

Highlights of Java in the Trenches Reflections along the Eclipse Way

Original
Erich Gamma & John Wiegand, IBM
JavaOne 2006 – May 18, 2006

Highlights
Eduardo Pelegri-Llopart, Sun
Santa Clara, CA - Oct 17, 2006

Original Presentation

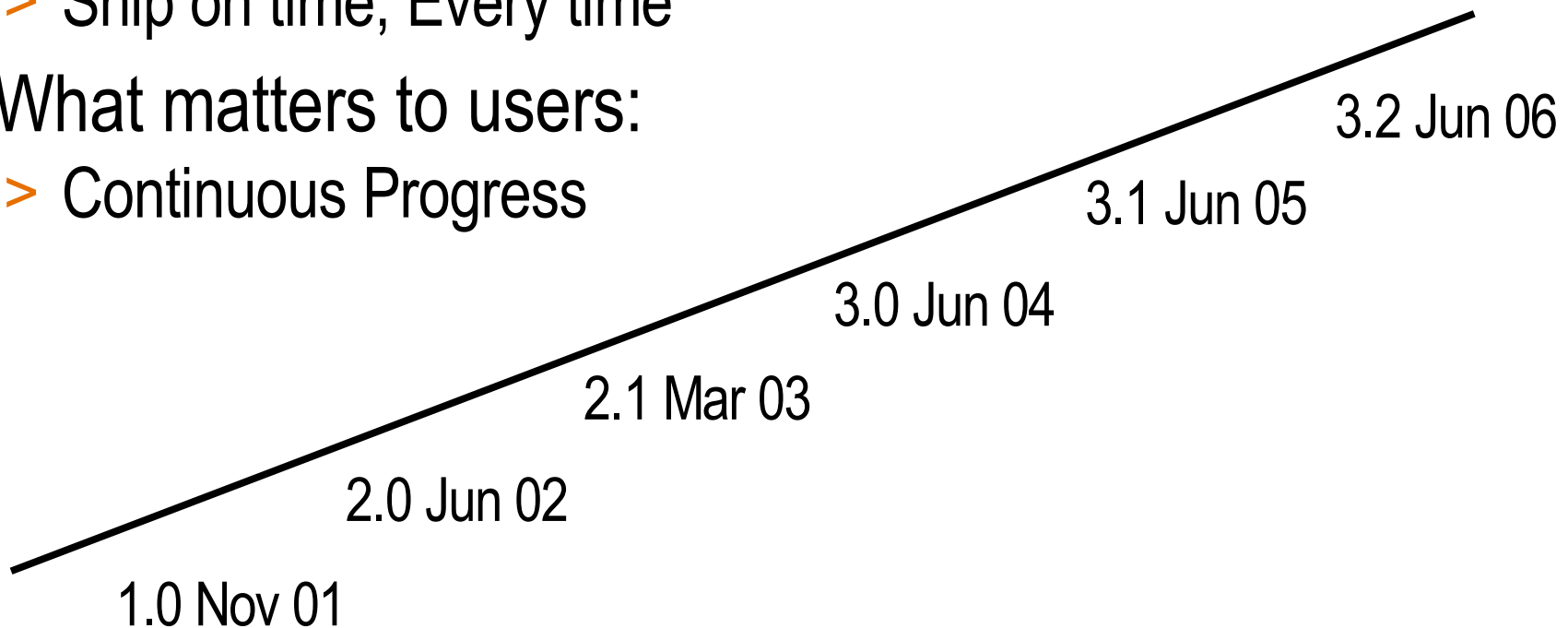
- Keynote at JavaOne 2006 – Day 3
 - > Day 3 – May 18th, 2006
 - > Erich Gamma & John Wiegand
- Available Online
 - > <http://java.sun.com/javaone/sf/sessions/general/day3.jsp>
- Slightly higher quality VHS tape available
 - > Could not get original slides, thus...

Agenda

- Intro / Background
- Rhythm
- End Game
- Endurance
- Jazz demo (not covered here)

Background – 5 Years Shipping on Time

- Globally Distributed Development since Eclipse 1
- What matters to us:
 - > Ship on time, Every time
- What matters to users:
 - > Continuous Progress



What Matters Most?

- Tools are important
- Methodology is important
- But...
 - > People, Team, Community is MOST IMPORTANT
 - > People and Interactions win over Processes and Tools

Values

- Shipping
- Predictability
 - > Ship on time. Other groups & community can rely on it
- Transparency
 - > No secrets about ship readiness. No surprises
- Feedback
 - > Are we ready to ship? More features?

Rhythm

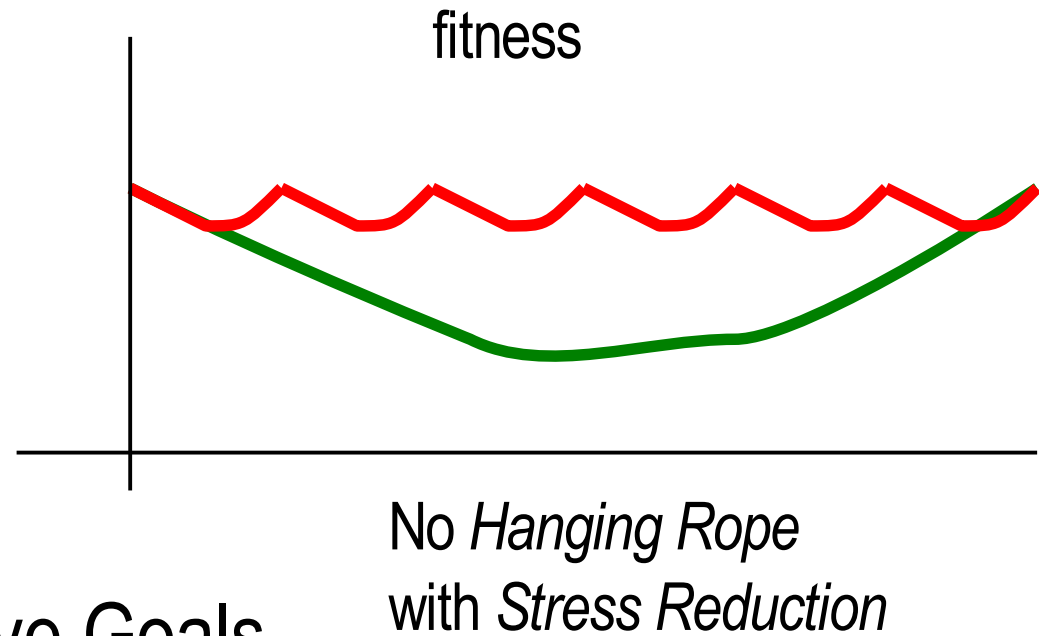
- Milestones
 - > *Warm-Up*: Recover, Restrospective, Initial Release Plan
 - > *Steady State*: Milestones (M1.....)
 - > Mini-releases: Plan, Develop, Stabilize. Best length: 6 weeks
 - > *End Game*: Stabilize. Sign-off
 - > Project is fit to use throughout the cycle

Continuous Transparent Planning

- Old: Static plans, but
 - > Accurate only at one point in time
 - > Non-existent until late in the cycle (secret plans)
 - > Transparent development but *not* transparent planning?
 - > No early feedback
- New: Items in 3 categories based on expectations
 - > *Committed* items
 - > *Proposed* items – initial state
 - > *Delayed* items
- Plan is updated *continuously*

Healthy Milestones

- Consumable
 - > Trusted
 - > Demoable
- Honest
 - > Time-based
 - > Bottom-up
- Allows for Aggressive Goals



Healthy Milestones - Stabilization

- Rhythm leads to lowered stress
 - > People know what to do
 - > How to recover from changes
- Example: Stabilization Week:
 - > Mon – Warm-up
 - > Tue – Validate
 - > Wed – Test
 - > Thu – Fix
 - > Fri – Verify / Go-NoGo / Release / Assess / Retarget

Continuous Integration

- Fully Automated Build Process
 - > Publish Results
- Unit Tests
 - > 30,000 Junit tests
- Staged Builds
 - > Daily Builds
 - > Green Integration Builds
 - > Milestone Builds

Live Betas

- Continuously Consumable
- Continuously Interesting
- Continuously Listening
 - > Users have influence
 - > Encourages feedback
- Feedback Loops
 - > Daily – each team
 - > Weekly – whole team
 - > Milestones – whole community

Live Betas - Marketing

- *New and Noteworthy Program*
- Increase Incentive to move to new milestone builds
 - > Need feedback on our work
 - > Reduce stale defect reports
- Market Milestones
 - > Publish *New and Noteworthy*
 - > Advertises what we have been doing
 - > Then community talks and blogs about it.

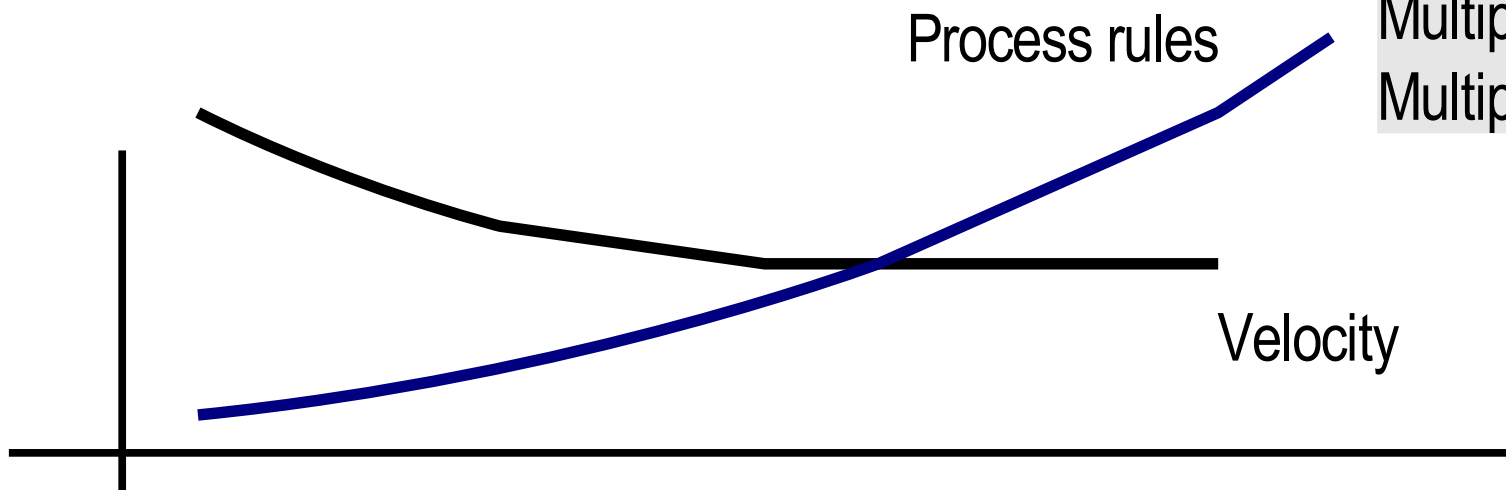
Endgame Rhythm

- Fix – spit & polish
 - > And then ... Test – fix – test – fix ... test – Sign Off
- Endgame Endurance
 - > Milestones reduce stress at the end of release
 - > Distribute Quality and Polish through the release
 - > No separate testing organization

Endgame Transparency and Convergence

- Convergence Process Applied *before* Release
- Sequence of test-fix passes
- Costs for fixing increase through sequence

Raising the bar
 Publish fixed bugs
 Multiple approvers
 Multiple reviewers



Endgame – Sign-Off

- Shared Responsibility and Commitment
 - > All sign up

Endurance – Through many Releases

- Getting Healthy (earlier slides)
- Steps to Component Happiness
 - > Define Architecture / APIs - Use them as a team - Iterate
- Staying Healthy
 - > Scripts, Notes
 - > Avoid Code Bloat, Software Rot, Bug Pileup, Broken Glass
 - > Rewrite! Scheduled as part of the releases

Continuous Performance

- Add Performance Tests
- Measure... Visualize... Track

Summary – Continuous *

- Continuous Health through continuous...
 - > Transparent Planning
 - > Design / Refactoring
 - > Integration
 - > Testing
 - > Listening
 - > Demos
 - > Consumption of own output
 - > Feedback
 - > Learning