eclipse CON 2004

JIT Software Development

Inside the Eclipse Software Development Process

Kevin Haaland – IBM Ottawa Lab

 Feb 4,2004

Background

“I've been following the development of Eclipse for some time and I'm continually amazed as the extended team hits each projected milestone and ship date with precision. Until this experience, I was convinced that such a feat was impossible for a software project of any size and complexity.

I've read about the PMC and the published plans and all that, but I've been involved in projects that had plans and leaders that were still significantly later than expected. So... the question is how is this really accomplished? And the immediate follow-up of course is, how can this process be replicated elsewhere?

I'm also interested in where this method of development came from, was it part of OTI's culture from the beginning or did it come from IBM? Did it spring from somebody's head fully formed or evolve over time based on project management research? Does it have a name?”

- a post to the eclipse developer’s mailing list

The history of the eclipse project

Key attributes of the eclipse 3.0 planning process

Started with a question

"What would it take to have the next version be the most cool and compelling release ever that would not only be good for the eclipse community but also fun to work on?"

Resulted in many ideas

- Each idea was written down on a "Post-It" note

Important themes were identified

- Plan ideas were organized and prioritized based on themes
- Every plan item is either: Committed, Proposed, Deferred

The eclipse 3.0 plan is public and is regularly updated

http://www.eclipse.org/eclipse/development/eclipse_project_plan_3_0.html

The “plan” is a living document

The 3.0 plan is a living document and is updated to reflect
- What we learn from each milestone
- New items
- Input from the community

The plan accurately reflects the teams commitments

Milestones

- Major releases
  - 6-12 months between major releases
  - Schedule depends on plan objectives
- Service releases
  - 3 to 4 times a year
- Milestones occur every 6 weeks
  - 5 weeks development
  - 1 week shutdown
- Milestones are stable builds
  - New and noteworthy
  - Show progress on overall 3.0 priorities
  - Opportunity to re-plan based on new information

What we build

- Daily builds from the latest code checked in by all teams
  - Builds from HEAD
  - Primarily used by eclipse platform developers only
  - Full regression tests run on every build
- Regular Integration Builds
  - Teams need to explicitly release versions of their code to be included
  - Intent is that these builds are “good enough” for the eclipse platform developers to use for self hosting
  - A staging process to create milestones
  - Published build schedules (Tuesday 08:00 Eastern)
  - Recovery builds if necessary the next day
  - Must have a successful integration build every week

Eclipse Release Process – Rules of Engagement

- The “rules” change over time
  - Phase 0 – Any committer can release changes
  - Phase 1 - All changes must be verified by another committer
  - Phase 2 - Pre-approval required by 2 component leads, same checks
  - Phase 3 - Pre-approval required by all component leads, same checks
- Making it progressively harder has several benefits
  - Less code gets changed therefore less chance of regression
  - The most important defects are addressed
  - It’s a team effort

Checkpoint

- What have we covered so far:
  - The “plan” is a living document
  - Milestones are the rhythm of the project
  - Continuous integration is critical
  - Rules of Engagement
- What’s next?
  - Actually getting work done in an open source project

The ecology of an open source project

- It’s more than following the requirements of an OSI approved license
- Successful open source projects are an eco system that encourages many forms of participation
  - Newsgroups, Mailing lists, Bugzilla
  - Books and Articles
  - Code Camps and Conferences
  - Bug fixes
  - Testing
  - New projects based on eclipse technology

Working on an open source project

Working in public changes the way you think and feel about software development

- Immediate feedback (both positive and negative)
- It's a fantastic feeling to know that your work matters

The Eclipse open source project is a meritocracy

- All contributions are valuable
- Earning commit rights is open to everyone

How we are organized

- The platform is composed of many components
  - Each component is owned by a team of developers (3-9 people)
  - Developers may work on multiple components
  - Bugzilla is used to track defects
  - Each component has a component lead
  - Component teams are responsible for:
    - The quality of their component
    - Coming though on their commitments
    - Nominating developers who have earned their commit rights
- It is a distributed team
  - The current set of committers are distributed across 9 time zones
  - Contributions come from many sources

Milestones

- Each milestone is a miniature development cycle
  - Plan, Execute, Test, Review
  - Teams refer to the 3.0 plan when creating their milestone plans
    - Milestone plans are public as well
- At the end of each milestone we do a post-mortem
  - What went well, what did not?
  - Did we achieve the goals that were set for the milestone
  - Are there changes that we need to make to the 3.0 plan?
- Milestones reduce stress

Dynamic Teams

- Some plan items require coordination between team
  - High risk of failure
- A dynamic team is a temporary assignment
  - Created to solve a specific problem
  - Improves overall communication between component teams
- Like regular teams, a dynamic team also has a lead developer
  - Responsibility and expectations have to be clearly identified and agreed to

The Rules for “Practical” Software Development

1. If it is broken, fix it.
2. If it works can you do it better?
3. Remember it’s a team effort.
4. Keep your plans honest
5. Make your schedules realistic
6. Be predictable
7. Automate wherever you can
8. Learn from your mistakes
9. Code is more important than process
10. People are more important than code

Questions?

