IBM Software

Innovate2012

The Premier Event for Software and Systems Innovation

Next NOW!

Waterfall to Agile for CICS TS for z/OS – the continuing journey

Nigel Hopper
IBM CICS TS for z/OS Managed Platform Focus Team Leader
Quality Software Engineering Development Top Gun

100 - CICS

30 billion transactions/day, >$300B/week

40+ ~~35~~ years invested in applications

16,000 customers worldwide

CICS

30 million users

950,000 programmers earn their living from CICS

First GA'd when:
- *Nixon* was president
- Man *landed* on the moon

Over 900,000 concurrent users/system

One of the top 35 technologies that shaped the industry*

Used by 490 + of IBM''s top 500 customers

5000 packages from 2000 ISVs

http://www.ibm.com/ibm100/us/en/icons/cics/

50,000 CICS licenses

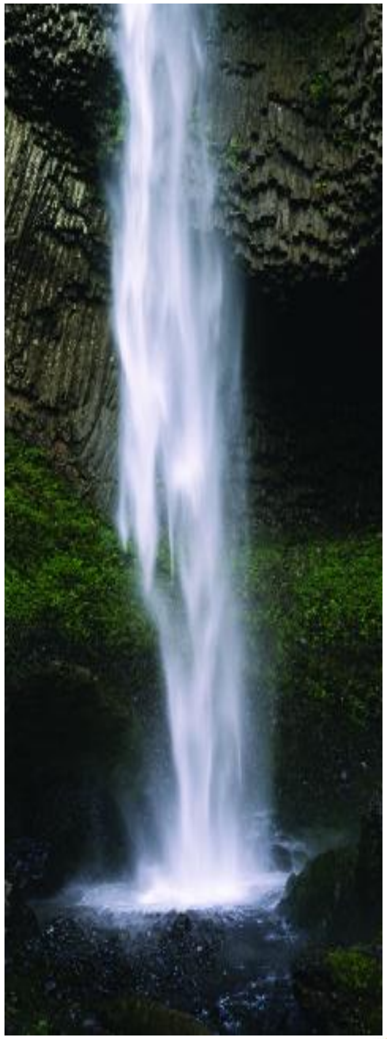
*According to Computerworld magazine

CICS Transaction Server code *is* complex!

- Started out over 40 years ago as a loose collection of programs
- Primarily written in assembler, PL/X and some COBOL
  - Now has Java, XML and more!
- Developed on the mainframe for the mainframe – z/OS!
- Eventually converted to domains
  - Currently in the region of 70 domains. Grouped in areas such as Application Services, Business Logic Applications, Base Runtime, CPSM
  - 3 APIs
  - Multiple tools such as CICS Explorer, CICS Deployment Assistant

The CICS process until 2005

- 2 year release cycle
- Upfront commitment for the release
- Typical waterfall oriented cycle
- The issues with this approach
  - Up front commitment can be very inflexible
  - Work sized at – 'what will hopefully fit'
  - Large overheads in project management and managing change
  - Coordination and scheduling of cross team work – different priorities
  - Typical late in release integration test and beta issues
  - Defect backlog would hit a peak of 600+ defects
  - A lot of post release tidy up – Finishing function, improving quality



Why Agile?

- IBM were sufficiently confident of it to make it the corporate direction for developing software
- “If CICS can do it, anyone can” message
  - Little skills or knowledge
  - Still a 2 year release cycle
  - Who's benefit is this for?
- In 2005 our CICS release became iterative
  - Still a 2 year release cycle
  - With 4 month iterations
- The intention was:
  - Work to be broken down into smaller 'chunks'
  - All development, functional testing, defects and docs done in iteration
  - System/Integration testing done in following iteration
  - Beta every 4 months

Initial Agile adoption - Reality

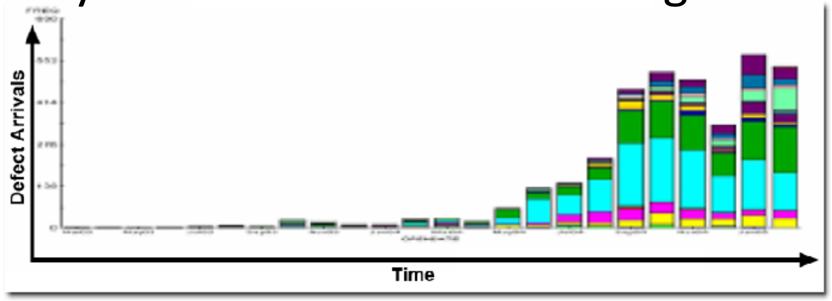
- Still 2 year upfront commitment
- 'Difficult' cultural change
- Difficult to contain work to 4 months
- Tooling fragmented and not Agile 'friendly'
  - 10+ day code promotion 'freeze' prior to beta shipping!
  - No way of integrating and prioritizing. For example:
    - Defects in one tool
    - Project tracking several others
- While the Agile term was used, reality was mini-waterfalls

Initial Agile adoption – benefits

- Beta shipped every 4 months
- System/integration testing done much earlier
- Defects found and handled much earlier – Peaked at 450
- Changed people's perception of what was possible – Still much skepticism
- Feedback on quality in the release has been high

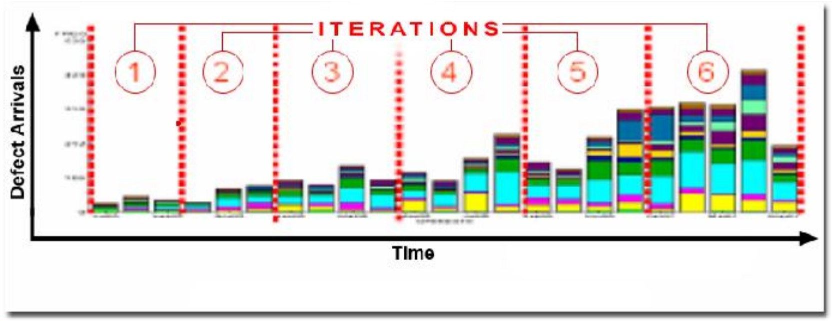
Waterfall Profile
Defects found later when they are more expensive to fix

CICS TS for zOS
Last waterfall delivery



Agile Profile
defects found early when they are cheaper to fix

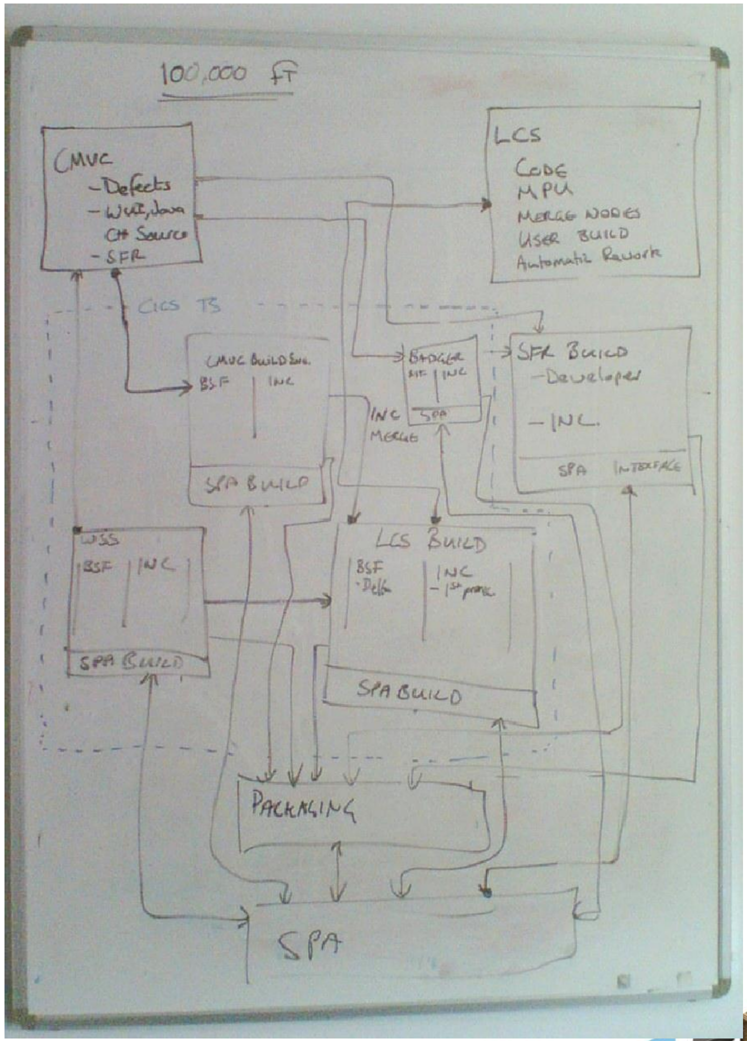
CICS TS for zOS
First 'Agile' delivery



Defect Raise Rate

2008/2009 Internal review

- A review of tools and processes was undertaken
- Using a wide variety of tools to manage project, for example:
  - 3 source code management systems
    - Completely alien to each other
    - Separate builds for two of the SCMs
    - Highly integrated build into main SCM
  - External database for customer requirements
  - Lotus Notes databases/documents for:
    - Internal requirements
    - Actions
    - Risks
- Large disparity in the way many things were done and tracked
- High learning curve for new starters



The Vision

Single non-proprietary environment for the delivery and service of future releases of CICS Transaction Server for z/OS

Where everyone (business, marketing, development, test, service, build, etc) can focus and collaborate via one single tool

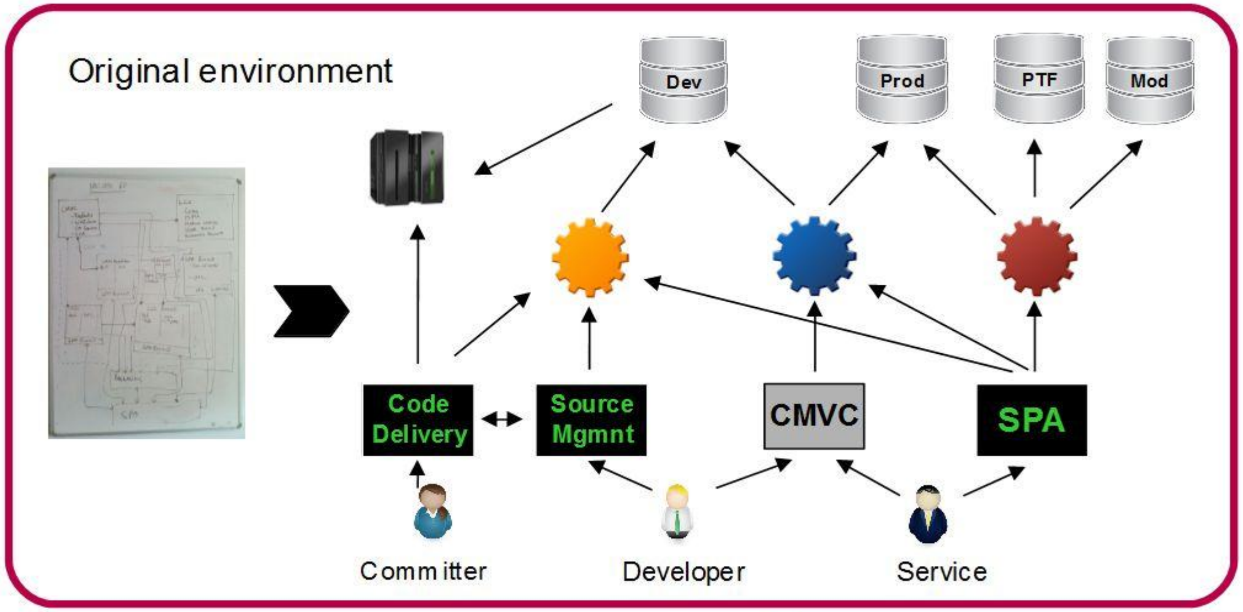
If we are really to adopt Agile principles, the tooling had to improve

Why RTC?

- A little luck was involved!
- Early trials showed great promise
- Hursley just started centrally hosting Jazz servers
- Highly configurable
  - We were in control!
- Provided great audit tracking
- Allowed for an end to end integrated development environment
  - Requirements, Approvals, Defects, Code, Tracking, Build and so on!
- Clearly a tool designed for Agile development

RTC Adoption

- First thing – June 2009 new release - 2 month iterations, 4 month betas
- Initial focus on work items, project planning and tracking
  - I had a number of key goals
    - Use RTC 'out of the box'
    - Use industry standard Agile methods/terms
    - Use a minimum number of roles – trust people
    - Use RTC to implement RTC
    - Big bang approach would not work
  - Review processes, set-up projects, migrate defects/requirements - July-October 2009
- Long term focus on migrating source code and service delivery
  - Rewrite the build – January 2010 – February 2011
  - Write a new code promotion tool – June 2010 – March 2011
  - Ship CICS TS for z/OS v4.2 from RTC – June 2011
  - Migrate existing SCM and new development environment – July 2011

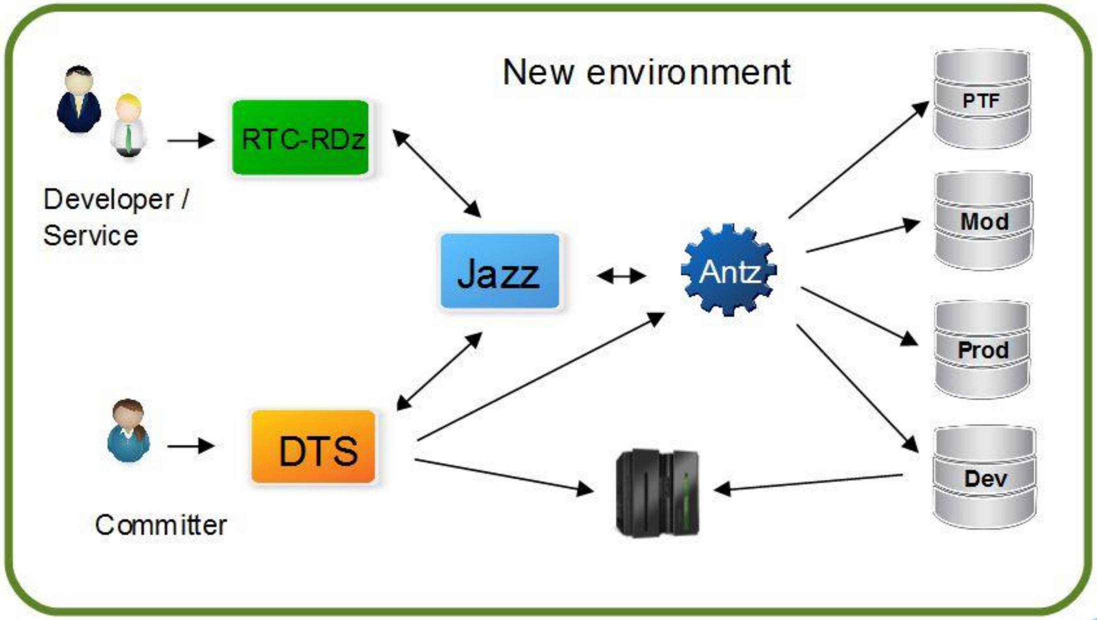


Environment Comparison

RTC/Jazz manages:

- Release/Iterations
- Work items
- Plans
- Source code
- Streams
- Change sets
- Lots more!

- Rational Team Concert v3
- Rational Developer for System Z v8
- Our Ant build technology is an Eclipse plug-in for build control
  - Hursley constructed build tool
  - Ant is an open source build tool
  - We added extensions to support zOS
- Antz has been shared with RTC and some aspects have been reused.



The issues...

- RTC was not the problem, does what it says on the box
  - There was a learning curve, but the basics were simple
  - We could not be where we are today without it.
- People issues
  - They do not want to change - Why change if it currently works?
  - Agile is a completely new mindset
  - Developers have used the same tool set for 20+ years
- Process issues
  - We had some serious holes in our process – still very waterfall
  - Need agreement of many senior people
  - Still learning the Agile way
- Understanding the existing build
  - 20+ years of integrating a build into a library system!
  - Any experience had retired!

The benefits...

- Integrated work item and reporting tools
  - Collaboration: Far greater collaboration across teams
  - Transparency: Of work items, dependencies and status
  - Integration: Status, plans, work items, e.g. Risks/Actions integrated in the same tool
  - Notification: Far greater notification when things change
  - Ability to ignore what is green and focus on the issues
- Responsiveness to business needs – aiming for greater value, earlier
  - Beta deliveries much earlier
  - Ability to adapt to change much easier
  - Current release – much more Agile in terms of requirements and prioritisation
- Dashboards – Instant status – Everyone has a much clearer view of project
  - Weekly status (Scrum of Scrums) – Practically no prep, meeting time halved
- End of iteration quality checks – 2-4 hours prep to < 30 mins, meeting time halved

The benefits... (cont.)

- Common tool set and skills for development and service
- Ability to cut new streams
  - For Risky development and when beta required
- Build supported by two people. Previous build supported by three.
- Quality has improved
  - Defects in RTC – defects part of the backlog
  - Pre-2005 defects peaked at 600+ - Waterfall
  - 2005-2009 defects focused on much earlier (450 peak) – Agile/Iterative
  - 2009-2011 again lower levels of defects (350 peak) – Agile/Iterative+RTC
- Far greater ability to use Agile practices
- Much reduced learning curve for the development environment
- Far less post-release 'churn' – much more in control

Over the last year...

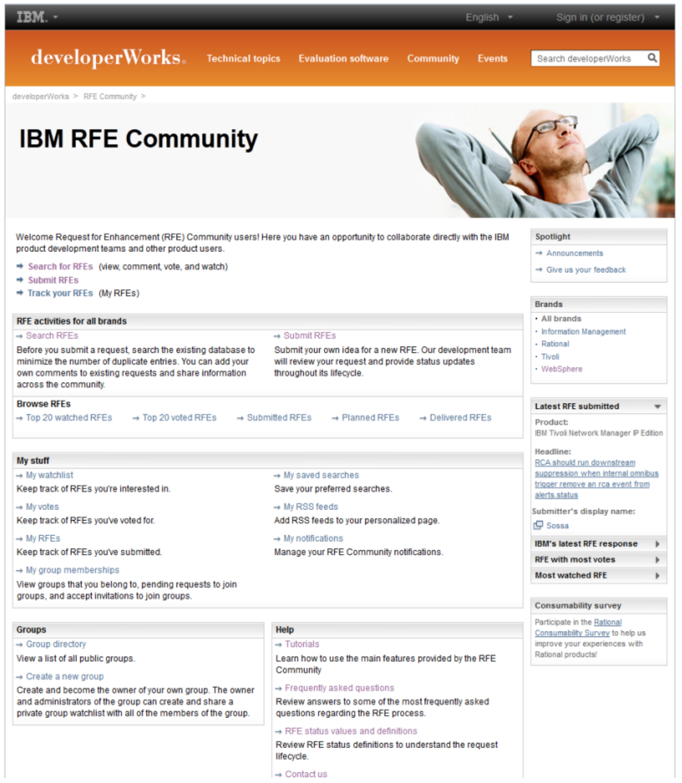
- Further adoption of Agile practices
- New customer requirements solution
- Portfolio planning with RRC and CLM
- Elaboration using RRC – Still have:
  - Designs in a Wiki
  - High level test plans in attachments
  - Information designs in the information centre
- Process issues to address

Further adoption of Agile practices

- Moved to 1 month iterations for most teams – shock to the system!
  - Does need a mindset change – chunking work even smaller
- Betas released every 2 months – alternating ISV and Customer betas
- Development team reorganization
  - Aim to have a broader set of skills in each team
  - Less of a silo approach
  - Large learning curve for less well known areas
- Direct access to customer requirements
- Iteration planning now more priority based with team driven commitment

New customer requirements solution

- developerWorks Request For Enhancement (dW RFE)
  - http://www.ibm.com/developerworks/rfe/
- The customer raises the requirement
- Bridge code synchronizes the requirement into an RTC project on the CICS server
- The RFE is assessed by CICS
  - Only status changes and #publish comments are synchronized back to the RFE
- Company and justification are private in dW RFE
- IBM can make the complete dW RFE private
- Public dW RFEs can be searched and commented on
- Public dW RFEs can also be voted on by other users
- First time we have been able to directly connect customer requirements to the work being developed



developerWorks > RFE Community >

IBM RFE Community

Welcome Request for Enhancement (RFE) Community users! Here you have an opportunity to collaborate directly with the IBM product development teams and other product users.

→ Search for RFEs (view, comment, vote, and watch)
→ Submit RFEs
→ Track your RFEs (My RFEs)

RFE activities for all brands

→ Search RFEs
Before you submit a request, search the existing database to minimize the number of duplicate entries. You can add your own comments to existing requests and share information across the community.

→ Submit RFEs
Submit your own idea for a new RFE. Our development team will review your request and provide status updates throughout its lifecycle.

Browse RFEs
→ Top 20 watched RFEs → Top 20 voted RFEs → Submitted RFEs → Planned RFEs → Delivered RFEs

My stuff

→ My watchlist
Keep track of RFEs you're interested in.

→ My saved searches
Save your preferred searches.

→ My votes
Keep track of RFEs you've voted for.

→ My RSS feeds
Add RSS feeds to your personalized page.

→ My RFEs
Keep track of RFEs you've submitted.

→ My notifications
Manage your RFE Community notifications.

→ My group memberships
View groups that you belong to, pending requests to join groups, and accept invitations to join groups.

Groups

→ Group directory
View a list of all public groups.

→ Create a new group
Create and become the owner of your own group. The owner and administrators of the group can create and share a private group watchlist with all of the members of the group.

Help

→ Tutorials
Learn how to use the main features provided by the RFE Community

→ Frequently asked questions
Review answers to some of the most frequently asked questions regarding the RFE process.

→ RFE status values and definitions
Review RFE status definitions to understand the request lifecycle.

→ Contact us

Spotlight
→ Announcements
→ Give us your feedback

Brands
- All brands
- Information Management
- Rational
- Tivoli
- WebSphere

Latest RFE submitted
Product:
IBM Tivoli Network Manager IP Edition
Headline:
RCA should run downstream suppression when internal omnibus trigger remove an rca event from alerts.status
Submitter's display name:
Sossa

IBM's latest RFE response
RFE with most votes
Most watched RFE

Consumability survey
Participate in the Rational Consumability Survey to help us improve your experiences with Rational products!

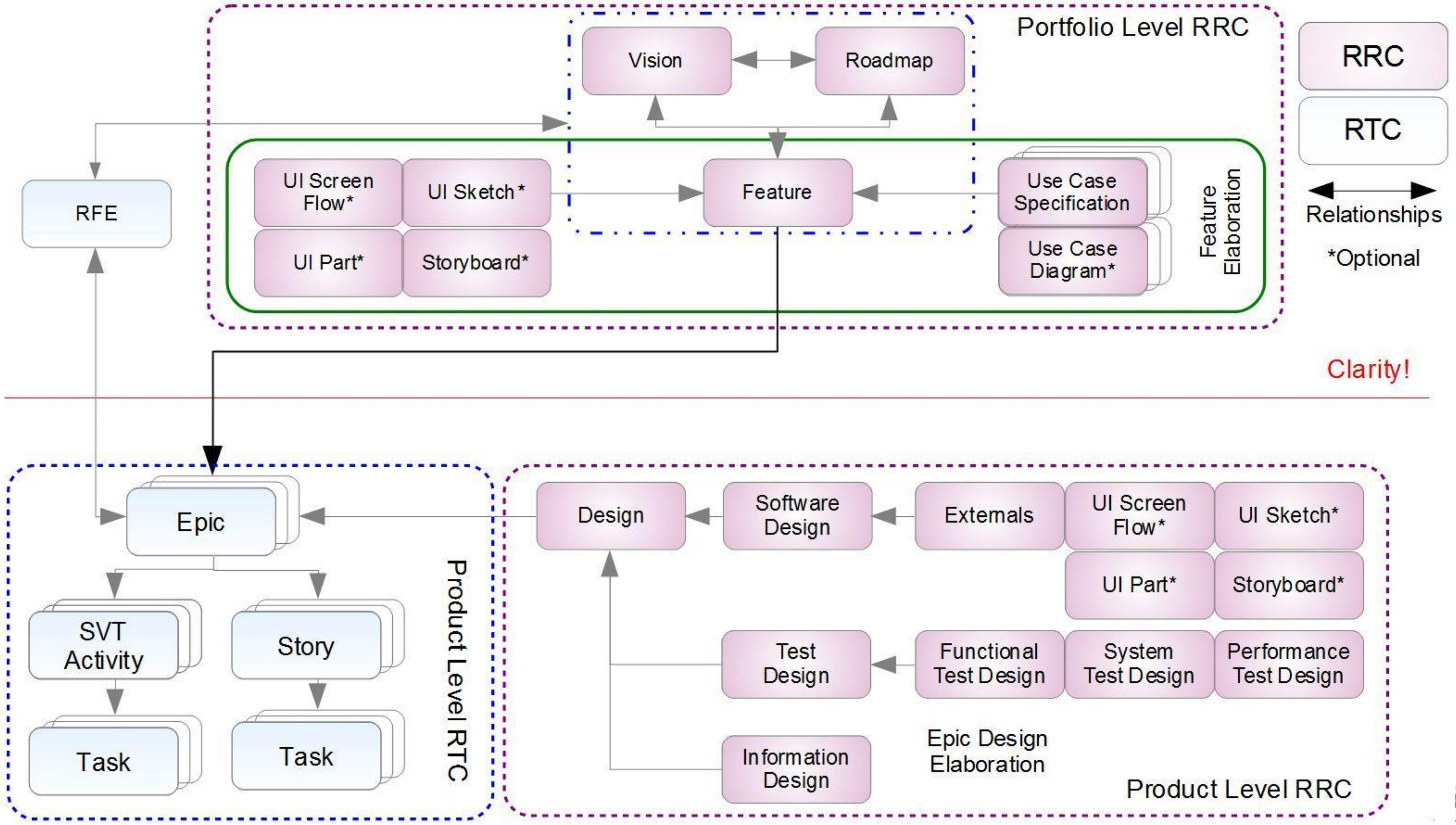
Portfolio planning using RRC

- Rational Requirements Composer appears to give us the ability to have an overview project for all the CICS products
  - A place where the visions and themes that were to be developed could be elaborated
  - An anchor where all the work for the portfolio would feed from and to
  - A separation of what is developed (RRC) from how it is developed (RTC)

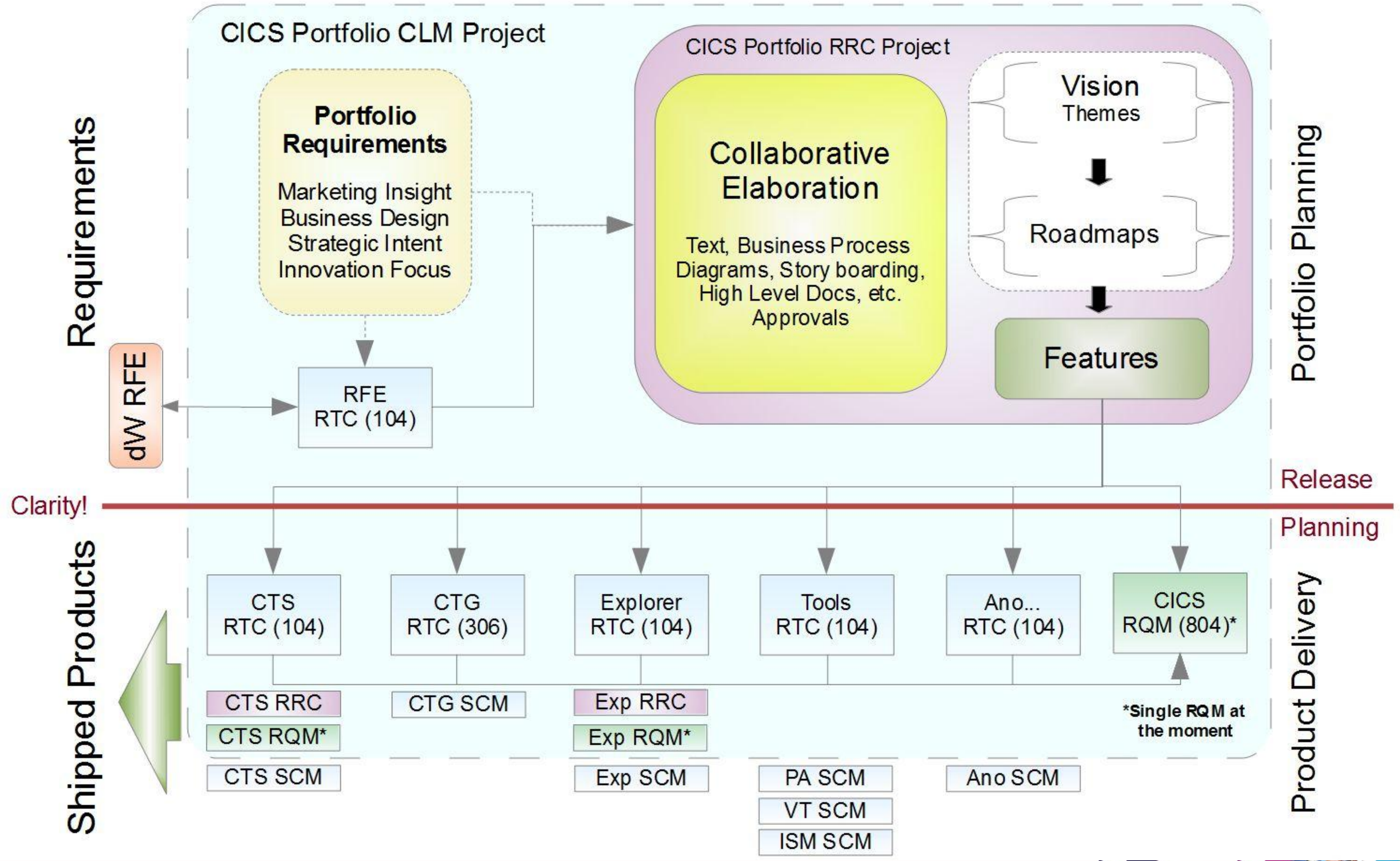
- Adoption of RRC will be staged in a similar manner as RTC was
  - March 2012 Strategy and planning writing Visions, Roadmaps and Features
  - Most of the adoption will be as we transition from the current release to the next release

- Migrating all of our designs from Wikis, Word documents, Lotus Notes databases, etc. to it

RRC / RTC Object Relationships



CICS Portfolio Jazz Architecture



RRC artifact types and RTC Work item types

- RRC
  - Vision
  - Roadmap
  - Feature
  - Text Document
  - Use Case Specification and Diagram
  - UI Screen Flow, Storyboard, Sketch and Part
  - Simple Flow Diagram
  - Meeting
  - Term (For glossary)

RTC

Epic
Story
SVT Activity
Dependency
Task
Defect
Risk
Action
Externals Change Request
Non-project Defect
Non-project Enhancement

Process issues highlighted

- Design clarity – need to revisit some of the key Agile principles
  - Agile does not mean ignoring design!
  - Not defining use cases prior to some of the development
  - Some development started prior to deciding what was needed
    - Portfolio planning exercise will help
- Leaner - 1 month iterations a real challenge
  - Many more smaller Epics and Stories
  - Many more reviews
  - Need to turn work around much more quickly
  - Fine balance required
    - Identified we need to streamline some of our process
    - Still need to maintain control – reasons of audit
  - Harder to control technical debt

What's next for CICS?

- RRC
  - New planning process using Features for the next release
  - Complete the migration to RRC for designs etc.
- Lessons learned for the current release – Jam session!
- Review our processes
  - Improved sizing and estimation – Story points and velocity
  - Can we be leaner?
  - Balancing control with freedom
- Review the project
  - Remove the clutter that has built up over the current release
  - Easy to add new things, need to remember to remove what is no longer used!
- Jazz V4
  - Particular investing time in RQM automation

Couple of tips...

Don't reinvent the wheel - Learn from what has gone before!

“Disciplined Agile Delivery” by Scott Ambler and Mark Lines:

"Although people are the primary determinant of success for IT projects, in most situations it isn’t effective to simply put together a good team of people and let them loose on the problem at hand. If you do this the teams run several risks, including investing significant time in developing their own processes and practices, in identifying the wrong processes and practices, in not identifying the right processes and practices, and in tailoring those processes and practices ineffectively. In other words, people are not the only determinant of success."

Keep it simple

The Agile Manifesto - Principle 10: Simplicity--the art of maximizing the amount of work not done--is essential

Do not apply this just to the product and what is being developed, but also to your process and tooling. If you are migrating from another process/tooling, it is far too easy to try and make it work the way it used to. **Don't!** Re-examine everything and make sure there is a real **need** for it. If it seems a good idea or because someone wants it is not a good enough reason for including it. Keep applying this principle (Lean?) as you progress and regularly re-examine how you are doing things to see if can be streamlined

Summary – An ongoing journey

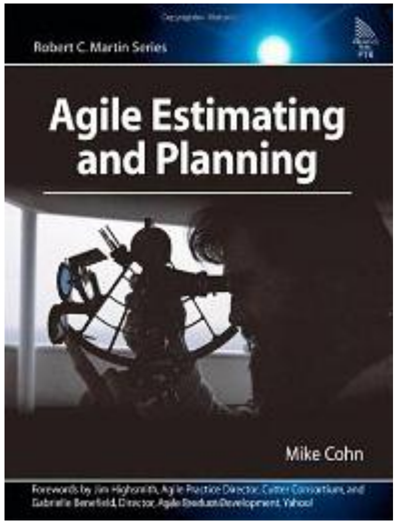
Waterfall to Agile is possible for Enterprise Systems	• Does not have to be new, small, co-located teams • Expect teething problems – generally not difficult to fix • Be careful of falling back into old habits
Adopting Agile tooling provides a catalyst for change	• Necessity for some aspects of Agile adoption • Re-examines processes in a new way • We could not be where we are without the right tooling!
Be Agile in adopting Agile and the tooling changes	• Possible to stage adoption of both Agile and tooling • Does not have to be for all aspects e.g. SCM • Plan ahead - For architectural design and configuration

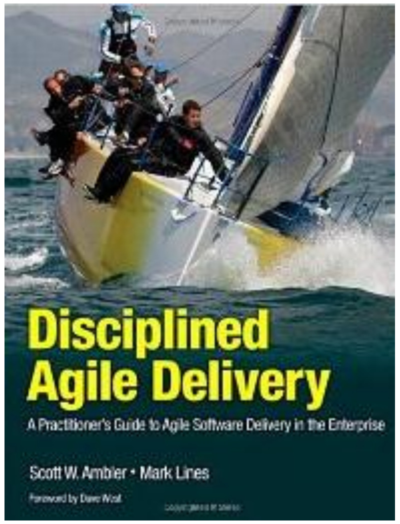
QUESTIONS

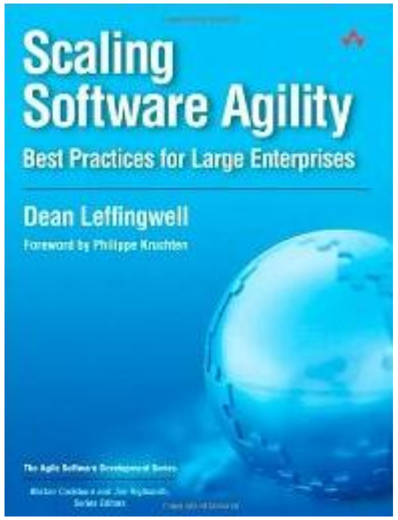
Backup Slides

Books to consider

- Agile Estimating and Planning by Mike Cohn
- Disciplined Agile Delivery by Scott Ambler and Mark Lines
- Scaling Software Agility by Dean Leffingwell

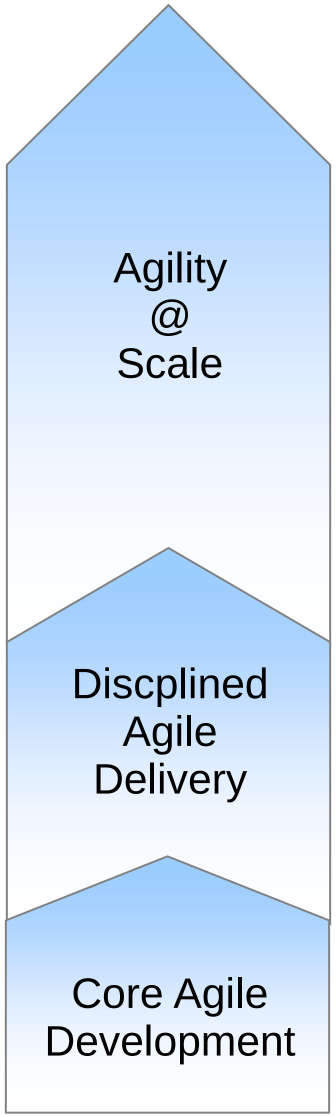



Disciplined Agile Delivery
A Practitioner's Guide to Agile Software Delivery in the Enterprise
Scott W. Ambler • Mark Lines


Scaling Software Agility
Best Practices for Large Enterprises
Dean Leffingwell
Foreword by Philippe Kruchten

The Agile Scaling Model (ASM)

- Disciplined Agile Delivery (DAD) when one or more scaling factors apply
  - Large team size
  - Geographic distribution
  - Regulatory compliance
  - Domain complexity
  - Organization distribution
  - Technical complexity
  - Organizational complexity
  - Enterprise discipline
- Basic Disciplined Agile Delivery for the Enterprise
  - Risk + Value Driven Lifecycle
  - Self organisation within appropriate governance framework
  - Full delivery lifecycle
- Agile Development. For example, Scrum, Lean, XP and Agile Modelling
  - Value driven lifecycle
  - Self organising teams
  - Focus on construction



The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over *processes and tools*

Working software over *comprehensive documentation*

Customer collaboration over *contract negotiation*

Responding to change over *following a plan*

The 12 Principles behind the Agile Manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
4. Business people and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation
7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity--the art of maximizing the amount of work not done--is essential
11. The best architectures, requirements, and designs emerge from self-organizing teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

THANK
YOU