Agile Software Development
-
Experiences from the Trenches

Johannes Rieken

IBM Rational Zurich Research Lab

Manifesto for Agile Software Development

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

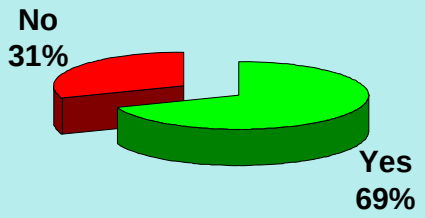
(see http://agilemanifesto.org/)

What is Agile

An iterative and incremental (evolutionary) approach performed in a highly collaborative manner with just the right amount of ceremony ***to produce high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders.***

So Agile is not a method. Its more like a conceptual framework (abstract class) with concrete implementations like, Eclipse Way, OpenUp, Scrum, … or the My Company Way ☺.

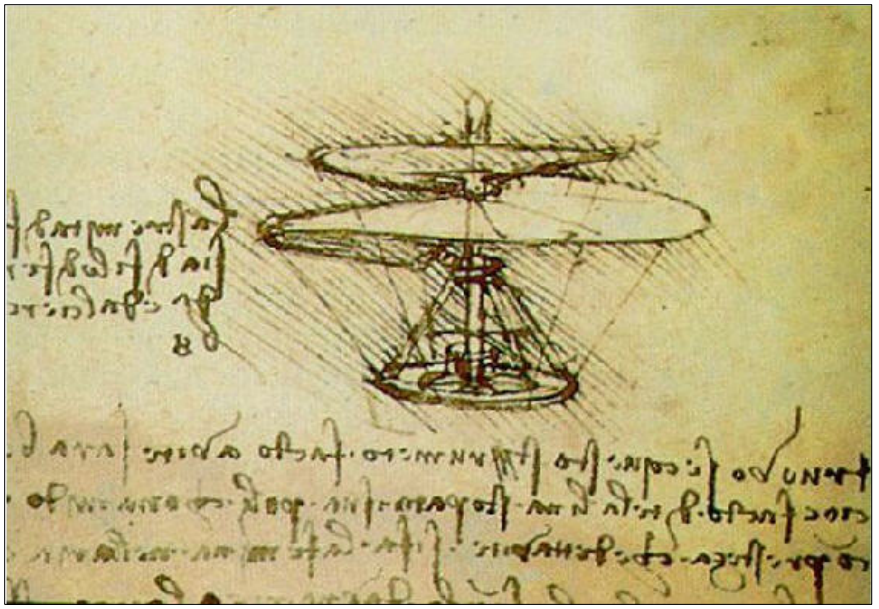
Have you adopted any Agile techniques?



Source: Ambler 'Agile Adoption Rate" & "Project Success" surveys - www.ambysoft.com/surveys/

Agile is Not

- **Low ceremony**
  - It is (very) formal and has (very) specific practices
- **Do what you want (cowboy coding)**
  - Requires lots of discipline
- **Easy to do**
  - See above ☺
  - Needs cultural changes
- **A silver bullet**

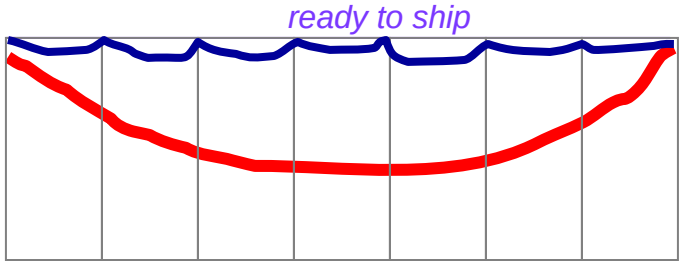


Core Principles

- **Short development iterations (3 – 6 weeks)**
- **Ongoing customer / stakeholder involvement**
- **Ongoing investment in code quality (refactoring)**
- **Retrospectives (improve the process)**
- **Self organizing teams**

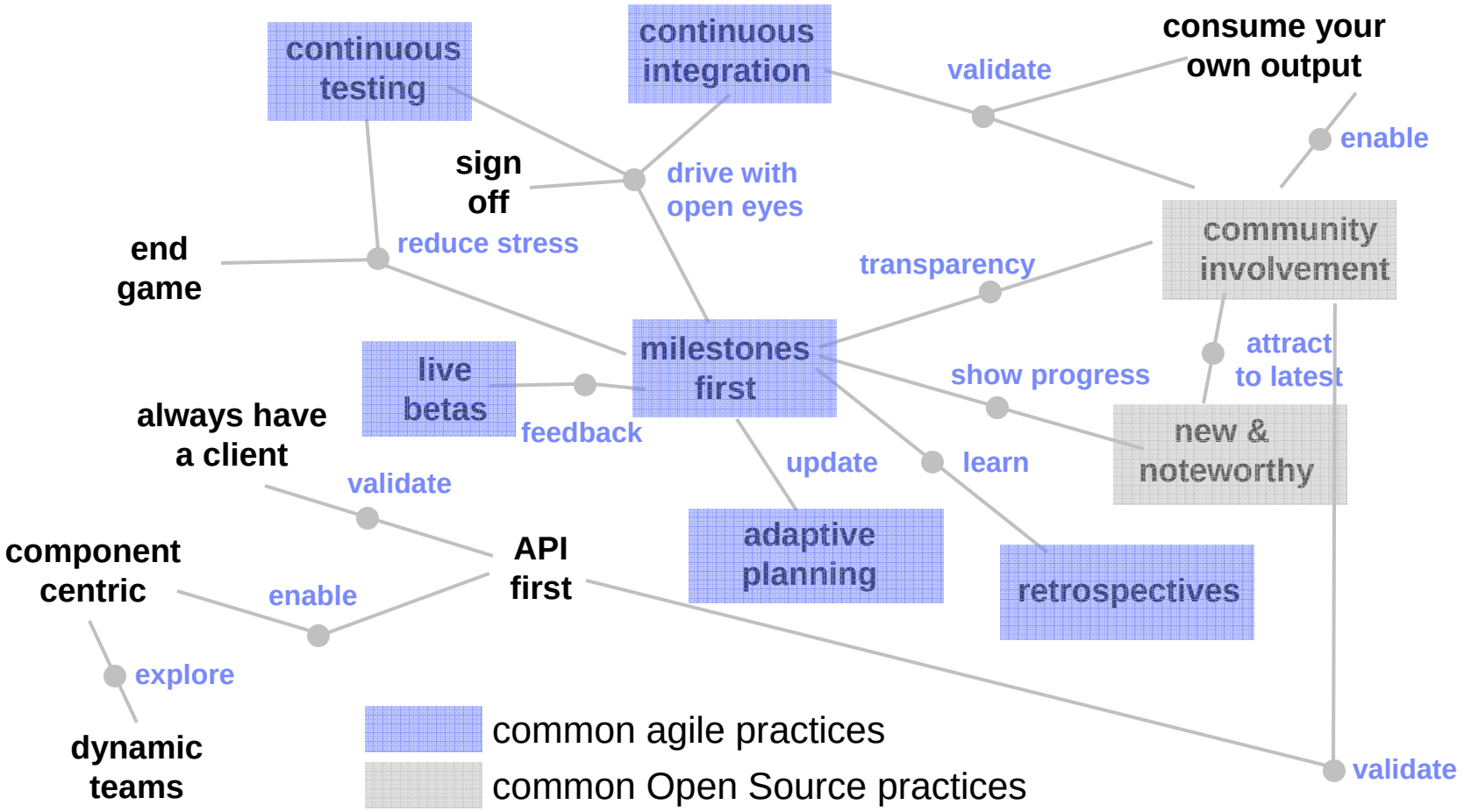
Short iterations – the main driver

- **Allows feedback / quality checks any n weeks**
- **Each milestone is a miniature development cycle**
  - plan, execute, test, ship, retrospective
- **The iteration outcome (library, product, …) must be shippable / consumed**
  - Other teams, betas, demos, …



➢ **Short iterations reduce stress !!**

The Eclipse Way Practices

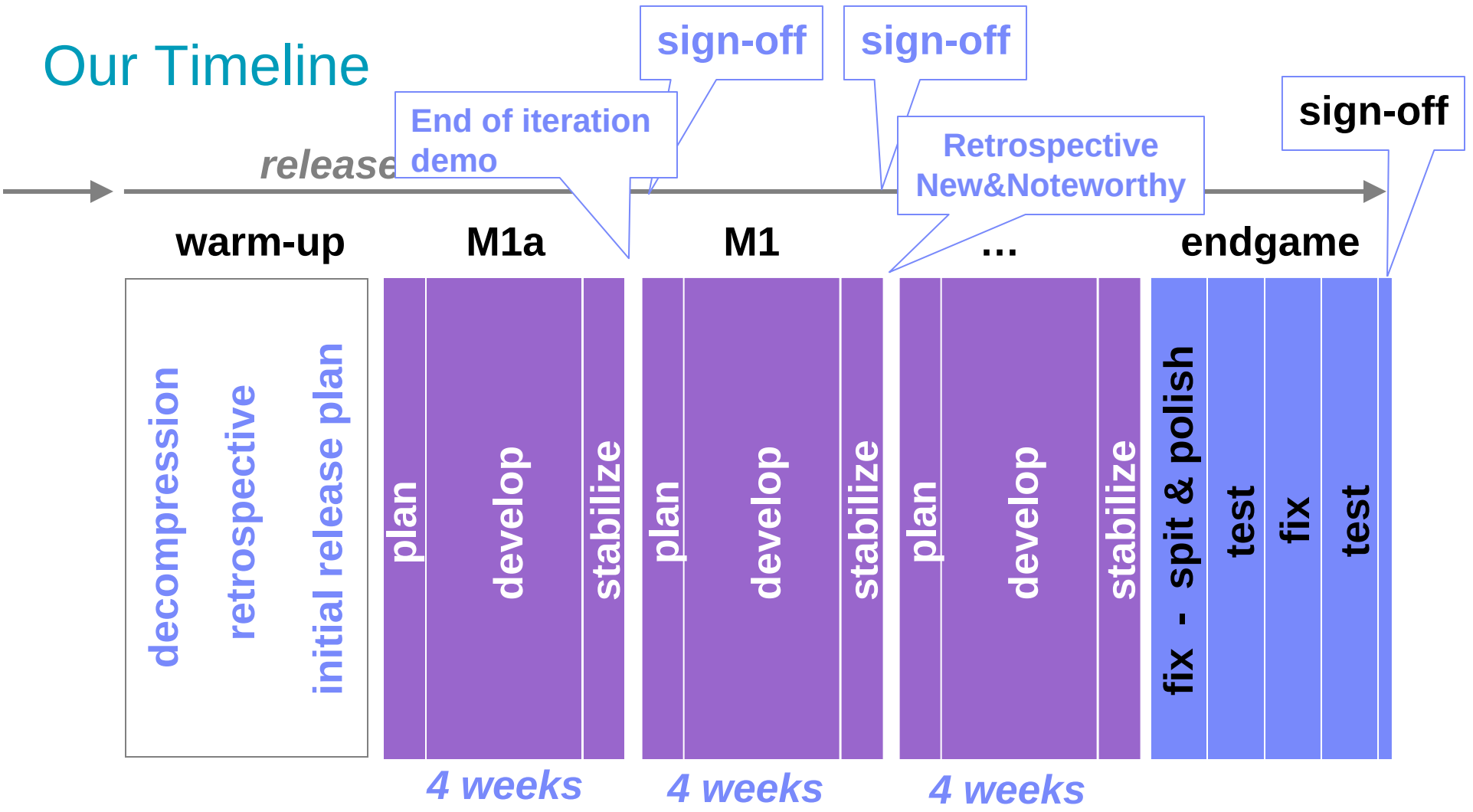


What is behind the Eclipse Way

- **Practices underpinned with** values
  - ship quality on time
- Used**,** developed **and** improved **over time**
- **Practices are from all kinds of sources**
  - XP, Scrum, Crystal Clear, RUP, …
  - Patterns - Organizational Patterns of Agile Software Development – Coplien
- **It is** not low ceremony
  - Approvals, verifications, reviews
- **It is** agile**: incremental, iterative, collaborative, transparent, customizable**

➢ Many effective teams work like this

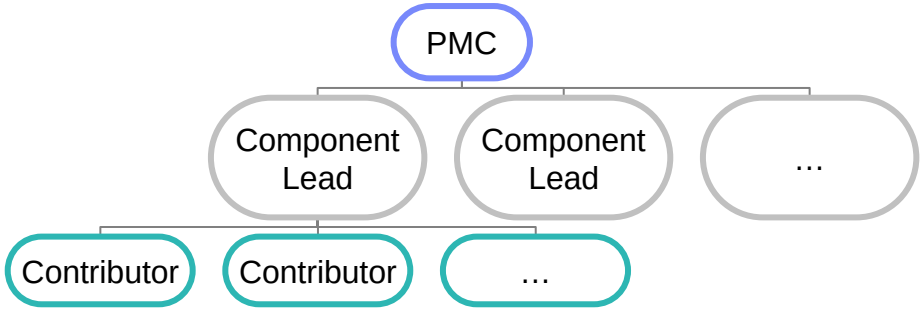
Our Timeline

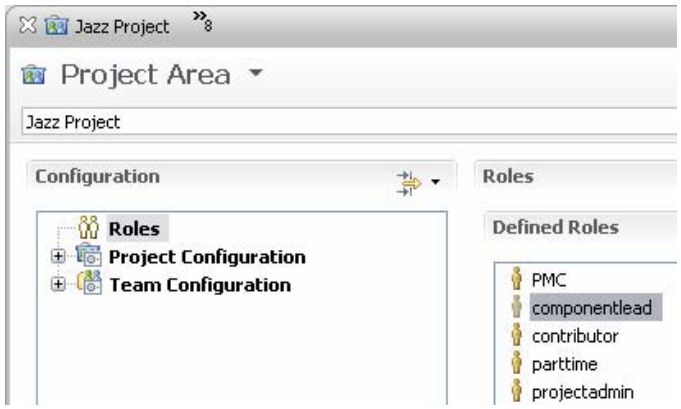


- 4 week iterations ⇒ end with an end of iteration demo
- 8 week milestones ⇒ announced with New & Noteworthy ⇒ retrospective at the end

Our Roles

- Project management committee (PMC)
  - –Accountable for release plan
  - –Themes to work on
  - –Facilitator, coordinator
    - encourages participatory decisions
      - e.g. top 5 architectural issues
- Component lead / Team
  - – Accountable for iteration plan, test plans
- Contributor
  - – Accountable for estimates, code, tests, design
  - – Plays many roles
    - Developer, Tester, Architect
    - Customer support, Release Engineering

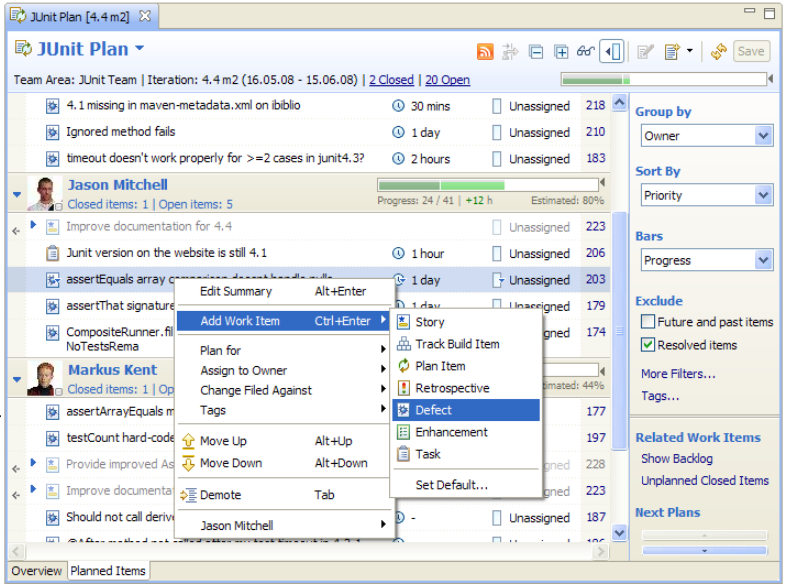




Iteration Plan Input

Community Stakeholders

PMC

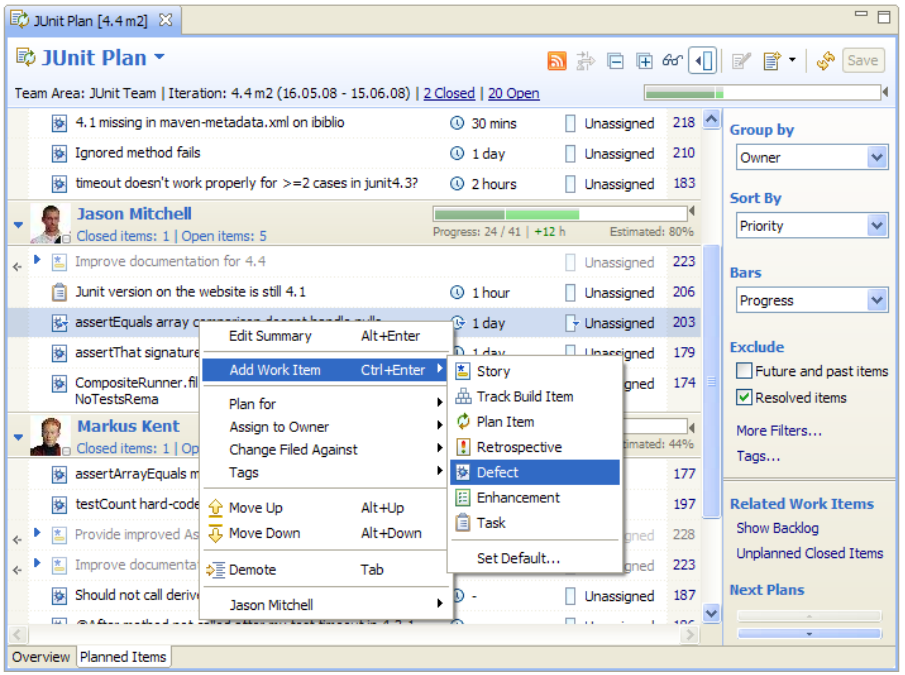


Defect Backlog

Team

Iteration Plan Input - continued

- **PMC**
  - Requirements (release plan): *traceability,* …
  - Themes: *improve performance*, …
  - Concrete tasks: *externalize strings*
- **Community / Stakeholders**
  - Requirements
  - Enhancements
  - Defect reports
- **Team**
  - Architectural issues
  - Redesigns
  - Product ideas
- **Defect Backlog**
  - Previous test passes
  - Self hosting



Iteration Plan

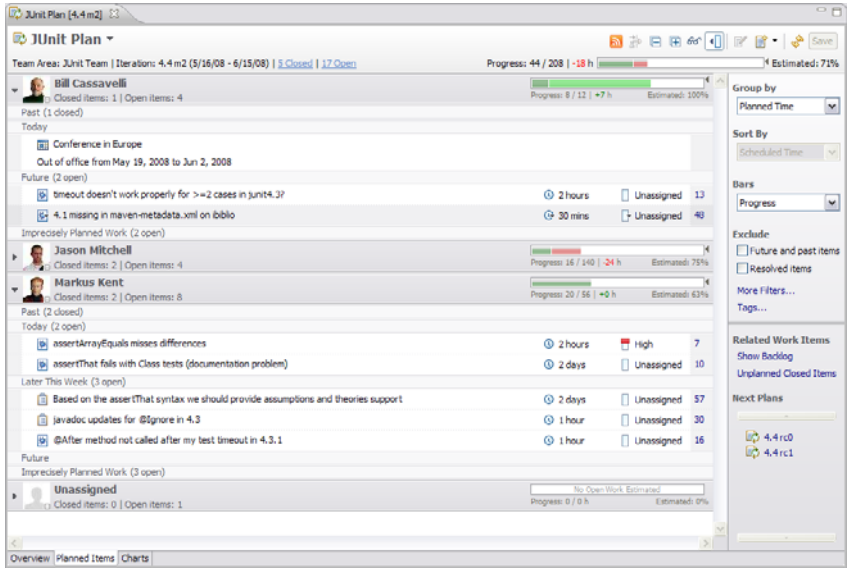
- **Goal: an estimated 4 weeks plan approved by the team, PMC, and stakeholders**
- **In the RTC project the steps to get there differ from team to team**
  - Team lead creates initial plan based on input from PMC, stakeholders
  - Team decides on defect backlog, architectural issues, …
  - Team members estimate their work
  - PMC buddy approves the plan
- **In Scrum the sprint (iteration) plan is defined in a sprint meeting:**
  - Stakeholders (product owner) are part of that meeting. They manage a prioritized list of product requirements, enhancements
  - Team manages defect backlog, architectural issues, …
  - Team estimates

Iteration Checkpoints

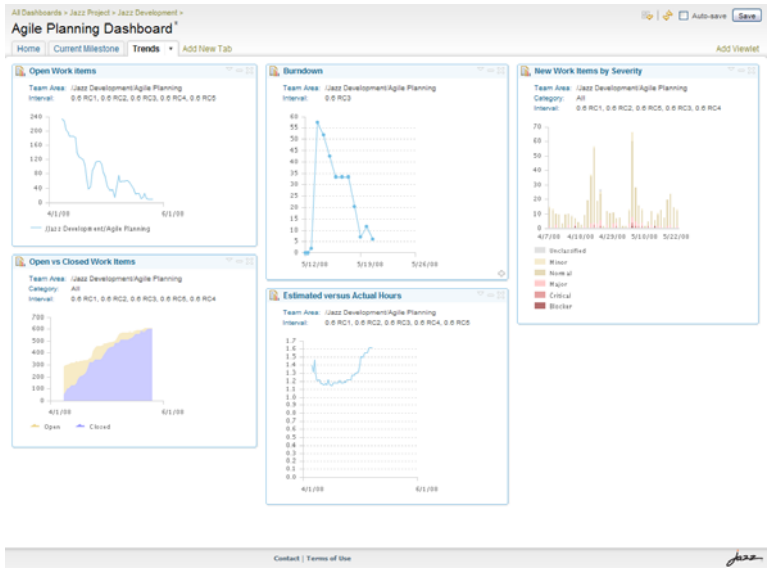
- **Daily (stand-up meeting, team lead – contributor discussion, …)**
  - What have you done yesterday
  - What are you going to do today
  - Any road blocks
  - How many work is remaining
- **2 times a week (Planning call)**
  - Inter team issues
  - Progress on PMC and Community work
- **Every two weeks**
  - Iteration Plan walkthrough with PMC buddy
- **End of iteration**
  - New & Noteworthy / Milestone
  - Demo for PMC, stakeholders and other teams

Tracking Progress

To hit the end target of an iteration it is essential for the team to track its progress. Teams in RTC do so using Agile Planning tools and Dashboards.



Single Iteration

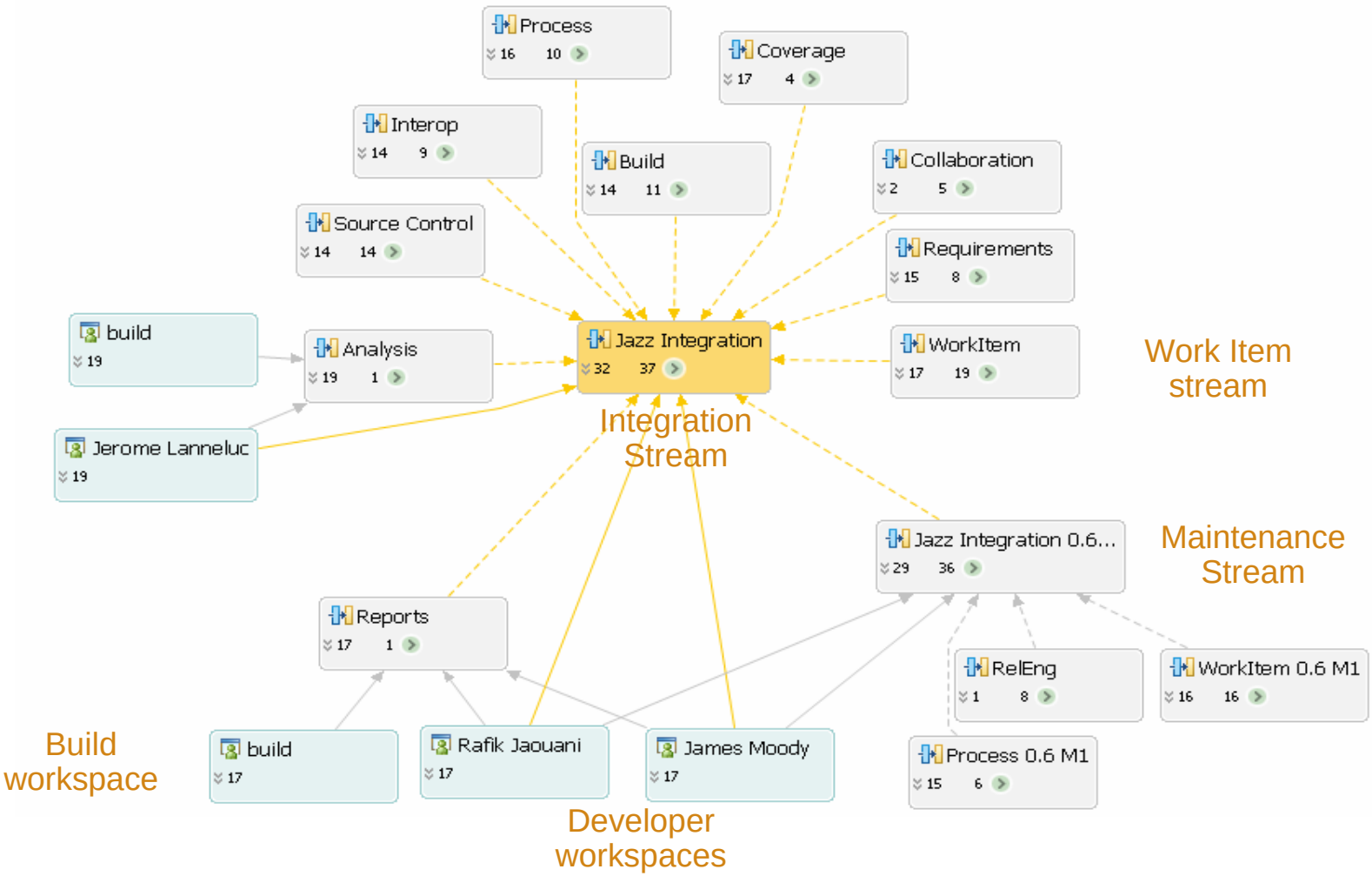


Trend

Development - Isolate Work not people

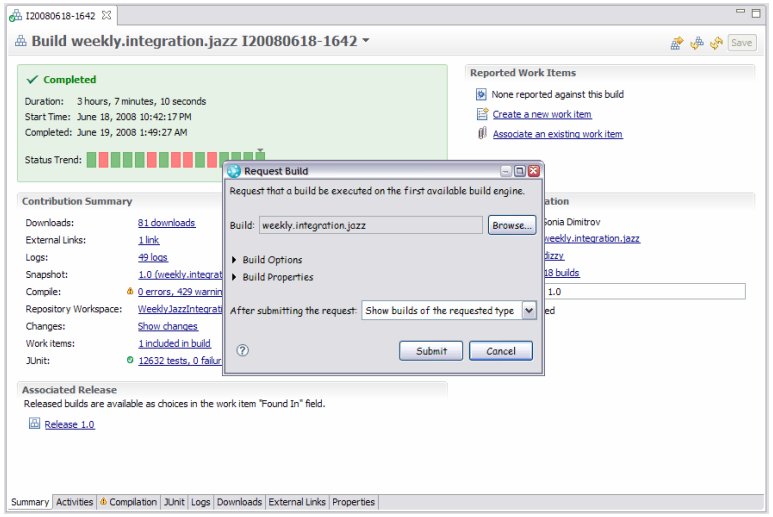
- Repository workspaces **– Provides individual isolation. You don't have to make your changes visible to the team just to backup or use the repository features.**
- Streams **– Provides team / project isolation.**
- Suspend and Resume **– Provides task level isolation for personal work.**
- Team areas **– Provides process isolation.**

Stream setup for RTC Development

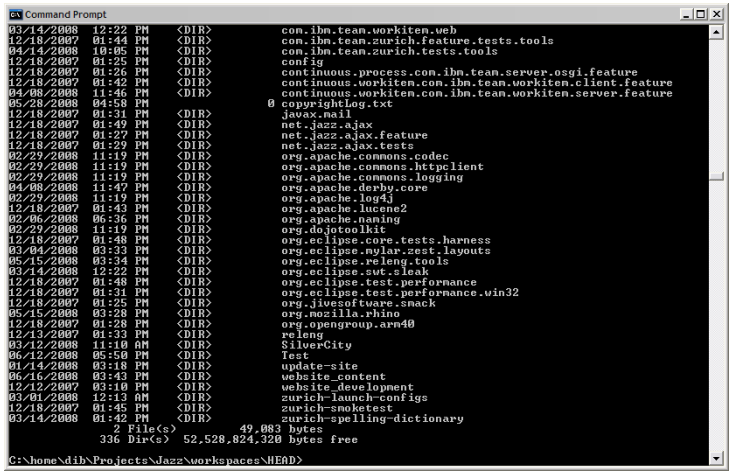


Build Process

To be able to ship a product every n weeks building it must be absolutely painless and fully automated. And the build quality must be verified using automatic unit tests.

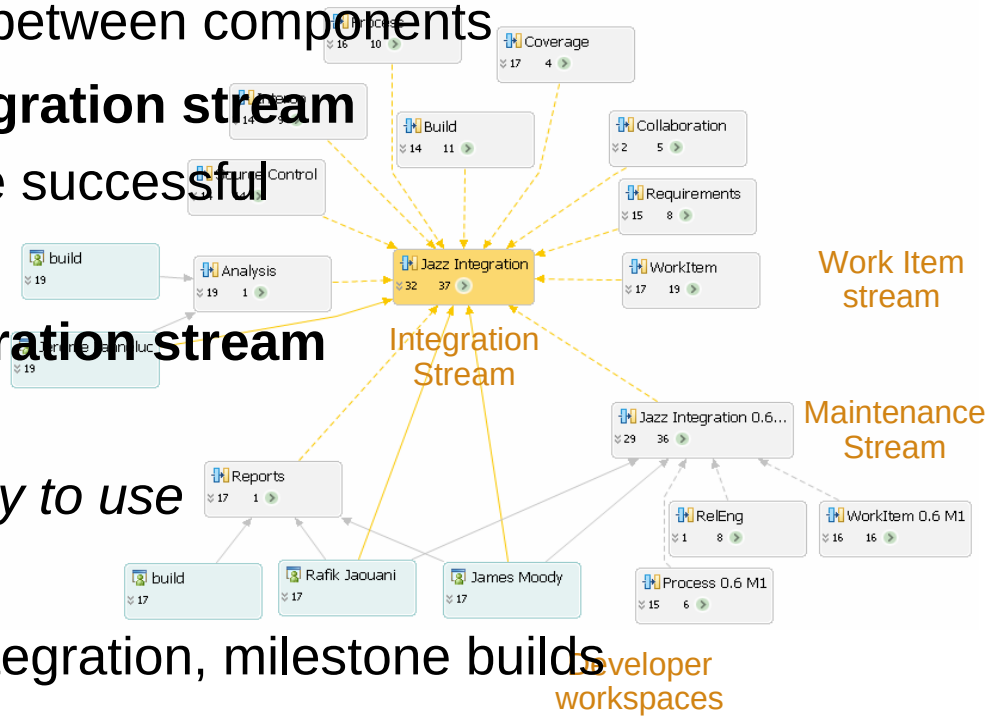


versus



Staged Builds

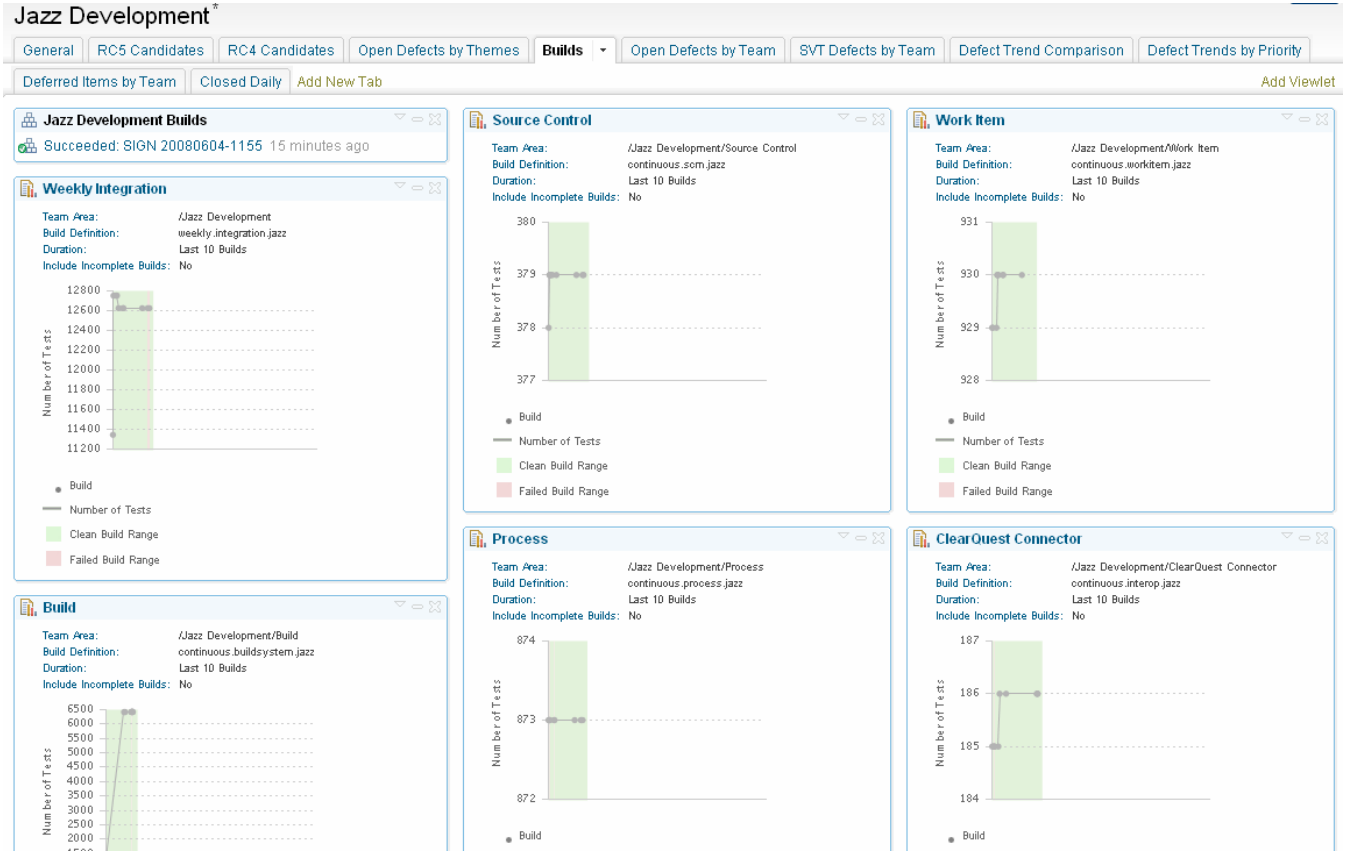
- **Team builds – team's integration stream / n times a day**
  - discover component problems
- **Nightly builds – project continuous integration stream**
  - discover integration problems between components
- **Integration builds – weekly integration stream**
  - all automatic unit tests must be successful
  - *good enough for our own use*
- **Milestone builds – weekly integration stream**
  - test / fix pass
  - *good enough for the community to use*
- **Reality: build failures occur**
  - rebuild to create acceptable integration, milestone builds



Rational. software

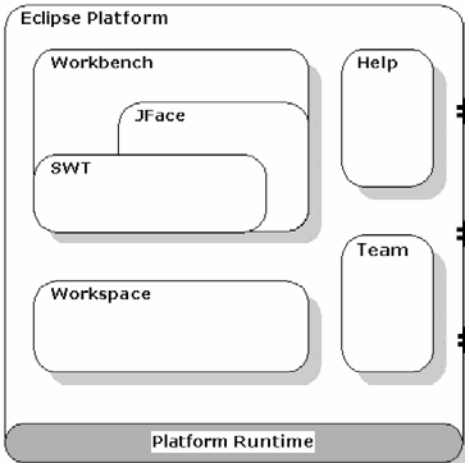
Build Tracking

Equally important is to track whether a build has compile errors and/or failing test case. The majority of the builds should be green.

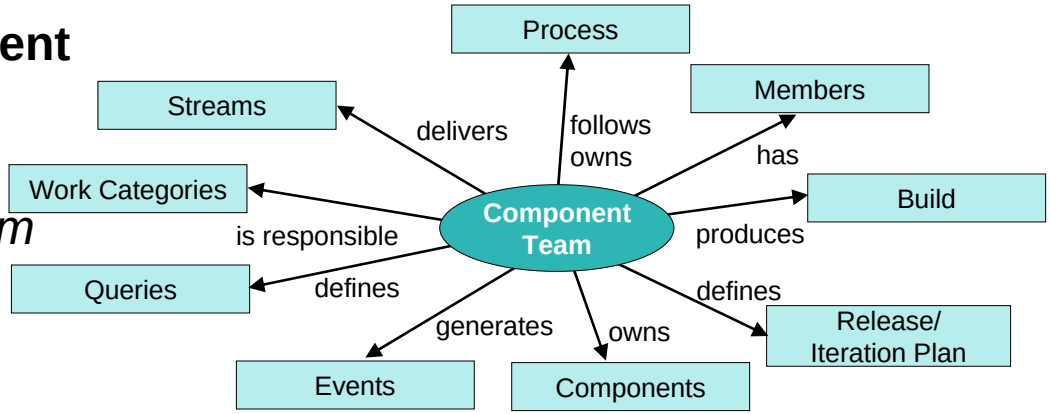


Agile @ Scale: Component Based Development

- **Component based**
  - A **team** is responsible for one or more **component** at one **site – co-location**
    - **"architecture follows organization"**
  - **Components are distributed across sites**
- **API first**
  - An API is a commitment
  - Producer / consumer relationships
- **Team Concert supports component based development**
  - Team owns a stream
  - Team shares changes in a *stream*
  - Team owns a *component*
  - *Stream* references components

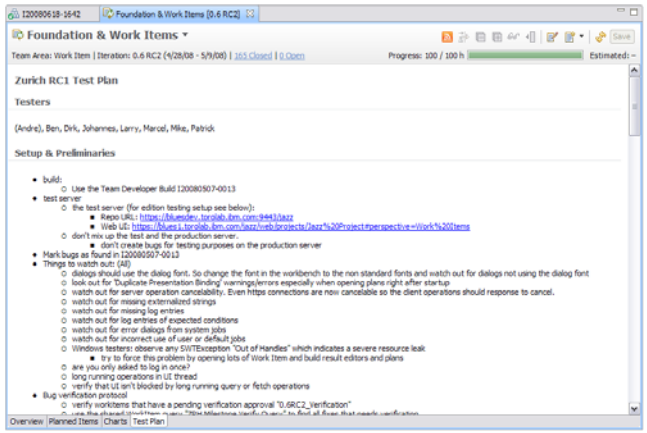


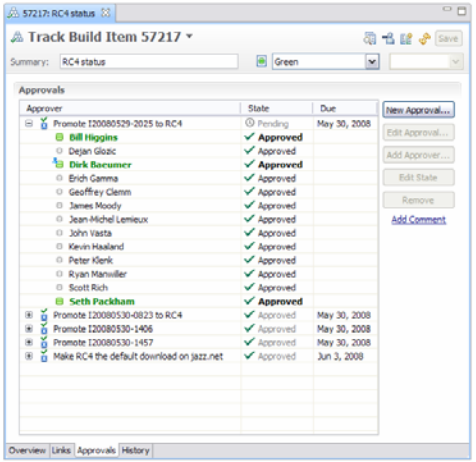
Eclipse Components



Stabilization

Each iteration ends with a stabilization phase. In the RTC project this is the last week of a 4 weeks iteration



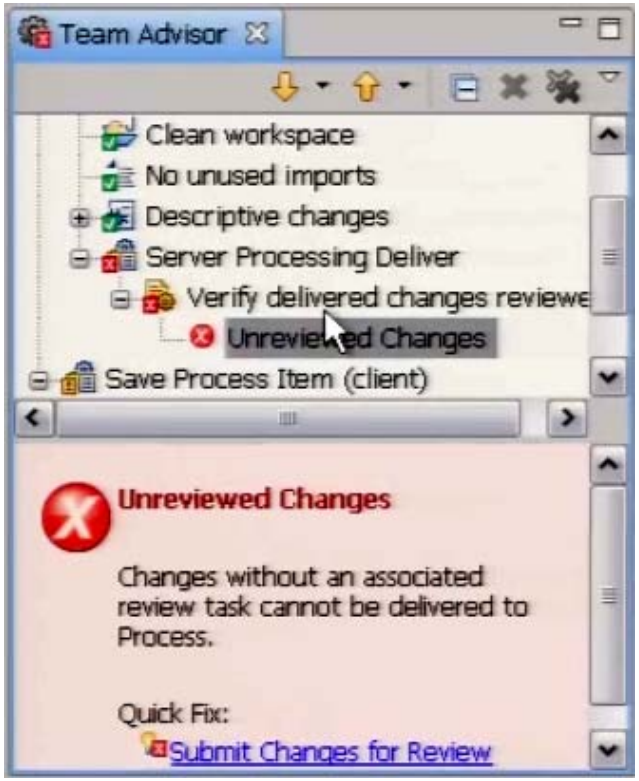


- **First two days is testing**
  - Teams create test plan
  - Teams do actual testing
- **Next two days is fixing**
  - Only critical defects are fixed
  - Every fix requires a team lead approval & a code review
- **Last day is sanity check day**
  - Rebuild only for stop ships
  - All teams sign off on Milestone build

Process Awareness

"While many aspects of process might be automatable, we found that productive processes emphasize the creative value added by the people in the process."

Organizational Patterns of Agile Software Development – Coplien

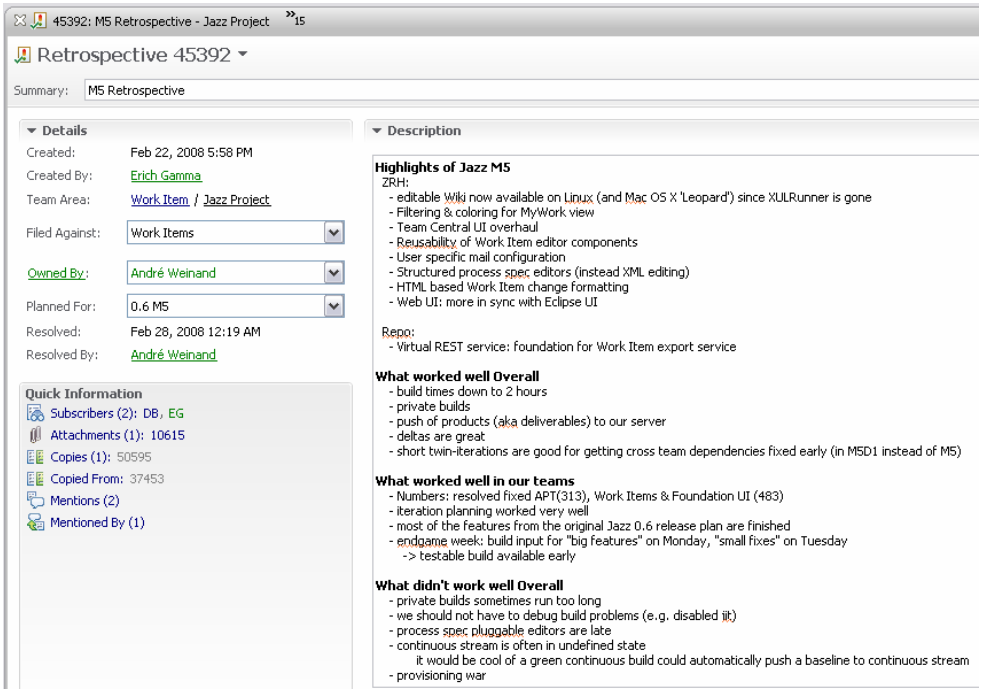


- **Support many** different **practices and processes**
  - Rational Team Concert is Process neutral
- **Reactive, not controlling**
- **Specific to the team, development phase**
  - **Reduce team member mistakes**
  - **Free the team members minds**

Retrospective

After each iteration teams reflect on what worked well and what didn't.

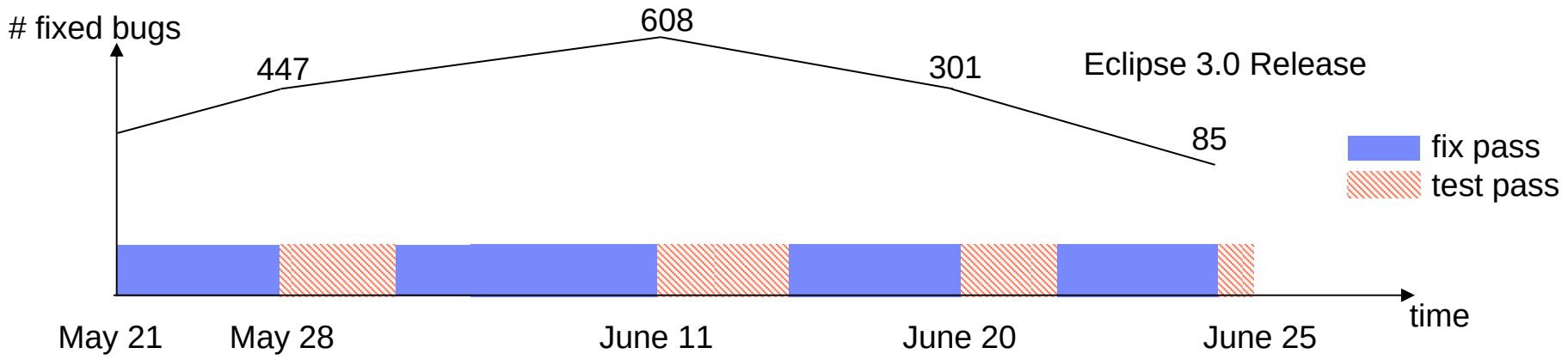
- Retrospectives are captured in a special work item
- Define actions how to tune the process to get more effective
- PMC does a retrospective as well



Endgame

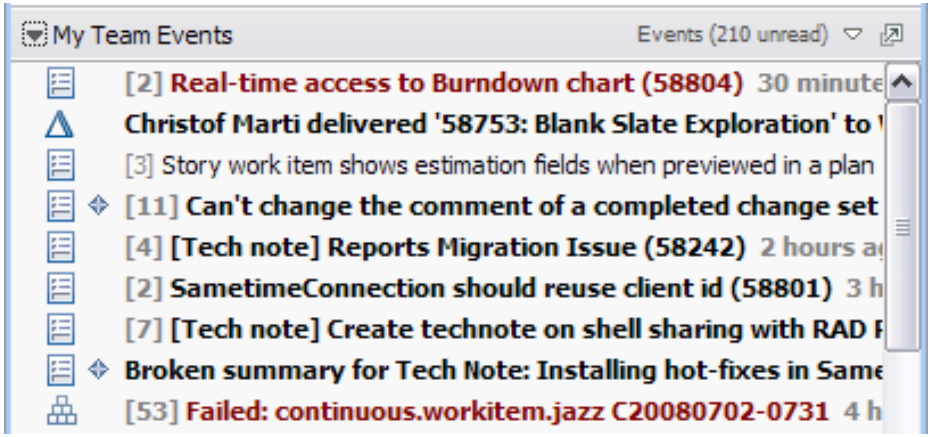
Convergence process applied before release

- **Sequence of test-fix passes**
  - community event
- **With each pass the costs for fixing are increased**
  - higher burden to work on fix for a problem
  - higher burden to release a fix for a problem
  - focus on higher priority problems



What Else is Important

- **Transparency**
  - Who is doing what
  - How good is the progress
  - …
- **Traceability**
  - Which change sets are in a build
  - Who fixed the bug
  - …
- **Team First**
  - Joining a team
  - Team controls the way it works
  - …



Recommendations

- **Make sure that you can build the product by pressing a button**
  - Run integration build if necessary
- **Ensure quality with automated unit tests**
- **Being iterative is the key. It will induce almost all the rest.**
  - Start with 4 – 6 weeks iteration and stick to the rhythm. Adjust (after a while) if necessary
  - Ensure that the iteration produces a deliverable that is consumed
  - Plan, execute, test
  - Review what you have done and improve (the process)