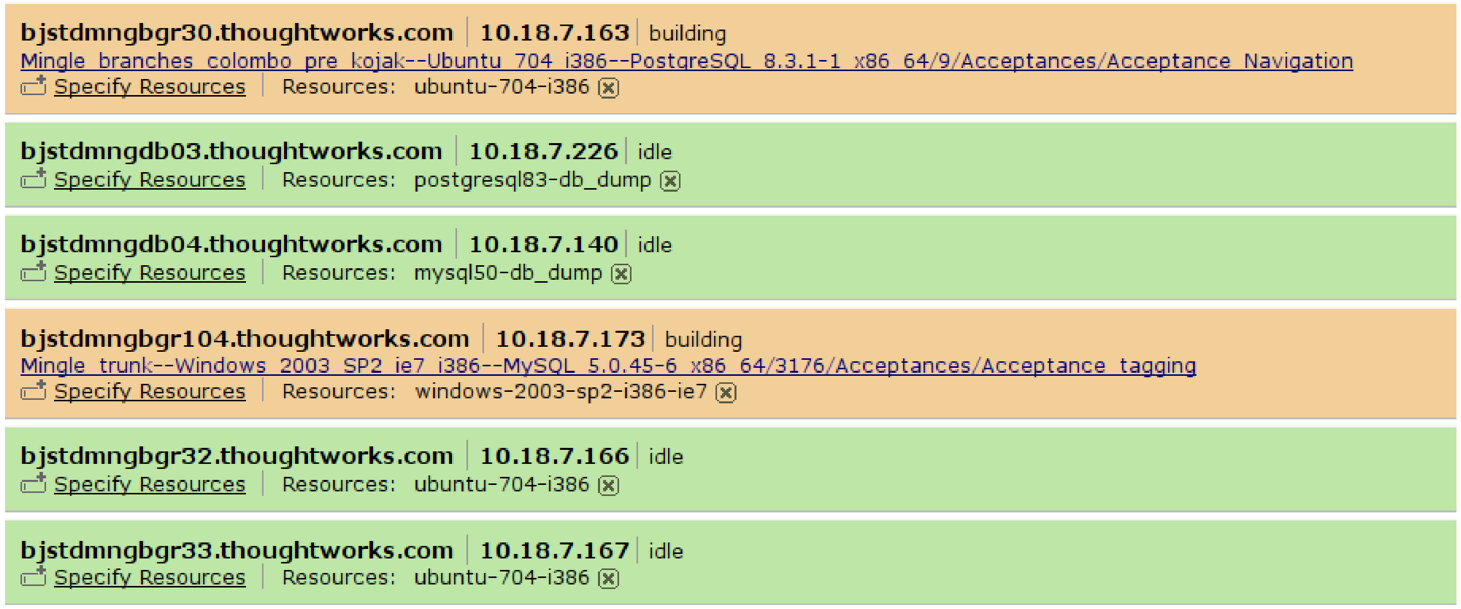
Anti-patterns:

Lots of builds, but no traceability or synchronization

Not listening to the build

Deliver fast

Zero configuration build cloud



bjstdmngbgr30.thoughtworks.com | 10.18.7.163 | building
Mingle_branches_colombo_pre_kojak--Ubuntu_704_i386--PostgreSQL_8.3.1-1_x86_64/9/Acceptances/Acceptance_Navigation
Specify Resources | Resources: ubuntu-704-i386

bjstdmngdb03.thoughtworks.com | 10.18.7.226 | idle
Specify Resources | Resources: postgresql83-db_dump

bjstdmngdb04.thoughtworks.com | 10.18.7.140 | idle
Specify Resources | Resources: mysql50-db_dump

bjstdmngbgr104.thoughtworks.com | 10.18.7.173 | building
Mingle_trunk--Windows_2003_SP2_ie7_i386--MySQL_5.0.45-6_x86_64/3176/Acceptances/Acceptance_tagging
Specify Resources | Resources: windows-2003-sp2-i386-ie7

bjstdmngbgr32.thoughtworks.com | 10.18.7.166 | idle
Specify Resources | Resources: ubuntu-704-i386

bjstdmngbgr33.thoughtworks.com | 10.18.7.167 | idle
Specify Resources | Resources: ubuntu-704-i386

thoughtworks.com/cruise

Respect people: practices

Create a deployment strategy at the beginning of the project

Get all stakeholders together in a room to participate

Improve your process continuously

Anti-patterns

Imposed corporate processes and procedures

Release management is “not my problem”

Release management as a line item

Optimise the whole

Only count work that has been showcased from staging as done

Non-functional testing through instrumentation and incremental delivery

Manage changes with automation

Anti-patterns

Silos – incentives not aligned with business value

Manage changes with heavy process

Maturity model

0. Version control everything
1. Automated build / continuous compilation
2. Automated unit tests
3. Automated functional tests
4. Automated deployment to UAT / performance testing etc.
5. Automated deployment to staging and production

Summary

Principles of Lean

Practices derived from the principles

Key thoughts:

- If it isn’t delivering value to your users, it isn’t done
- Involve everyone from the beginning
- Continually improve your process: PDSA
- If it hurts, do it more: deploy often, release often
- Be able to revert every change

Thank you!

Download it today for free: http://thoughtworks.com/cruise

Thanks to: Richard Durnall, Jann Thomas, John Johnston, Michael Robinson, Jason Yip, Chris Leishman

Questions and Answers

Please post your questions now using the “Ask a Question” box on left side of the screen

And the Winner is...

View other Webcasts in the Series at www.cmcrossroads.com/wc

cm//crossroads The configuration management community