Successful Community Development

Wayne Beaton
(narrated by Ralph Mueller)
Eclipse Foundation
Grenoble, Nov 10, 2010

Agenda

- Define Community
- Working with the Community
- Doing the right things
- Case Study

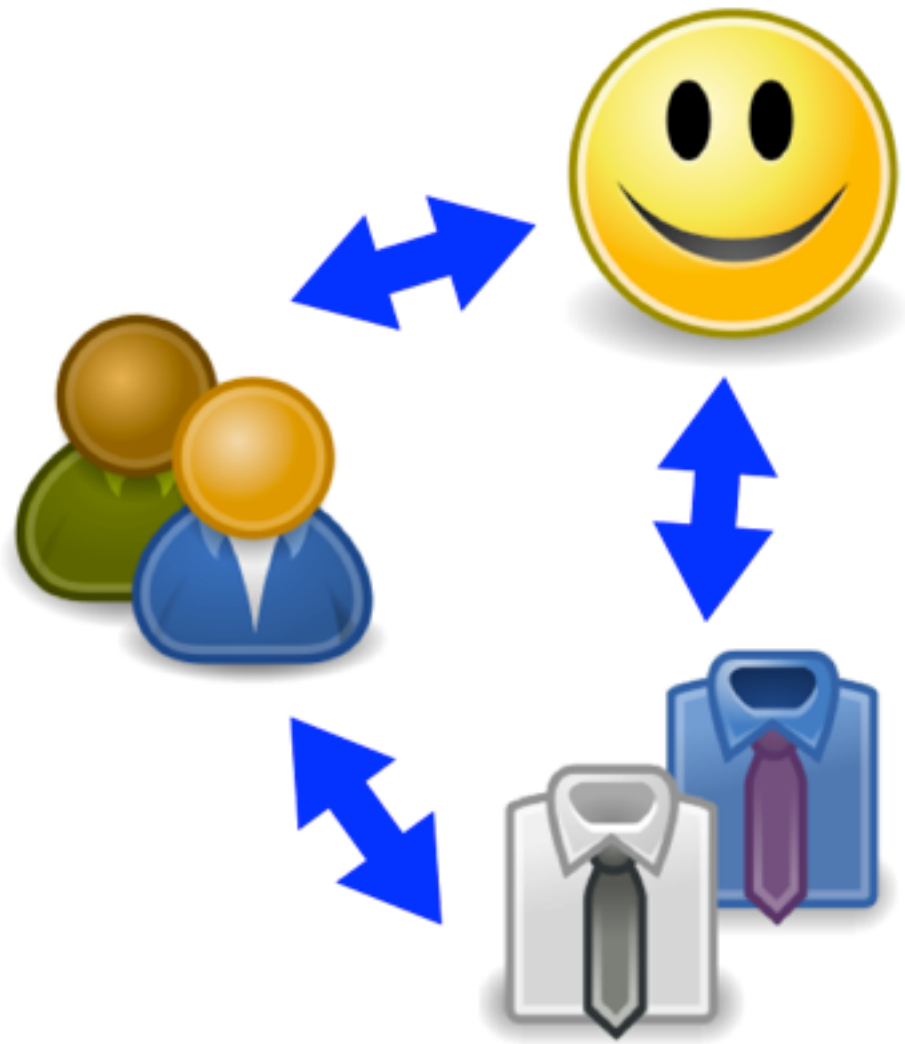
Define Community

Why Community?

- Shared Development burden
- Ubiquity of a Framework/Platform
- Acknowledge the Need!
  - Document it
  - Make it part of your project charter

Different Types

- End Users
- Adopters
- Committers



End Users

- Quality
- Information
- Documentation
- Easy to Find, Install, Use
- Support

Adopters

- Personalize and Extend
- Easy Programming Model
- Reliable APIs
- Low Barrier of Entry

Committers

- Be Part Of a Cool Project
- Low Barrier of Entry
- Align Project Goals with Own Goals
- Get Stuff Done

Working With The Community

Leadership

- Invite Contribution
- Mediate Conflicts and Diputes
- Set The Bar
- Balance (potentially) Opposing Viewpoints and Goals

Entry Barrier

- Can Everybody Be A Committer?
- Should It Be Difficult To Become A Committer?
- Can You Trust Your Committers?
- How Do You Establish Trust?

Growing Committers

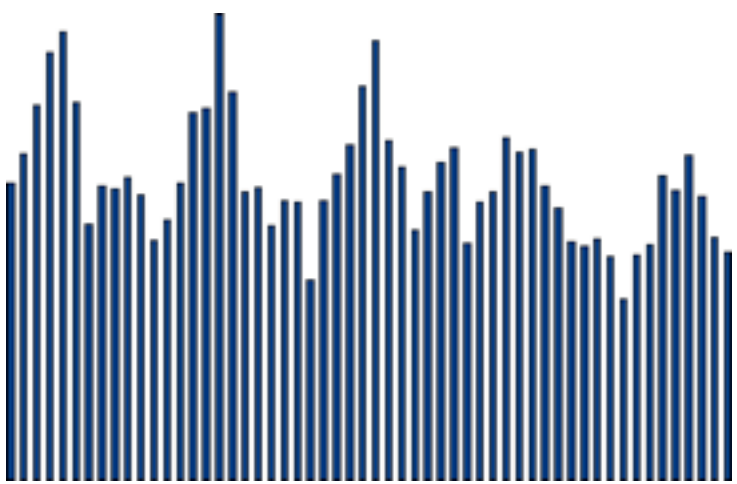
- Make Contribution As Easy As Possible
- Define Clear Processes Where
- Mentor and Educate
- Provide Sandbox

Diversity

- Generalization of Competing Needs and Goals
- Outlives Your Involvement
- Independence From Single Organization
- Now This Is Interesting to Corporations

Open And Transparent

- Everybody Can Participate (Code Speaks!)
- Many Ideas, Many Approaches, Many Use Cases
- Everybody Can See Everything
- Even Your Problems ... That‘s a Tough One



Realism

- How Large Is Your Potential Community?
- Is Your Project Niche Or Mainstream?
- Will Your Academic Research Attract Corporate?
- Plan for Transition to Industrial
- Define Success Realistically

Be Pro-Active

- Find The Community
- Planes, Trains And Automobiles ...
- T-Shirts Are A Good Start ...
- ... But They Only Take You So Far
- Demo Camps, Stammtisch, Webinars, User Groups, Bar Camps, ...

Doing The Right Things

Community Is Key

- Jour Fixe
- Little Things
- Bug Reports Are Love Letters
- Set Time Aside

Quality

- Good Enough Is Not Good Enough
- A Milestone Is A Milestone, A Promise Is A Promise
- Plan For Quality - And Expect The Same From Your Team
- Educate New Committers
- Be „Quality Driven“

License And IP

- Define Your Business Case
- Find The Appropriate License
- Oh - And Who Owns What?
- And How Do you Track It?
- Trust Is Good, Control Is Better (Lenin)

Access & PR

- Be Highly Visible
- Dowenloads
- Good And Up-To-Date Web Site
- Solicit Backlinks
- Aggregate Bloggers

ADVERTISE IT

- Blogs
- Forums
- Twitter, Facebook And The Likes
- Invite Others To Write
- Talk To The Media
- Market Yourself
- Buy Drinks As A Last Resort

Case Study

Community Driven

Erich Gamma: I think this is independent of open or closed-source. Software creates communities and transparent development is important if you want to grow a *community*. Open source in particular, though, is not just about making source available under some license; it is really about building up a community. And you build a community by showing them what you're up to, which means you **make your plans visible**. All of our milestone plans and project plans are visible on the web. All of our **bugs are visible**. The community really sees what's going on. Of course, what we hope for in return is that the community participates. And participation can come in many different forms—for example **providing feedback** in bug reports, contributing newsgroup replies, **providing patches**, implementing additional plug-ins, or writing articles. These are the ingredients of a tight feedback loop, and this kind of **feedback loop** is the key to having a good, shippable product in the end. The fact that Eclipse has such an active community is really cool and a major asset. Having such a community is an asset **no matter whether the environment is open-source or closed**.

Magic Number 42

Erich Gamma: We split the release cycle into milestones at a granularity of six weeks, and each milestone ends with an improved and useable Eclipse build. In general, those **six weeks** are like a small development cycle, in which we **plan, develop, and test**. With this kind of fractal major plan, we get in effect several small development cycles for each release. We slow down at the end of each milestone. We have a day where everybody gets out of the water and does testing. Doing testing for each milestone **avoids** that we accumulate a **larger testing effort** until the end of the release cycle. Then we document what's new and noteworthy, and we announce it to the community so **they can observe our progress** and provide early feedback. Then we plan the next milestone, taking into account both the overall plan and individual component plans.

A Bug Report Is A Love Letter

Erich Gamma: As far as the agile practices we follow when developing Eclipse, we always test early, often, and automated. For each build we run over 20,000 tests. We have nightly builds that are automated. We get build reports that tell us the failures. Recently in 3.1, we added performance tests. So we not only test for correctness, but also for performance. This has helped us a lot during the 3.1 cycle and actually you will notice significant performance improvements in version 3.1.

Impact

- Committers Spend 20 - 40 % Of Their Time On Community
- Management Forges Relationships With Other Organizations
- Outreach ... Outreach ... Outreach

Transparency And Openness

- Even Hallway Discussions Get Recorded
- PMC and Component Leads Meet Once A Week
- Meeting Notes Are Public
- Private Communication Is Deferred To Public Mailing Lists
- Sounds Easy, But Is Tough!


Eclipse Helios


Eclipse Helios
39 Projects
490 Committer
33 Million Lines of Code

Summary

- Know Your Target Community
- Know What You Want
- Have A Plan
- Be Repsonisve
- Be Open And Transparent
- Be Aware Of The Effort

Thank you

Thank You Wayne For The Insights!

ralph.mueller@eclipse.org