So you want to run a project in Eclipse?

Doug Gaff
Wind River Systems
Introductions

- How many of you are new to Eclipse?
- How many of you are proposing a project?
- How many of you are already running a project?

- Who am I?
  - Device Software Development Platform (DSDP) PMC Lead
  - Wind River Sr. Engineering Manager
  - EclipseCon 2008 Technical Program Chair
  - Newly-elected Eclipse Board Member (Committer Rep.)
About this tutorial

- It’s a jumping off point, not a cookbook.
- I’ll try to make it a living presentation with updates as there are process changes.
- This tutorial covers things in the order you will see them.
- I do not cover
  - Eclipse membership, governance, or stats!
  - Some nitty gritty mechanical details.
- I will skip some detail that’s only meant for reference.
- Please ask questions along the way.
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way

Most of this presentation
Successful projects – a partial list

- **Eclipse Platform**
  - Oldest and very mature technically
  - Largest committer population (but not diverse)
  - The minimum technology required to use and/or extend Eclipse
  - Consistent and well planned

- **CDT**
  - Almost as old as the platform
  - Extremely large commercial adoption, especially in embedded
  - Feature mature
  - Much smaller committer base, but very diverse
Successful projects – a partial list

- **Web Tools**
  - Fundamental, much needed technology
  - Functional breadth

- **Modeling**
  - One of the “next big things”
  - Good committer diversity
  - Functional breadth

- **Mylyn**
  - Freakin’ cool technology
  - Strong evangelism
  - Commitment to rapid improvement
Successful projects – why?

- Support the 3 Communities (Contributor / Adopter / User)
- Transparency
- Diversity

- Technology demand
- Trajectory toward technical and process maturity

- Alignment between open source and commercial product offerings (will discuss in detail later)

- “The Eclipse Way” – the subject of the rest of this presentation

*Not all successful projects are adept at following The Eclipse Way*
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Three Eclipse Communities

1. **Committer / Contributor**
   - Project leads and developers
   - Often (but not always) from Strategic or Add-in Provider Members
   - Company diversity (ideally) on a per-project basis

2. **User**
   - The “Free Beer” community
   - Might submit bug reports or occasional patches

3. **Adopter**
   - Plug-in providers – the ecosystem
   - Commercialize Eclipse projects and technology
Why Worry About These Communities?

- **Committer / Contributor**
  - The software’s not going to build itself!
- **User**
  - Your external testers. Are the exemplary tools beefy enough?
  - Free bug reporting and enhancement ideas – costs less than a commercial customer, and it’s faster to get.
- **Adopter**
  - Eclipse survives because companies adopt open source commercially.
  - Eclipse is about ecosystems of products built on frameworks.
  - Frameworks can create open standards.

*Good technology abhors a vacuum!*
More About Committers

- **Rights**
  - Write access to CVS / SVN repositories for your project.
  - Commit rights are a privilege earned through a “meritocracy” – *The more you contribute the more responsibility you will earn.*

- **Responsibilities**
  - Architectural oversight
  - IP Process adherence
  - Community responsiveness (newsgroups, bugs, dev-lists, etc.)
  - Applying patches from contributors
  - Project, technology, ecosystem evangelism
Three (Committer) Laws of Eclipse

1. A committer **may not**, through action or inaction, **violate IP cleanliness**
   - Clean Code – legally contributed in a correct manner
   - Approved Libraries – all 3rd party libraries approved by Eclipse legal
   - “Papered” Committers – all legal paperwork signed

2. A committer **may not**, through action or inaction, **disenfranchise contributors**
   - Public Queue – everything tracked in Bugzilla
   - Available tools – anyone can build the project
   - Transparent elections – open, public, justified

3. A committer **may not**, through action or inaction, **surprise the membership**
   - Open Plans – public planning process and published plans
   - Public Reviews – all major releases (0.N, 1.N, 2.N, etc.)
   - Prominent Announcements – new development, releases

Community Tension

- **Committers**
  - “We’re going to build what we want.”
  - “If you want it, do it yourself.”
  - “Should we build feature X in the open or in our product?”

- **Adopters**
  - “This framework is missing…”
  - “Don’t give away that! We sell that.”

- **Users**
  - “This technology sucks!”
  - “Fix my @$%# bug”
  - “Please add this feature”
Community Tension

- **Fact 1:** Eclipse exists to enable commercial adoption
- **Fact 2:** This goal is the singular driver of Eclipse growth
- **Fact 3 (BAD!!):** Most new projects are single-company sponsored
  - Framework and exemplary tool biased towards company
  - Company is the primary developer and adopter (initially)
  - Company decides the “commodity line”

**Result**
- Users may not get much out of the exemplary tools
- Other adopters may not benefit from the framework design

_You’re an open source project now. Don’t forget: competitors need to adopt your frameworks and users need enough tooling to prove your technology (and maybe more)._
“Coopetition” – two scenarios

1. **Competitive Weapon** – create a free low-end version to drive commodity among competitors, e.g. “competitive weapon”

2. **Reduced Cost of Goods (COG)** – companies collaborate on base technology and frameworks – value added “above the line”
   - Successful projects work with their competitors.
   - The “commodity line” is established by whomever is willing to give the most stuff away. Line moves over time.
   - Frameworks must be multi-vendor.

*Project leads who understand this evangelize it inside their companies.*

*Companies who understand this build an open source strategy.*

*Companies with an open source strategy are the future of software.*
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Project Lifecycle

www.eclipse.org/projects/dev_process/development_process.php
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Phase 0: Declaration (Pre-proposal)

Requirements

1. Does it already exist in Eclipse? If yes, go talk to that project.
2. What top-level project does it augment? If none, then go to Technology.
3. Contact the PMC of the desired top-level project and give them your pitch.

www.eclipse.org/projects/dev_process/pre-proposal-phase.php
Phase 0: Declaration (Pre-proposal)

4. Write up a project proposal
   - www.eclipse.org/projects/dev_process/pre-proposal-phase.php

5. Get it approved by the EMO
   - Requires PMC and EMO review and multiple edits.
   - The PMC should help you with this.

6. It gets approved and posted
   - www.eclipse.org/proposals/project
Good Project Proposals / Common Pitfalls

- **Good**
  - Clear, concise, **realistic scope**
  - Unique value add to existing Eclipse technology base
  - Commercial contribution to open source, and at least one commercial supporter

- **Bad**
  - “Kitchen sink” / “save the world” proposals
  - Duplicative technology (sometimes ok)
  - An idea with zero code behind it – you’re in for an uphill climb
  - An idea with a fully-formed product code base (sometimes ok)

_Eclipse philosophy: 1) easy to propose projects, 2) slightly harder to create them, 3) hard to exit incubation, 4) tough to ship_
Proposal Requirements (dev process)

1. Project must be **in scope** of its parent project.
2. **Dependencies** and overlap with other projects must be described.
3. Technical overlap with other projects is discouraged.
   - Competing projects often combine before graduating
4. New efforts must be **announced to** membership and not hidden.
5. Projects must include exemplary **tools** and extensible **frameworks**.
6. Project must have a **memorable and descriptive project name**.
7. There must be a **resource commitment** to the project.
8. Project must be **vendor neutral**.
9. Two **Mentors** from the architecture council are required.

- A +1 from the proposer, Bjorn, and Mike is required to post.

www.eclipse.org/projects/dev_process/pre-proposal-phase.php
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Phase 1: Proposal (Gathering Community)

- Work *in public* to enhance, refine, and clarify the proposal
  - Advertise on your project’s newsgroup
  - Advertise on related mailing lists for other projects (including non-Eclipse)
  - Recruit potential contributors through networking (other Eclipse leaders, EMO, business contacts)
- Hold public discussions
- (Start your legal paperwork)

www.eclipse.org/projects/dev_process/proposal-phase.php
Ready for Creation Review vs. Proposal Limbo

- Signs of “sufficient interest”
  - Sponsoring company has engineers ready to go
  - Technical ideas and/or initial code based established
  - More than one company interested in participating
  - Scope, tools, and frameworks are refined and clear
  - Newsgroup activity!

- Signs of “withering on the vine”
  - Stale proposals
  - Sponsoring company not ready
  - A “time slicing” (proposed) project lead
  - Newsgroup silence

Judgment at the discretion of Mentors, proposed project lead, EMO, and PMC.
Creation Review - Content

- An archival quality presentation – “docuware”
  - Executive summary
  - Mentors listed
  - Initial Committer List with bios and justification
  - Communities – speculate outside of your developers
  - Diversity – not required, but highly recommended
  - Technical scope justification - it’s realistic and achievable
  - Maturity plan – timescale, approximate release roadmap
  - Future directions
- Neutral file format (PDF)
- Correct copyright and license
  - © by you or your employer, made available under EPL v1.0
Creation Review – Process

- Reviews are done twice a month.
- Schedule by emailing emo@eclipse.org
- Send “docuware” at least one week before the review – it’s posted
- EMO announces the review to the community

- Review happens on an (optional) conference call
  - Call scheduled, but only happens if the community requests it.
  - Usually don’t give your slides, just answer questions.
  - Looking for statements from the community that are not in favor of the proposal, rather than a positive vote.

- If there are concerns (unusual), the proposal is revised and you try again.
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Phase 2: Incubation

- Skills (in the order you’ll use them)
  - Provisioning and Infrastructure
  - IP Process
  - Community Building
  - Using Bugzilla
  - Development Process
  - Release Review

This is your 0.X release phase.
Incubation Requirements

- IP Due Diligence
- API and Framework Development
  - Develop for the 3 communities
- Regular Milestones
  - Not required, but recommended, e.g. 0.1M1, 0.1M2, etc.
  - Gives predicable early access to your community
- Interim Releases
  - Only 0.N releases allowed (0.1, 0.2, etc.)
  - Recommend one interim release before attempting a 1.0 release (Graduation)
- “Incubation Conforming”
  - Logo on home and download pages
  - Downloads: DSF-SDK-incubation-N20070413-0200.zip
  - Bundle-Name: Foo Plug-in (Incubation)
  - Update Manager: Foo Platform (Incubation)
Phase 2: Incubation

- Skills
  - Provisioning and Infrastructure
  - IP Process
  - Community Building
  - Using Bugzilla
  - Development Process
  - Release Review
Starting Incubation – Provisioning Request Form

- [www.eclipse.org/projects/project_provisioning_request.php](http://www.eclipse.org/projects/project_provisioning_request.php)
- CVS / SVN repositories
  - org.eclipse.*project.component1*
  - org.eclipse.*project.component2*
  - org.eclipse.*project.component3*
- Recommended Structure
  - `/cvsroot/toplevel`
    - `org.eclipse.project`
      - `org.eclipse.*project.component`
      - `plugins/`
      - `freatures/`
    - `...`
      - `org.eclipse.project.relen`
    - `...`
      - `relen-common`
Starting Incubation – Provisioning Request Form

- Initial Committer list
  - As nominated in the Creation Review
  - Committer paperwork required (hopefully done during Proposal Phase)
- Website area
  - [www.eclipse.org/toplevel/project](http://www.eclipse.org/toplevel/project)
  - URL hierarchy is optional, but often useful
Starting Incubation – Provisioning Request Form

- Mailing list(s)
  - `toplevel-pmc@eclipse.org`
  - `toplevel-project-dev@eclipse.org`
  - `toplevel-project-build@eclipse.org`
  - etc.

- Downloads area – FTP access
  - `download.eclipse.org/toplevel/project/downloads`
  - You have to create the html page

- Bugzilla components
  - `component1, component2`, etc.

- Newsgroup was created during Proposal Phase
  - `eclipse.toplevel.project`
Immediately after Provisioning

- Create a skeleton website with the incubation graphic
  - The easiest way to start is to copy an existing website.
- Portal (accessed with your committer ID and password)
  - portal.eclipse.org
- Submit a IP Contribution Questionnaire (CQ) for your initial code contribution.
- Update the Project Info Meta-Data (see screen shot)
  - www.eclipse.org/projects/dev_process/project-status-infrastructure.php
  - project-info.xml file is deprecated!
- Setup a Wiki
  - wiki.eclipse.org/toplevel/project
- Get your blog aggregated with Planet Eclipse
  - File a bugzilla: Community – PlanetEclipse.org component
### Project Meta-Data (from Portal)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>article url</td>
<td><a href="http://wiki.eclipse.org/index.php/DSDP/TM">http://wiki.eclipse.org/index.php/DSDP/TM</a></td>
</tr>
<tr>
<td>blog</td>
<td><a href="http://tmober.blogspot.com/atom.xml">http://tmober.blogspot.com/atom.xml</a></td>
</tr>
<tr>
<td>bugzilla</td>
<td><a href="https://bugs.eclipse.org">https://bugs.eclipse.org</a> ... gi?product=Target%20Management</td>
</tr>
<tr>
<td>committers url</td>
<td>/dspd/tm/development/contributors.php</td>
</tr>
<tr>
<td>contributing url</td>
<td><a href="http://wiki.eclipse.org/TM_and_RSE_FAQ">http://wiki.eclipse.org/TM_and_RSE_FAQ</a></td>
</tr>
<tr>
<td>contributors url</td>
<td>/dspd/tm/development/contributors.php</td>
</tr>
<tr>
<td>source repository</td>
<td>/cvsroot/dspd/org.eclipse.tm.core/, /cvsroot/dspd/org.eclipse.tm.rse, /dspd/org.eclipse.tm.tcf</td>
</tr>
<tr>
<td>description url</td>
<td>/dspd/tm/project-info/project-page-paragraph.html</td>
</tr>
<tr>
<td>gettingstarted url</td>
<td>/dspd/tm/tutorial/index.php</td>
</tr>
<tr>
<td>ip log url</td>
<td>/dspd/tm/development/tm-log.csv</td>
</tr>
<tr>
<td>leaders url</td>
<td>/dspd/tm/development/contributors.php</td>
</tr>
<tr>
<td>mailing list</td>
<td>dsdp-tm-dev, dsdp-tm-cvs-commit, dsdp-tm-svn-commit</td>
</tr>
<tr>
<td>newsgroup</td>
<td>eclipse.dspd.tm</td>
</tr>
<tr>
<td>paragraph url</td>
<td>/dspd/tm/project-info/project-page-paragraph.html</td>
</tr>
<tr>
<td>project name</td>
<td>Target Management</td>
</tr>
<tr>
<td>project plan url</td>
<td><a href="http://www.eclipse.org/dspd/tm/plan.php">http://www.eclipse.org/dspd/tm/plan.php</a></td>
</tr>
<tr>
<td>project short name</td>
<td>Target Management</td>
</tr>
<tr>
<td>project url</td>
<td><a href="http://www.eclipse.org/dspd/tm/">http://www.eclipse.org/dspd/tm/</a></td>
</tr>
<tr>
<td>release</td>
<td>1.0.1, 1.0, 2.0, 2.0.0.1, 2.0.1, 2.0.2, 2.0.3, 3.0</td>
</tr>
<tr>
<td>summary url</td>
<td>/dspd/tm/project-info/executive-summary.html</td>
</tr>
<tr>
<td>team url</td>
<td>/dspd/tm/development/contributors.php</td>
</tr>
<tr>
<td>update site url</td>
<td><a href="http://download.eclipse.org/dspd/tm/updates/2.0">http://download.eclipse.org/dspd/tm/updates/2.0</a></td>
</tr>
</tbody>
</table>
Infrastructure Summary – Website Properties

- Eclipse.org
  - www.eclipse.org/toplevel/project – project home page
  - toplevel.eclipse.org – virtual server
- Eclipse Plugin Central (EPIC)
  - www.eclipseplugincentral.com
- Blogs – Planet Eclipse
  - www.planeteclipse.org
- Wiki
  - wiki.eclipse.org/toplevel/project
- Eclipse Live
  - live.eclipse.org
  - Podcasts, webinars, etc.
Infrastructure Summary – Code and Bugs

- Source Repositories
  - CVS
  - SVN
  - (You can convert from CVS to SVN once!)
- CVS vs. SVN
  - Many folks still recommend CVS because the Eclipse Platform support is better.
- Features, Bugs, IP Submissions
  - Bugzilla (bugs.eclipse.org/bugs)
  - IPzilla (see portal)
Infrastructure Summary – Communication

- Communication
  - Mailing lists (www.eclipse.org/mail)
  - Newsgroups (www.eclipse.org/newsgroups)
  - IRC (wiki.eclipse.org/index.php/IRC)
- Blogs, Wiki, EclipseLive
  - and other web properties are all good for communications
Infrastructure Summary – Project Management Tools

- Committer Tools
  - [www.eclipse.org](http://www.eclipse.org) then click on “Committers”
  - Password changes, Unix groups, Bugzilla components

- Portal
  - [portal.eclipse.org](http://portal.eclipse.org)
  - IPZilla Contribution Questionnaire (CQ) filing
  - Committer votes
  - Contact info

- Project Dashboards
  - [dash.eclipse.org/dash/commits/web-app](http://dash.eclipse.org/dash/commits/web-app)
  - (screen shots to follow)
### Lines of Change by Projects by Month

<table>
<thead>
<tr>
<th>Project</th>
<th>Lines of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>birt</td>
<td></td>
</tr>
<tr>
<td>datatools</td>
<td></td>
</tr>
<tr>
<td>datatools.connectivity</td>
<td></td>
</tr>
<tr>
<td>datatools.enablement</td>
<td></td>
</tr>
<tr>
<td>datatools.modelbase</td>
<td></td>
</tr>
<tr>
<td>datatools.sqltools</td>
<td></td>
</tr>
<tr>
<td>dsdp.dd</td>
<td></td>
</tr>
<tr>
<td>dsdp.ercp</td>
<td></td>
</tr>
<tr>
<td>dsdp.mtj</td>
<td></td>
</tr>
<tr>
<td>dsdp.nab</td>
<td></td>
</tr>
<tr>
<td>dsdp.tm</td>
<td></td>
</tr>
<tr>
<td>dsdp.tml</td>
<td></td>
</tr>
<tr>
<td>eclipse</td>
<td></td>
</tr>
<tr>
<td>eclipse.equinox</td>
<td></td>
</tr>
<tr>
<td>eclipse.jdt</td>
<td></td>
</tr>
<tr>
<td>eclipse.pde</td>
<td></td>
</tr>
<tr>
<td>eclipse.platform</td>
<td></td>
</tr>
</tbody>
</table>
### Lines of Change by Committer for Project dsdp.tm

#### By Login

<table>
<thead>
<tr>
<th>Login</th>
<th>One</th>
<th>Three</th>
<th>Six</th>
<th>Nine</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddykstal</td>
<td>409</td>
<td>7,069</td>
<td>7,440</td>
<td>10,076</td>
</tr>
<tr>
<td>dgaff</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dmcknight</td>
<td>8,124</td>
<td>13,079</td>
<td>23,836</td>
<td>28,347</td>
</tr>
<tr>
<td>eutarass</td>
<td>633</td>
<td>633</td>
<td>633</td>
<td>633</td>
</tr>
<tr>
<td>jmontalvo</td>
<td></td>
<td></td>
<td>503</td>
<td>978</td>
</tr>
<tr>
<td>kdoyle</td>
<td>880</td>
<td>923</td>
<td>1,535</td>
<td>1,535</td>
</tr>
<tr>
<td><em>kimanir</em></td>
<td></td>
<td></td>
<td></td>
<td>81</td>
</tr>
<tr>
<td>mischarf</td>
<td>30</td>
<td>62</td>
<td>13,877</td>
<td>48,512</td>
</tr>
<tr>
<td>moberhuber</td>
<td>7,589</td>
<td>9,193</td>
<td>38,528</td>
<td>46,587</td>
</tr>
<tr>
<td><em>root</em></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><em>tewillia</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ustieber</td>
<td>33</td>
<td>33</td>
<td>332</td>
<td>620</td>
</tr>
<tr>
<td>xchen</td>
<td>167</td>
<td>2,096</td>
<td>7,198</td>
<td>12,822</td>
</tr>
</tbody>
</table>

#### By 3-6-9 Month Activity

<table>
<thead>
<tr>
<th>Login</th>
<th>One</th>
<th>Three</th>
<th>Six</th>
<th>Nine</th>
<th>(9-6-3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ddykstal</td>
<td>409</td>
<td>7,069</td>
<td>7,440</td>
<td>10,076</td>
<td></td>
</tr>
<tr>
<td>dmcknight</td>
<td>8,124</td>
<td>13,079</td>
<td>23,836</td>
<td>28,347</td>
<td></td>
</tr>
<tr>
<td>eutarass</td>
<td>633</td>
<td>633</td>
<td>633</td>
<td>633</td>
<td></td>
</tr>
<tr>
<td>kdoyle</td>
<td>880</td>
<td>923</td>
<td>1,535</td>
<td>1,535</td>
<td></td>
</tr>
<tr>
<td>mischarf</td>
<td>30</td>
<td>62</td>
<td>13,877</td>
<td>48,512</td>
<td></td>
</tr>
<tr>
<td>moberhuber</td>
<td>7,589</td>
<td>9,193</td>
<td>38,528</td>
<td>46,587</td>
<td></td>
</tr>
<tr>
<td><em>root</em></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ustieber</td>
<td>33</td>
<td>33</td>
<td>332</td>
<td>620</td>
<td></td>
</tr>
<tr>
<td>xchen</td>
<td>167</td>
<td>2,096</td>
<td>7,198</td>
<td>12,822</td>
<td></td>
</tr>
<tr>
<td><em>kimanir</em></td>
<td></td>
<td></td>
<td></td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>dgaff</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>tewillia</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## commits by top by year

<table>
<thead>
<tr>
<th></th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>birt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44921</td>
<td>56807</td>
<td>38581</td>
<td>5385</td>
</tr>
<tr>
<td>datatools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5013</td>
<td>9050</td>
<td>19051</td>
<td>2346</td>
</tr>
<tr>
<td>dsdp</td>
<td></td>
<td></td>
<td>925</td>
<td></td>
<td>13728</td>
<td>24458</td>
<td>29508</td>
<td>5419</td>
</tr>
<tr>
<td>eclipse</td>
<td>82804</td>
<td>128460</td>
<td>161895</td>
<td>172124</td>
<td>201763</td>
<td>128148</td>
<td>114273</td>
<td>21722</td>
</tr>
<tr>
<td>modeling</td>
<td>765</td>
<td>9034</td>
<td>39176</td>
<td>92915</td>
<td>217277</td>
<td>215954</td>
<td>24930</td>
<td></td>
</tr>
<tr>
<td>stp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>technology</td>
<td>182</td>
<td>452</td>
<td>1086</td>
<td>8362</td>
<td>204000</td>
<td>628350</td>
<td>663045</td>
<td>198572</td>
</tr>
<tr>
<td>tools</td>
<td>7841</td>
<td>25552</td>
<td>36897</td>
<td>59748</td>
<td>93410</td>
<td>87656</td>
<td>121965</td>
<td>53159</td>
</tr>
<tr>
<td>tppp</td>
<td></td>
<td></td>
<td></td>
<td>62162</td>
<td>55099</td>
<td>26704</td>
<td>8934</td>
<td></td>
</tr>
<tr>
<td>webtools</td>
<td>755</td>
<td>23996</td>
<td>90083</td>
<td>83119</td>
<td>112027</td>
<td>19080</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

# SELECT YEAR AS X, TOPPROJECT AS Y, COUNT(*) AS COUNT FROM commits
GROUP BY YEAR, TOPPROJECT

This automatically collected information may not represent true activity and should not be used as sole indicator of individual or project behavior. See the [wiki page](#) about known data anomalies. See [details](#) for all items in the table. See [raw data](#) we use.
## percent of lines of change by project by company by year

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dsdp_dd</td>
<td>ARMLimited</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EricssonAB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>29%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WindRiver</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>70%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsdp_ercp</td>
<td>IBM</td>
<td>85%</td>
<td>69%</td>
<td>53%</td>
<td>85%</td>
<td>99%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nokia</td>
<td>14%</td>
<td>27%</td>
<td>44%</td>
<td>1%</td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ProsystSoftware</td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>individual</td>
<td>2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>13%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsdp_mti</td>
<td>IBM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nokia</td>
<td>36%</td>
<td>92%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>48%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsdp_nab</td>
<td>FujitsuLimited</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsdp_tm</td>
<td>IBM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Symbian</td>
<td>1%</td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WindRiver</td>
<td>31%</td>
<td>60%</td>
<td>35%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>individual</td>
<td>9%</td>
<td>2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>unknown</td>
<td>1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dsdp_tml</td>
<td>Motorola</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>eclipse</td>
<td>IBM</td>
<td>100%</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Company / Project Commit Details

This automatically collected information may not represent true activity and should not be used as sole indicator of individual or project behavior.

See the wiki page for known data anomalies. To report issues or request enhancements, see bug 209711.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ActuateCorporation</td>
<td>27 (6.29%)</td>
<td>17 (3.41%)</td>
<td>44 (4.74%)</td>
<td>61.36%</td>
<td>6,120 (1.8%)</td>
<td>215,725 (1.85%)</td>
<td>35</td>
</tr>
<tr>
<td>AnywareTechnologies</td>
<td>3 (0.7%)</td>
<td>0 (0%)</td>
<td>3 (0.32%)</td>
<td>100%</td>
<td>418 (0.12%)</td>
<td>5,565 (0.05%)</td>
<td>13</td>
</tr>
<tr>
<td>BEA</td>
<td>5 (1.16%)</td>
<td>2 (0.4%)</td>
<td>7 (0.75%)</td>
<td>71.43%</td>
<td>453 (0.13%)</td>
<td>8,317 (0.07%)</td>
<td>18</td>
</tr>
<tr>
<td>BorlandSoftwareCorp.</td>
<td>11 (2.56%)</td>
<td>1 (0.2%)</td>
<td>12 (1.29%)</td>
<td>91.67%</td>
<td>5,264 (1.55%)</td>
<td>400,836 (3.44%)</td>
<td>76</td>
</tr>
<tr>
<td>CloudsmithInc.</td>
<td>4 (0.93%)</td>
<td>0 (0%)</td>
<td>4 (0.43%)</td>
<td>100%</td>
<td>3,029 (0.89%)</td>
<td>5,517 (0.05%)</td>
<td>2</td>
</tr>
<tr>
<td>comppeopleAG</td>
<td>3 (0.7%)</td>
<td>0 (0%)</td>
<td>3 (0.32%)</td>
<td>100%</td>
<td>872 (0.26%)</td>
<td>9,341 (0.08%)</td>
<td>11</td>
</tr>
<tr>
<td>Compulogic</td>
<td>2 (0.47%)</td>
<td>2 (0.4%)</td>
<td>4 (0.43%)</td>
<td>50%</td>
<td>990 (0.29%)</td>
<td>3,204 (0.03%)</td>
<td>3</td>
</tr>
<tr>
<td>EricssonAB</td>
<td>2 (0.47%)</td>
<td>1 (0.2%)</td>
<td>3 (0.32%)</td>
<td>66.67%</td>
<td>181 (0.06%)</td>
<td>15,247 (0.13%)</td>
<td>84</td>
</tr>
<tr>
<td>FujitsuLimited</td>
<td>1 (0.23%)</td>
<td>2 (0.4%)</td>
<td>3 (0.32%)</td>
<td>33.33%</td>
<td>362 (0.11%)</td>
<td>2,522 (0.02%)</td>
<td>7</td>
</tr>
<tr>
<td>GeminiTechLLC</td>
<td>1 (0.23%)</td>
<td>0 (0%)</td>
<td>1 (0.11%)</td>
<td>100%</td>
<td>192 (0.06%)</td>
<td>3,114 (0.03%)</td>
<td>16</td>
</tr>
<tr>
<td>IBM</td>
<td>153 (35.58%)</td>
<td>168 (33.67%)</td>
<td>321 (34.55%)</td>
<td>47.66%</td>
<td>154,536 (45.5%)</td>
<td>6,702,713 (57.5%)</td>
<td>43</td>
</tr>
<tr>
<td>individual</td>
<td>121 (28.14%)</td>
<td>195 (39.08%)</td>
<td>316 (34.02%)</td>
<td>38.29%</td>
<td>105,610 (31.1%)</td>
<td>2,087,269 (17.91%)</td>
<td>20</td>
</tr>
<tr>
<td>Innooprac</td>
<td>6 (1.4%)</td>
<td>1 (0.2%)</td>
<td>7 (0.75%)</td>
<td>85.71%</td>
<td>1,540 (0.45%)</td>
<td>46,279 (0.4%)</td>
<td>30</td>
</tr>
<tr>
<td>IntaliorInc.</td>
<td>1 (0.23%)</td>
<td>1 (0.2%)</td>
<td>2 (0.22%)</td>
<td>50%</td>
<td>28 (0.01%)</td>
<td>150 (0%)</td>
<td>5</td>
</tr>
<tr>
<td>IntelCorporation</td>
<td>7 (1.63%)</td>
<td>3 (0.6%)</td>
<td>10 (1.06%)</td>
<td>70%</td>
<td>791 (0.23%)</td>
<td>36,578 (0.31%)</td>
<td>46</td>
</tr>
<tr>
<td>ITONeTechnologies</td>
<td>3 (0.7%)</td>
<td>2 (0.4%)</td>
<td>5 (0.54%)</td>
<td>60%</td>
<td>280 (0.08%)</td>
<td>3,961 (0.03%)</td>
<td>14</td>
</tr>
<tr>
<td>JavaMedical</td>
<td>2 (0.47%)</td>
<td>0 (0%)</td>
<td>2 (0.22%)</td>
<td>100%</td>
<td>380 (0.11%)</td>
<td>10,490 (0.09%)</td>
<td>28</td>
</tr>
<tr>
<td>Motorola</td>
<td>1 (0.23%)</td>
<td>0 (0%)</td>
<td>1 (0.11%)</td>
<td>100%</td>
<td>127 (0.04%)</td>
<td>542 (0%)</td>
<td>4</td>
</tr>
<tr>
<td>Nokia</td>
<td>3 (0.7%)</td>
<td>3 (0.6%)</td>
<td>6 (0.65%)</td>
<td>50%</td>
<td>131 (0.04%)</td>
<td>2,820 (0.02%)</td>
<td>22</td>
</tr>
<tr>
<td>OBEO</td>
<td>7 (1.63%)</td>
<td>1 (0.2%)</td>
<td>8 (0.86%)</td>
<td>87.5%</td>
<td>3,477 (1.02%)</td>
<td>227,283 (1.95%)</td>
<td>65</td>
</tr>
<tr>
<td>Oracle</td>
<td>28 (6.51%)</td>
<td>5 (1%)</td>
<td>33 (3.66%)</td>
<td>84.85%</td>
<td>39,230 (11.56%)</td>
<td>827,437 (7.1%)</td>
<td>21</td>
</tr>
<tr>
<td>ProsysSoftware</td>
<td>3 (0.7%)</td>
<td>0 (0%)</td>
<td>3 (0.32%)</td>
<td>100%</td>
<td>187 (0.06%)</td>
<td>993 (0.01%)</td>
<td>5</td>
</tr>
<tr>
<td>SAPAG</td>
<td>2 (0.47%)</td>
<td>0 (0%)</td>
<td>2 (0.22%)</td>
<td>100%</td>
<td>237 (0.07%)</td>
<td>5,513 (0.05%)</td>
<td>24</td>
</tr>
<tr>
<td>SAS</td>
<td>2 (0.47%)</td>
<td>1 (0.2%)</td>
<td>3 (0.32%)</td>
<td>66.67%</td>
<td>8 (0%)</td>
<td>46 (0%)</td>
<td>6</td>
</tr>
<tr>
<td>SerenaSoftwareInc.</td>
<td>3 (0.7%)</td>
<td>1 (0.2%)</td>
<td>4 (0.43%)</td>
<td>75%</td>
<td>3,331 (0.98%)</td>
<td>685,647 (5.88%)</td>
<td>206</td>
</tr>
<tr>
<td>Sybase</td>
<td>6 (1.4%)</td>
<td>1 (0.2%)</td>
<td>7 (0.75%)</td>
<td>85.71%</td>
<td>1,035 (0.3%)</td>
<td>27,615 (0.24%)</td>
<td>27</td>
</tr>
<tr>
<td>Symantec</td>
<td>1 (0.23%)</td>
<td>1 (0.2%)</td>
<td>2 (0.22%)</td>
<td>50%</td>
<td>260 (0.08%)</td>
<td>7,753 (0.07%)</td>
<td>30</td>
</tr>
<tr>
<td>Thales</td>
<td>1 (0.23%)</td>
<td>1 (0.2%)</td>
<td>2 (0.22%)</td>
<td>50%</td>
<td>1,974 (0.58%)</td>
<td>11,564 (0.1%)</td>
<td>6</td>
</tr>
<tr>
<td>unknown</td>
<td>2 (0.47%)</td>
<td>56 (11.22%)</td>
<td>58 (6.24%)</td>
<td>3.45%</td>
<td>5 (0%)</td>
<td>60 (0%)</td>
<td>6</td>
</tr>
<tr>
<td>Versant</td>
<td>1 (0.23%)</td>
<td>1 (0.2%)</td>
<td>2 (0.22%)</td>
<td>50%</td>
<td>118 (0.03%)</td>
<td>261 (0%)</td>
<td>2</td>
</tr>
<tr>
<td>WindRiver</td>
<td>10 (2.33%)</td>
<td>2 (0.4%)</td>
<td>12 (1.29%)</td>
<td>83.33%</td>
<td>5,621 (1.66%)</td>
<td>86,154 (0.74%)</td>
<td>15</td>
</tr>
<tr>
<td>ZendTechnologies</td>
<td>8 (1.86%)</td>
<td>0 (0%)</td>
<td>8 (0.86%)</td>
<td>100%</td>
<td>2,825 (0.83%)</td>
<td>216,240 (1.86%)</td>
<td>77</td>
</tr>
</tbody>
</table>

Total                   | 430               | 499                 | 929              | 339,617       | 11,656,828     |
Phase 2: Incubation

- Skills
  - Provisioning and Infrastructure
  - IP Process
  - Community Building
  - Using Bugzilla
  - Development Process
  - Release Review
Contributors: Developers vs. Committers

- Anyone who makes a contribution to the Eclipse Foundation website or to Eclipse projects is considered a *Contributor*.
- For Contributors, Eclipse differentiates between *Developers* and *Committers*.
- Committers have more responsibilities than developers.
  - Check-in rights to parts of the project’s CVS repository
  - Active participation in mailing lists and newsgroups
  - Voting in discussions
  - Code review of developer submissions
  - IP due diligence on submissions and assisting project lead in tracking IP
- Developers who provide frequent and valuable contributions to a project can be nominated to be committers.
- The *Eclipse Standard Charter* describes the roles of developers and committers and the process for nominating and voting on committers.

More on Committers

- Process for becoming a new Committer

- Member Committer
  - A committer who works for an Eclipse member company
  - Company must execute a Member Committer Agreement
    www.eclipse.org/legal/EclipseMemberCommitterAgreementFinal.pdf
  - Each member committer must fill out a Member Committer Questionnaire
    www.eclipse.org/legal/committer_process/member_committer_questionnaire.php

- Individual Committer
  - A committer not working for an Eclipse member company
  - Committer must execute an Individual Committer Agreement
  - Employer must execute a Committer Employer Consent Form
  - Committer must fill out an Individual Committer Questionnaire
    www.eclipse.org/legal/committer_process/individual_committer_questionnaire.php

- Committer guidelines
Code submission and IP Due Diligence

1. Code Submission
   - Any contributor can submit code to a project.
   - All contributions from non-committers MUST come through a traceable method.
   - While mailing list and newsgroup submissions are technically ok, the preferred and recommended submission method is via **Bugzilla attachments**. Contributor implicitly accepts terms of use by using Bugzilla.
   - Contributions from Committers do not require Bugzilla attachments.

2. Is the code a “Significant Contribution” requiring IP due diligence?
   - Guidelines:
     - Rule of thumb is > 100 lines of code **not** from a Committer
     - Third-party licensed code
     - Code with potential cryptography
     - Bug fixes and minor enhancements are excluded from the “significant” label, but the **committer** must review the code for appropriateness prior to check-in: technical correctness, no legal or copyright references, no profanity, no cryptography, etc. See the committer guidelines for more information.
Code submission and IP Due Diligence

3. If the submission is a “significant contribution”, the following steps must occur:
   a. PMC must approve the technical suitability of the code for the project.
   b. Committer must complete the Contribution Questionnaire. (see portal)
   c. EMO must perform IP due diligence and approve the contribution.

4. Committer commits the code to the project’s CVS repository
   - **Step 3 (if required) must be completed prior to check-in.**
   - Appropriate legal documentation must be included
     - about.html
     - EPL notice
     - Source code copyright and licenses
     - Additional licenses for third-party content
5. Tracking

- Each **Project Lead** must track **ALL** contributions from non-committers and for “significant contributions”.

- This traceability is public and is available on the projects website.


- Using Bugzilla for everything allows mostly automatic generation of the project log. You must use the “contributed” keyword on the bugzilla for this to work.
All Code Must be approved by your PMC

Next Step: What is the origin of the code?

See Figures 1 through 8 below


So you want to run a project in Eclipse? | © 2008 by Doug Gaff; made available under the EPL v1.0
Despite appearances, it’s not really that difficult

- The process…
  - provides full traceability on Eclipse code.
  - ensures that code can carry the EPL label.
  - gives companies confidence in their commercial products built upon Eclipse.
- Using Bugzilla for all submissions automates the traceability and should simplify this process for developers and contributors.
Parallel IP Process

- For Incubation Phase only!
- Provide parallel development while IP review is in progress.
Phase 2: Incubation

- Skills
  - Provisioning and Infrastructure
  - IP Process
  - Community Building
  - Using Bugzilla
  - Development Process
  - Release Review
If you build it…

- …they (might) come
  - Community participation isn’t guaranteed.

- Good tools = more users
- Good frameworks = more adopters
- *The architecture must encourage collaboration*

- Contribution requires active *evangelism* and *solicitation*.
- Contribution *costs companies money*!
- Don’t be afraid to *leverage commercial partnerships*.

- Selling point: those who contribute get to decide the direction! This is a recruiting statement, not an excluding statement
Company Recruiting

- Unlike other open source communities, Eclipse isn’t powered as much by “labor of love”.
- Getting companies to work on projects is critical to a project success.
- Companies must have strategic alignment with open source.
- Facets
  - Leverage commercial partnerships.
  - Talk to your competitors!
  - Evangelize your technology.
Stages of Eclipse Participation/Adoption

- **User**
  - *Silent*: Download (or purchase) and use Eclipse technology.
  - *Contributing*: File bugs and patches against the technology.

- **Add-in Provider**
  - *Silent*: Build a commercial product using Eclipse frameworks.
  - *Contributing*: File bugs and patches against the frameworks.

- **Beginning Contributor**
  - Staff project development with one or more engineers (committers).

- **Strategic Contributor**
  - Make Eclipse a core part of a product line.
  - Staff one or more projects with several engineers.

*Open Source is free as in “free speech” not “free beer” … need to get involved in order to get what you want*
Why Make Eclipse Strategic?

- Influence and control technical direction
  - Drive open standards that benefit the industry.
  - Evolve projects in a way that is most beneficial to your commercial implementation.
- Decide what is commodity
  - Strategically choose when to push proprietary technology into the open.
  - Community lives longer than individual efforts: integration vs. differentiation
  - Use as competitive weapon.
- Mitigate commercial risk
  - Becoming a Strategic Member increasing influence over the overall direction of the community and Foundation.
- “Support the movement”
  - Open source software has forever changed the software industry.
  - Eclipse has found a balance between free software and commercial software.
Making Eclipse Strategic for Your Company

- Goal: keep your project alive, staffed, and relevant

- "Marketing"
  - Internal project visibility – over-communicate what you are doing
  - Promote the ecosystem
  - Senior management / executive sponsorship

- Project Management
  - Align commercial and open source schedules
  - Establish commercial adoption mechanics
  - Define customer support processes
Commercial Adoption Mechanics

- Planning Cycles
  - Commercial release cycles should be near Eclipse cycles.
    - Gives time for adoption and final commercial testing.
  - Early planning required for features needed in open source.
  - Continuous commercial integration and testing required.
  - Architectural changes must be considered very early to meet API freeze deadlines.
  - Roadmaps must be aligned between open source and commercial releases.
Commercial Adoption Mechanics

- Development and CM
  - Commercial Eclipse releases should follow a Milestone development model similar to Eclipse.
  - Milestone dates between commercial and open source should not overlap.
  - “Continuous adoption” should be employed
    - Each M-Build pulled into commercial CM.
    - I-Builds are used for more frequent integration points.
  - Avoid “hacking the platform” for a release, except in very specific cases
    - Commercial release timing vs. maintenance releases in open source.
    - Patches not yet accepted in open source projects.
  - Some developers may work in both Commercial CM tool and CVS (Eclipse) at the same time.
  - Bugs are often cross-listed in Commercial Bug Tracker and Bugzilla (Eclipse) at the same time.
Customer Support

- *Problem:* customer defect escalations where the defect must be fixed in open source.
- **Worst case solution**
  - Patch open source package
  - Might create compatibility issues for “install into” customer scenarios.
- **Better solution**
  - Generate an official point fix in open source, e.g. x.y.z.1.
  - Much easier when you have committers on a project, e.g. CDT, TM, DD
  - Difficult / Impossible with some projects, e.g. Eclipse Platform
- **Best solution**
  - Fix in open source and adopt at next scheduled Eclipse train release, e.g. x.y.1
  - For projects where there are no committers, supplying a patch does not guaranteed that it will be adopted.
  - Sometimes open source must be patched in commercial delivery.
Phase 2: Incubation

- Skills
  - Provisioning and Infrastructure
  - IP Process
  - Community Building
  - Using Bugzilla
  - Development Process
  - Release Review
Bugzilla Workflow

www.eclipse.org/projects/dev_process/bugzilla-use.php
So you want to run a project in Eclipse? © 2008 by Doug Gaff; made available under the EPL v1.0
Bugzilla - Some common practices

- By social convention, users don’t change the committer owned fields.
- Components often use a general inbox, e.g. dd.general-inbox@eclipse.org – setup at provisioning time
- You can set “watches” on email address like this general inbox.
- Bug Triage
  - Validation
  - Prioritization
    - P1 - stop ship fix, need immediate attention
    - P2 - must fix before the release, but can make progress without fix
    - P3 - should/would like to fix for the next release
    - P4 - would be nice, but not critical, can ship without fixing
  - Development
  - Verification
Find a specific bug by entering words that describe it. Bugzilla will search bug descriptions and comments for those words and return a list of matching bugs sorted by relevance.

For example, if the bug you are looking for is a browser crash when you go to a secure web site with an embedded Flash animation, you might search for "crash secure SSL flash".

Status: Open
Product: All
Words: 

Actions: Home | New | Search | Find | Reports | My Requests | My Votes | Preferences | Log out doug.gaff@windriver.com | Terms of Use
Saved Searches: My Bugs | DSDP (old DD) | eclipsecon down | Wind River Bugs
Add ▼ the named tag ▼ to bugs ▼ Commit
<table>
<thead>
<tr>
<th>ID</th>
<th>Sev</th>
<th>Pri</th>
<th>OS</th>
<th>Assignee</th>
<th>Status</th>
<th>Resolution</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>105433</td>
<td>nor</td>
<td>P3</td>
<td>All</td>
<td><a href="mailto:doug.gaff@windriver.com">doug.gaff@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Container for Device Debugging Documents</td>
</tr>
<tr>
<td>120888</td>
<td>nor</td>
<td>P3</td>
<td>All</td>
<td><a href="mailto:doug.gaff@windriver.com">doug.gaff@windriver.com</a></td>
<td>NEW</td>
<td></td>
<td>Platform Debug View Improvements For Device Debugging</td>
</tr>
<tr>
<td>137391</td>
<td>enh</td>
<td>P3</td>
<td>All</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Create a Memory View Rendering with Traditional Look &amp; Feel</td>
</tr>
<tr>
<td>146659</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Traditional memory rendering: A few bug fixes and one feature addition</td>
</tr>
<tr>
<td>146667</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Traditional memory rendering: How should this rendering be deployed? The current implementation may be problematic in the future.</td>
</tr>
<tr>
<td>146670</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>CLOS</td>
<td>FIXE</td>
<td>Traditional memory rendering: Editing memory when memory is highlighted does not behave as one would expect</td>
</tr>
<tr>
<td>146679</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Traditional memory rendering: When adding a second traditional rendering to the same memory drawing object, double clicking on a cell should highlight that cell</td>
</tr>
<tr>
<td>146683</td>
<td>min</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Traditional memory rendering: Ctrl+G for goto address does nothing</td>
</tr>
<tr>
<td>146684</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>ASSI</td>
<td></td>
<td>Traditional memory rendering: Default endianness should be retrieved from debugger</td>
</tr>
<tr>
<td>149092</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Discussion for the Breakpoints View Technology Subgroup</td>
</tr>
<tr>
<td>152624</td>
<td>enh</td>
<td>P3</td>
<td>All</td>
<td><a href="mailto:doug.gaff@windriver.com">doug.gaff@windriver.com</a></td>
<td>NEW</td>
<td></td>
<td>Make DSF work smoothly in a java 1.4 environment</td>
</tr>
<tr>
<td>153944</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:pawel.piech@windriver.com">pawel.piech@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>DsFQuery should use Generics</td>
</tr>
<tr>
<td>153947</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:pawel.piech@windriver.com">pawel.piech@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Add traceability to DSF runnables</td>
</tr>
<tr>
<td>153952</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:dd.general-inbox@eclipse.org">dd.general-inbox@eclipse.org</a></td>
<td>RESO</td>
<td>DUPL</td>
<td>Remove references to Riverbed in the code</td>
</tr>
<tr>
<td>153959</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:dd.general-inbox@eclipse.org">dd.general-inbox@eclipse.org</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Make the org.eclipse.dd plugin run in a 1.4 environment</td>
</tr>
<tr>
<td>154273</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:pawel.piech@windriver.com">pawel.piech@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Need to create an annotation to help identify methods that must be called on display objects (exemple: show a label beside the “Go To Address” text field)</td>
</tr>
<tr>
<td>154346</td>
<td>min</td>
<td>P3</td>
<td>Linu</td>
<td><a href="mailto:pawel.piech@windriver.com">pawel.piech@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>The DSF plug-ins miss the provider name</td>
</tr>
<tr>
<td>157530</td>
<td>enh</td>
<td>P3</td>
<td>All</td>
<td><a href="mailto:pawel.piech@windriver.com">pawel.piech@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Added simple initial MI Register Service and Adapter</td>
</tr>
<tr>
<td>158553</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>vertical scroll bar thumb control does not scroll the window</td>
</tr>
<tr>
<td>158554</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>Able to scroll beyond the addressable range</td>
</tr>
<tr>
<td>158557</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:ted.williams@windriver.com">ted.williams@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>“Go To Address” area grows after being subsequently invoked</td>
</tr>
<tr>
<td>158591</td>
<td>enh</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:doug.gaff@windriver.com">doug.gaff@windriver.com</a></td>
<td>RESO</td>
<td>FIXE</td>
<td>show a label beside the “Go To Address” text field</td>
</tr>
<tr>
<td>158592</td>
<td>nor</td>
<td>P3</td>
<td>Wind</td>
<td><a href="mailto:doug.gaff@windriver.com">doug.gaff@windriver.com</a></td>
<td>RESO</td>
<td>DUPL</td>
<td>Memory window always shows Big Endian</td>
</tr>
</tbody>
</table>
Tabular Reports
Progress over Time

So you want to run a project in Eclipse? | © 2008 by Doug Gaff; made available under the EPL v1.0
Phase 2: Incubation

- Skills
  - Provisioning and Infrastructure
  - IP Process
  - Community Building
  - Using Bugzilla
  - Development Process
  - Release Review
Principle #1 – Open Source Rules of Engagement

- **Open**
  - Anyone can participate, including competitors.
  - Equal opportunity, consistent rules.

- **Transparency**
  - Discussions, meetings, minutes, plans, etc. must all be public.
  - Wiki, bugzilla, website, mailing lists, newsgroups.

- **Meritocracy**
  - The more you contribute the more responsibility you will earn.
  - Committers are voted in by other committers.

www.eclipse.org/projects/dev_process/development_process.php
Principle #2 – Eclipse Ecosystem

- Eclipse Foundation must
  - “cultivate an ecosystem of complementary products, capabilities, and services.”

- So Projects must
  - Communicate *project plans* in a timely, open, and transparent manner.
  - Create *quality frameworks* sufficient for commercial grade products.
  - Ship *exemplary tools* that highlight the frameworks and solve real world problems.
  - Participate in the annual Eclipse *roadmap process*.

- Maintaining quality is critical to the success of the Ecosystem
Principle #3 – Three Communities

1. Committer / Contributor
2. User
3. Adopter

- Introduced at the beginning.
- Diversity
  - #1: More than one company doing contributions
  - #2: Users applying the technology in different ways
  - #3: Multiple companies building products on top
- Focus your energy here and it will pay huge dividends in your project.

A project cannot exit the Incubation Phase without demonstrable evidence of these three communities!
Principle #4 – Clear and Concise

- The Eclipse Development Process is designed to be clear and concise.
- Get projects up to speed and running efficiently as quickly as possible.

- The Development Process seeks to balance project freedom and autonomy with the collective qualities of the entire Eclipse community.

- The Development Process is designed to evolve in all aspects with the guidance of the community.

- Norms for project behavior are part of the collective knowledge of the Eclipse community. The Development Process is not the all-inclusive check list.
Project Structure (new!)

- Nesting allowed, 2 levels typical (but not required)

- Projects contain
  - Committers
  - Code and Releases
  - IP Records
  - Community Awareness
Project Organization (new!)

1. **Committers**
   - An operating project has a self-managing set of committers
   - Committers cannot be imposed by a parent project

2. **Code and Releases**
   - An operating project is the smallest unit of infrastructure
     - Source Repository
     - Single Unix Group
     - Single Bugzilla Component
   - Container projects do not have file infrastructure
   - Projects at any level can make a release – roll-up of sub-projects
Project Organization (new!)

3. **IP Records**
   - Any project at any level can receive IP clearance on contributions

4. **Community Awareness**
   - Projects are the level of communication
     - They can have their own website, mailing lists, newsgroups, etc.
     - OR they can be part of the parent project’s communications.
   - Parent projects should reference communications channels of their sub-projects
   - Parent projects with an incubating subproject must indicate that they are incubating a sub-project on their own website.

5. **Scope**
   - The Top-Level Project has a board-approved charter.
   - Sub-projects inherit the charter of the top-level project.
Project Organization (new!)

6. Leaders
   - Top-Level Projects
     - Managed by a Project Management Committee (PMC)
     - PMC Lead(s) is the Top-Level Project Leader(s)
   - Sub-projects
     - Managed by one or more Project Leaders
   - Elections
     - Initial PMC / Project Leaders – elected during Creation Review
     - Subsequent Project Leaders nominated by Project Committers
     - EMO approves all Leaders.
     - Board approves PMC Lead (when they approve the top-level project creation)
Project Organization (new!)

7. Committers and Contributors

- Project Leaders lead a Development Team
  - Contributors – code contributions, fixes, tests, docs
  - Committers – above + write access to CVS/SVN
- New committers must be elected using the Portal
  - Nomination by an existing committer
  - Demonstrable track record of contribution
- Election Process
  - For a one-week period, votes are collected.
  - +1 if the committer knows the nominee and thinks they’re good
  - 0 if they don’t
  - -1 if they disagree with the nomination (and why)
- Committer is elected: at least three +1 votes and no -1 votes
Project Organization (new!)

8. Councils
   - Requirements (RC)
     - Eclipse Roadmap
     - Themes and Priorities
   - Planning (PC)
     - Release Train, e.g. Callisto, Europa, Ganymede
     - Cross-project planning, architecture, integration, etc.
   - Architecture (AC)
     - Eclipse Platform Architecture
     - Project Mentoring
     - Cross-project technical ideas and guidance
       - Package namespace rules
       - Avoid duplicating technology
Roadmap Process

- Prepared annually
  - Themes and Priorities from RC
  - Project Plans from Projects

- Project Plan
  - EMO-defined file format
  - Framework and tool changes
  - Consistent with Themes and Priorities
  - Cross-project dependencies
  - Addresses requirements for ecosystem and membership
  - Advances project in functionality, quality, and performance
Releases

- **Builds**
  - Nightly (N) – Nyyymmddtttt
  - Integration (I) – Iyyymmddtttt
  - Maintenance (M) – Myyymmddtttt
  - Stable (S)
    - Milestone – S-3.0M1-yyymddtttt
    - Release Candidate (RC) – S-3.0RC1-yyymmdtttt
  - Release (R) – R– 3.0-yyymmddtttt

- **Version numbers**
  - x.0 – some breaking API changes
  - x.y – some compatible API changes (upward but not backward compatible)
  - x.y.z – maintenance release, bug fixes only, no new features, compatible
  - x.y.z.p – nightly build qualifier, sometime used for point fixes
Releases – Typical Eclipse Train Dates

- M1: mid-Sept
- M2: mid-Oct
- M3: mid-Nov
- M4: mid-Jan
- M5: end-Feb
- M6 – API Freeze: mid-Apr
- M7 – Feature Freeze: mid-May
- Multiple RC’s: mid-May to mid-June
- GA: June 30

- [wiki.eclipse.org/Ganymede_Simultaneous_Release](http://wiki.eclipse.org/Ganymede_Simultaneous_Release)
Grievance Handling

- Member -> Project Leader, …, PMC, EMO, Board
- All grievances are open, transparent, and public

- Possible member concerns about a project
  - Out of Scope
  - Inconsistent with Purposes (roadmap)
  - Dysfunctional – my personal favorite
  - Contributor Appeal
  - Invalid Veto
Development
- project-status-infrastructure.php
- notifying-membership.php
- new-committer.php
- architecture-council.php

Phases
- pre-proposal-phase.php
- proposal-phase.php
- incubation-phase.php
  - eclipse-quality.php
  - parallel-ip-process.php
  - project-operations.php (words of wisdom)
  - index.php (how-tos, checklists, guidelines)

Phases (cont)
- mature-phase.php
- top-level-phase.php
- archived-phase.php

Reviews
- creation-review.php
  - project_provisioning_request.php
- graduation-review.php
- release-review.php

Note: these pages will be moving to the Wiki soon, but the links will continue to work.
Phase 2: Incubation

- Skills
  - Provisioning and Infrastructure
  - IP Process
  - Community Building
  - Using Bugzilla
  - Development Process
  - Release Review
Release Review

1. **Features**
   - Summarize major features
   - Reference New and Noteworthy documentation

2. **Non-Code Aspects**
   - User documentation, localization/externalization, examples, tutorials, articles, etc.

3. **API’s**
   - Provisional / Non-provisional

4. **Architectural Issues**
   - How the project is an extensible framework (Adopters)

5. **Tool Usability**
   - How the project supplies free tools (Users)
Release Review (cont)

6. **End-of-Life**
   - Any deprecated API’s or Features

7. **Bugzilla**
   - Stats – open/closed/deferred, etc. Breakdown of priorities.

8. **Standards**
   - Does this project use or comply with any standards?

9. **UI Usability**
   - Conformance to Eclipse UI Guidelines
   - (Currently being revised by the UI best practices group.)

10. **Schedule**
    - Original schedule plus any changes made along the way
Release Review (cont)

11. **Communities**
   - Summary progress and presence of the 3 communities

12. **IP Issues**
   - About files, use licenses in place
   - Statement of adherence to IP policy
   - Project Log complete and in-place

13. **IP Issues – Speak Up Now!**
   - Question to Members: Has this project infringed on your IP rights?

14. **Project Plan**
   - Cover any draft plan for the next release (if available)
Release Review (cont)

- Process is the same as Creation Review
- Docuware is in PDF and is archival quality
- Phone call is only for questions
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Phase 3: Mature Project

- Time for a Graduation Review
- All that stuff you learned in incubation still applies!

Graduation Review

- Requirements
  - Working code base with extensible frameworks and exemplary tools.
  - Three active communities
    - An active framework user (adopter) community.
    - An active tool user community.
    - An active multi-organization committer/contributor/developer community.
  - Full transparency
  - Mastery of the Eclipse Development Process

- Schedule the review and the process are similar to Creation Review.

- After you pass the review, the incubation logo goes away.
What does this phase look like?

- (Graduation Review Requirements)
- Like any other mature software product
  - Quality Focus
  - Predictable Schedule
  - Detailed Plan
- You still do Release Reviews for major releases
  - 1.1, 1.2, 2.0, etc.
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Phase 4: Top-Level Project

- When you have enough technology to create your own set of sub-projects.
- Sometimes reorganization of existing projects.
- Historically, some Member Companies started with Top-Level Project proposals.
- You can have this goal in mind, but you can no longer start here.
- Time for a Promotion Review

We have so much stuff!
Promotion Review

- Similar to a Graduation Review

- Demonstrate
  - A collection of related technologies
  - A new (typically vertical) space in Eclipse

- Examples
  - DSDP – started as a top-level to deal with Mobile and Embedded
  - Runtime (RT) – reorganization of several run time technologies
  - Modeling – reorganization and additions
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Continuation / Termination

- Sometimes projects go stale.

- Triggers
  - Change in sponsoring company strategy
  - Loss or inactivity of majority of committers
  - Inactive project lead
  - The code just isn’t going anywhere
  - The technology is not longer relevant

- Can be initiated by the mentors, proposed project lead, EMO, and the PMC.

- Can also be a withdrawal of a proposal by the submitter.
Continuation Review

- Reason: Company strategy change
  - Find a new company to sponsor (“Project Reboot”)
  - Archive

- Reason: Committer Inactivity
  - Demonstrate increased staffing plan from an existing company
  - Find a new company to sponsor (“Project Reboot”)
  - Archive

- Reason: Inactive Project Lead
  - Find a new project lead
  - Archive

- Reason: Lack of Code / Technical Relevance
  - Archive
Termination Review

- Purpose is to give the community a final chance to resurrect the proposal if sufficient interest and staffing exist.

- The choice is either continuation or termination.

“My mistake, it’s dead.”
Archived Project

- Project leaders: summary of goals, accomplishments, and remaining work
- Mailing lists, newsgroups, CVS/SVN, website stored in archive
- Mailing lists and newsgroups turned off
- CVS repository removed
- Last download only retained
- Project website replaced with a single landing page
- Project website moved to archive page
  - www.eclipse.org/technology/archived.php

- Can be reopened with a Creation Review

www.eclipse.org/projects/dev_process/archived-phase.php
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Good Eclipse project leaders

- Communicative
- Technical
- Organized
- Passionate
- Humble – running a project is about making mistakes and learning from them, accepting feedback, and dealing with conflict among committers

Most important: willingness to embrace the benefits and practices of open source

Ideal candidates
- Technical leads/architects with a bias for pragmatic processes
- Very technical managers who can get their hands dirty
- (not me)
Bad Eclipse project leaders

- (in addition to the opposite of the previous slides)
- **Too much work** on their plate.
- **Time imbalance** between commercial responsibilities and open source responsibilities (commercial *always* wins).
  - This is hard to see when you get started.
  - Try logging your time to see what your commercial / Eclipse split actually is over time.
- **Lack of willingness** to understand of open source development and processes and *change their behavior* to fit.
  - A mindset change is almost always required.
Embracing Eclipse

- Eclipse open source development doesn’t have to be a religion.
- It’s ok to make money on software.
- But there is always a balance between what you contribute and what you retain as IP.
- **Communication** is the most important skill.
Contents

- Successful Eclipse Projects - what do they look like?
- Three Eclipse Communities
- Starting and Running a Project
  - Phase 0: Declaration
  - Phase 1: Proposal
  - Phase 2: Incubation
  - Phase 3: Mature
  - Phase 4: Top-Level
  - Continuation / Termination / Archived
- Good Project Leadership
- Successful Eclipse Projects - challenges you will face along the way
Challenges – the beginning

- Gathering the community is slow!
- Building technology from scratch is slow!
- Initial committers from sponsoring company aren’t allocated to the project!
  - Didn’t the companies know what they’re getting into?
- Why won’t anyone volunteer engineering time?
  - Everyone wants to talk about requirements.
Challenges – the process

- Steep learning curve
- Transparency is hard
  - Announce, plan, and document everything in public!
- Diversity is harder
  - Coopetition
  - Getting engineering contributions
- IP Due Diligence – wow!
Challenges – committers

- They don’t work for you (usually)
  - Leadership by influence, not by control
- Turnover
  - It’s going to happen sooner than you think
- Disruptive behavior
- Architectural disagreements
- Working in a vacuum
  - Designs
  - Communication
- NIH (not invented here) syndrome
  - Look for duplicative technology
Challenges – contribution and adoption

- Competing strategic needs
  - Different contributing companies need different things
- From your own company
  - Don’t give that away!
  - Our commercial version is better! (“Not Invented Here” again)
  - “Strategic Membership” doesn’t translate to 8+ engineers
  - Keeping the project relevant to your company – internal evangelism
  - Your company changes its mind (sometimes frequently)
- Your competitors contribute before you do.
Challenges – over time

- So many users, so many questions – the price of success
- Committers/companies are losing interest
- Industry is changing
- Technology has a shelf life
- “Old projects never die, they just fade away”

You will make mistakes (remember humility?)
Be ready to correct and adapt
Yes, it’s worth it!

- Cool, smart people
- Cool, versatile technology
- Cooperation like you’ve never experienced it before
- Everyone is on a peer relationship with everyone else
- People do things because they want to do things

- It’s good for your company.
  - Stop building “me too” technology.
  - Share the burden, spread out the risk.
  - Increase the industry state of the art.

- And yes…it’s good for your career.
Questions?