Eclipse Tools PMC

Status Update

March 10, 2008

Author: John Duimovich
With Tools PMC Project Leaders

Introduction

The Tools PMC is one of the original Top Level Projects within the Eclipse community. The Tools Project was created to help bootstrap the Eclipse consortium and the initial formation contained no actual projects. The Tools project charter was written with the goal to enable the PMC to go out and attract “gold standard” projects to be part of the Tools Project. The intent would be that these projects would showcase the Eclipse Platform and it’s integration capabilities, help develop the Eclipse Ecosystem and enable wider participation by the growing Eclipse membership. As part of this bootstrapping effort, the Tools project was also a testbed to prototype some of the initial Eclipse processes including project creation reviews, project graduation and migration to other projects and even project reboots.

The project has grown from this modest start to include ten (12) projects to date, spanning language IDEs, compilers, common development frameworks and development automation.

During the lifetime of the Tools Project, there have also been two projects that have grown into their own top level projects - the Eclipse Modeling Project and the Test and Performance Top Level Projects.

This report was written at the request of the Eclipse Board as a mechanism to increase board awareness on what’s happening in the projects. This includes the heath and vitality of the projects, compliance with eclipse processes but primarily to raise issues where the board could assist these projects to become successful. As such, this report does not contain deep technical review of the projects. For that information, the reader should refer to the respective project’s websites which can be found at (www.eclipse.org/tools).

Project Scope Review

The Eclipse Tools Project charter contains the following mission and scope declaration.

Mission
The mission of Eclipse Tools Project is to foster the creation of a wide variety of exemplary, extensible tools for the Eclipse Platform. The Eclipse Tools Project provides a focal point for diverse tool builders to ensure the creation of best of breed tools for the platform, consistent with the Purposes of the Eclipse Foundation. The Eclipse Tools Project provides a place where a single point for interaction and coordination for all tools developers can occur which will minimize overlap and duplication, ensure maximum sharing and creation of common components and promote seamless interoperation between the diverse types of developed tools. The Eclipse Tools Project will promote the Eclipse vision and attempt to foster Projects that set the "gold standard" for tools implementers. The Eclipse Tools Project will use its experience in developing tools for the Eclipse Platform, to provide technical requirements and feedback to the Eclipse PMC.
Scope
The Eclipse Tools Project encompasses the tools being built for the Eclipse Platform. The main focus of these tools will be development tools such as computer programming language tools (compilers, editors, debuggers), performance tools, and test tools.
The set of tools being built by the Eclipse Tools Project will grow over time. Initially, we expect tools like a C/C++ IDE, as well as tools for existing languages that don't have existing IDEs (Lisp, Scheme). In addition, we expect that a wide range of debuggers for programming languages will leverage the Eclipse debug frameworks.

As the name suggests, the Tools project scope is primarily driven by the creation of developer focused tools. The current projects hosted in Tools include language IDEs, Compilers and varying developer targeted projects.

- Language IDE's (CDT, PTP, AJDT, PDT, COBOL and Hibatchi)
- Language Compilers - Aspect/J
- Graphical Tools and Frameworks –VE, GEF
- Common Components – Orbit
- Development Automation – Buckminster

For the most part, these projects all fit well within the definition of the current charter, each one creating developer focused tools and/or frameworks. The one exception is Orbit which can be described as more of a meta-project since it does not contains major development but exists primarily as a packaging project for third party open source components for use by other Eclipse projects. This was only partially stated as a goal at the start of the project, ("ensure maximum sharing and creation of common components") but the mechanism of using a container project for the inclusion of third party components is an obvious way to do this, even if not explicitly stated in the charter.

One of the stated goals of the project was to encourage more sharing between projects where components are common. This was driven by the assumption that much of the technology in software like programming language IDEs would have significant similarities. There has been some success in this as projects like CDT and PTP share significant amounts of there implementations. The rest of the language IDEs have less sharing of technology possibly due to the fundamental differences in their respective languages.

Charter Updates

The scope of the project to include test and performance tools is no longer necessary with the creation of the Eclipse TPTP Project.

The charter also contains an appendix with a list (which is out of date) of projects within the scope of the project and a roadmap. These are obsolete and as charter updates require board approval, this is too heavyweight to maintain and update. This section should be removed in a charter update.

The Tools Project is in full compliance with the Eclipse Roadmap and does not envision any changes required in that roadmap specifically for the Tools project.

Release Status

The Tools project works a little differently than the other Top Level Projects at eclipse.org. The projects within Tools operate as primarily independent projects as such, do not create a single release plan nor do the projects synchronize milestones, or releases unless they join the Eclipse train releases. Of the 12 projects, 5 are participating in the Ganymede simultaneous release. The remaining projects have scheduled releases now through 2008.

Project	**Ganymede Train ?**	**Major Features , Issues, Comments**
AJDT	No	Issue - no plan updates
Aspect/J	No	March 08 1.6 release – Java 6 support, memory usage improvements
Buckminster	Yes	
COBOL	No	Summer 2008 – Update release
CDT	Yes	http://wiki.eclipse.org/CDT/planning/5.0
GEF	Yes	Zest integration from Mylyn
Hibatchi	Incubator	Awaiting initial code contribution.
Mylyn	Yes	1. Frameworks & APIs: Tasks, Context, Team, Monitor, headless use

		2. UI: Tasks List, Task Editor, Task-focused UI 3. Connectors: Bugzilla (reference implementation), Trac (committer supported), JIRA (community supported)
Orbit	Yes[1]	Components updated as required
PDT	No	PDT 1.02 January 08
PTP	No	Release Review Scheduled
VE	No	Project in transition

[1] Orbit is not officially listed in the Ganymede release train list of projects but all new third-party Ganymede components must consume them via Orbit to prevent duplication.

Project Assessments

The Tools project is made up of essentially separate projects, so this report is divided into individual sections which describe the status of the projects. Each section contains a brief description of the project, a PMC summary and a "self assessment" of the project status.

The project assessments below were created with the participation of the project leaders with additional evaluation by the PMC. The format for assesments are compiled using the template requested by the Eclipse Foundation for this report. The assessment may contain additional summary from the PMC if deemed appropriate.

Eclipse Foundation Assesment Template

Self-assessment of the performance of the project under the following headings (inspired by the Three Communities section of the Development Process):

- Performance as an Eclipse open source project, with specific self assessments on the following:
  - i. Openness
  - ii. Transparency
  - iii. Meritocracy
  - iv. Diversity
  - v. Compliance with the Purposes (e.g. are they successfully "…supplying frameworks and exemplary, extensible tools.."?)

- End user community and adoption. E.g. are there lots of downloads, bugs, contributors, newsgroup postings ? Note that I believe that while the absolute numbers are interesting, the more important data-point is the project's assessment of how those numbers compare to their own expectations for the project.
- Commercial community and adoption. E.g. is the technology from the project showing up in products

AspectJ, AJDT

AspectJ is a seamless aspect-oriented extension to the Java programming language that enables clean modularization of crosscutting concerns, such as error checking and handling, synchronization, performance optimizations, monitoring, logging, debugging support, and multi-object protocols.

The AspectJ Development Tools (AJDT) project provides Eclipse based tool support for Aspect-Oriented Programming with AspectJ. Our goal is to deliver a user experience that is consistent with the Java Development Tools (JDT) when working with AspectJ projects and resources.

AspectJ and AJDT are closely coupled and have overlapping committer populations. The official project lead is from IBM however the majority of the active development work is from SpringSource. The activity on the project is moderate. The project has had resource issues over the past few years.

The interesting part of these projects is their respective adoption in the Spring Framework, a lightweight container framework which is becoming popular in the Java enterprise space. For this reason, the PMC expects interest in this project to increase from all the Enterprise Java vendors. There also appears to be a growing community making queries about the technology which indicates growth for the project and consumption of the technology.

The self assessment for the AJDT and AspectJ projects is below.

- Performance as an Eclipse open source project

i. Openness/ Transparency

The lightweight processes of bugzilla and the mailing lists make it very easy to stay open and transparent. Maintaining up to date plans upon the website is always tricky - everything gets out of date so quickly, but bugzilla is always kept up to date as the publically visible 'TODO' list for both projects. Recently on the AspectJ project we have opened bugzilla voting to help prioritize work items - and this is publically viewable of course.

ii. Meritocracy

Unfortunately the bar for contributions to both projects is quite high, for AspectJ in particular because it is so complex. However, we get a few contributions each release and I try to integrate what I can or support those on the list looking to get to grips with the code. The company I work for is recruiting new talent to work specifically on these projects, but there will be no shortcuts to getting promoted to committer: they will have to earn commit rights after contributing good patches.

iii. Diversity

There is a lack of diversity on this project. The key developer works for SpringSource which means, of course, the issues relating to Springs usage of AspectJ/AJDT get extra focus. However the voting that is available on both projects means that any serious problems the community has will get addressed as high priority issues. There are some contributors who's focus is research in AOP. As such they tend to dig into corner cases or problems that don't really affect project adoption or success however does create a more solid foundation.

iv. Compliance with the Purposes

Both AspectJ and AJDT do as intended. AspectJ is now on the Eclipse 3.3 compiler which makes it a Java6 compiler that should go to release candidate shortly.

- End user community and adoption
  Mailing list is about as active as I'd like at the moment and it isn't always me answering questions, some of the long term users chip-in with solutions to help out new users. Bug reports are 'healthy' although we do have a couple of long standing problems that regularly crop up that need addressing but will take significant development effort.
  We have a wide range of users, from those in commercial environments using RAD that need support on Eclipse 3.2 through to those on the bleeding edge running Eclipse 3.4M5.
  Throughout 2007 the projects were perhaps a little starved of resource, but that has changed now and we are actively looking to recruit new development effort for them - if we can sort out some of the long standing development issues with the new resource, I think we can really increase adoption.

- Commercial community and adoption.
  We have consumers in various industrial settings making use of AspectJ, usually consuming it through AJDT.
  The loadtime weaving capability of AspectJ is packaged with the spring framework (open source) and the forthcoming SpringSource Tool Suite bundles AJDT (and will hopefully have an integrated ajdt/mylyn bridge shortly). From the queries we see on the springframework forums, users are making use of the technology.

CDT

The CDT (C/C++ Development Tools) Project is working towards providing a fully functional C and C++ Integrated Development Environment (IDE) for the Eclipse platform. Our focus is development on Linux for deployment on Linux, but we are interested in participation from others who would like to extend our work in other directions (e.g. a Windows client, targeting Unix(R) or embedded platforms, wizards for developing applications that use particular library or database or messaging APIs, or extension to other languages).

The CDT is also being used as a core platform for Fortran (PTP) and for Ada in the newly incubated Hibatchi project.

CDT is most likely the best run project in the Tools PMC and probably in all of Eclipse itself. The project has developed an active community, and exemplifies the open source development model. The project lead (and a PMC member) has been able to walk the fine line between open source development and cooperation with competitors. Commercial adoption indicates that every embedded RTOS provider is using CDT which on it's own is unremarkable, but when coupled with the fact that the project lead is employed by a competitor, demonstrates how a project can be run fairly and with an open process. This is clearly indicated by the diverse set of active committers from a wide range of companies.

The self assessment for CDT is here.

- Performance as an Eclipse open source project

  i. Openness

  Pretty much all discussion happens on the cdt-dev mailing list, bugzilla reports, and our biweekly conference call. All are open to anyone.

  ii. Transparency

  Often, our feature work is done by individuals who do not necessarily provide regular status reports. We have moved our conference calls from monthly to biweekly to help promote more frequent updates from contributors. All work is done based on bugzilla entries allowing anyone to track progress and ask questions.

  iii. Meritocracy

  Our current standard is about a dozen good patches before accepting nominations for new committers. In the past, at times of need, we have loosened that qualification to help build up the committer community.

  iv. Diversity

  On average, we have 1-2 contributors from each vendor contributing to the CDT and we have committer representation from 9 or 10 different organizations. And we get contributions from many more. I think we can claim to be the most diverse project at Eclipse.

  v. Compliance with the Purposes

  The CDT has been mainly focused on providing frameworks by which vendors can build integrations. We provide exemplary integrations with the open source gnu tool chain as well as other platforms such as IBM's and Microsoft's compiler suite.

- End user community and adoption
  Downloads are huge. The last time we were able to accurately measure a release we had around 400,000 which is well beyond expectations. We have steady bug report traffic and currently have over 1200 bugs open which is a fairly steady number (i.e. they are raising them as fast as we are fixing them). We are seeing an increasing number of patches being attached to incoming bug reports, a great sign of increasing contributions. And the newsgroup is fairly steady with standard user questions.

- Commercial community and adoption.
  The CDT has always seen great commercial adoption. The vast majority of embedded operating system and platform vendors redistribute the CDT with integrations with their build and debug tool chains. The CDT is a part of most commercial Linux distributions. In fact it appeared there as a value-add before appearing in the standard free distros. It is also distributed by Intel as a part of their compiler suite for Linux. The most disappointing area here has been the commercial adoption for Windows development. This is still a large market but the dominance of Visual Studio has resulted in poor prospects at a commercial CDT based competitor.

COBOL

The COBOL IDE project will build a fully functional COBOL Integrated Development Environment (IDE) for the Eclipse platform. The focus is COBOL application development on Windows/Solaris/Linux for deployment on each platform.

The COBOL project is one of the more difficult projects in Tools. The core team is in Japan, and very isolated from the community. Some of this may be due to language barrier of the core team and the fact that the homogeneous developer population, in that all the developers are from one company.

The COBOL project was very responsive when asked to comment on their self assessment if their status.

There has been interest from the community in broader participation in the COBOL project and it is the intention of the PMC to add these new members to the project in an attempt to invigorate the development process and induce more openness. There is currently an effort on the mailing list to identify new committers for the project.

The self assessment for COBOL is here.

- Performance as an Eclipse open source project

i. Openness
   Members are all in-company so I can't say the project is open.

ii. Transparency
Project status have not been issued so transparency is relatively-low.

iii. Meritocracy
Power of development is relatively-low because of short of man-power.

iv. Diversity
Members are all in-company so I can't say the project is very diverse

v. Compliance with the Purposes
The project is tasked with creating an IDE for COBOL. It has delivered an IDE but there is a lack of community around the project and new members have found it difficult to be involved.

- End user community and adoption
  Low participation by the end user community (34 mails to Mailing List for the last year). The number of downloads is unknown, but product version that is based on opensource COBOL IDE, have been selling more than 1,000 in Japanese market. I personally often get asked about COBOL IDE or the project from Japanese customers. They all well-know the eclipse tools project and our COBOL project is also in it.

- Commercial community and adoption.
  Yes. In our commercial product "NetCOBOL", we strongly promote COBOL IDE on eclipse. Some serious bugs are fixed, performance is dramatically improved, and some features are added, remote build/debug, support Oracle pre-compiler, support eclipse 3.2, and so on (of course stable). Sold in a commercial version that is based on the opensource COBOL IDE.
  It is unknown if any other commercial adoption has taken place with this technology.

GEF

The Graphical Editor Framework (GEF) allows developers to take an existing application model and easily create a rich graphical editor. GEF allows a developer to quickly map any existing model to a graphical editing environment. The graphical environment is the SWT-based drawing plugin "draw2d" (which is part of the overall "GEF" component). The developer can take advantage of the many common operations provided in GEF and/or extend them for the specific domain. GEF is suitable for creating a wide variety of applications, including: flow builders, GUI builders, UML diagram editors (such as work-flow and class modeling diagrams), and even WYSIWYG text editors like HTML. GEF does not assume that you must build one of these applications and is application domain neutral.

The GEF project is fairly mature and has not had extensive development in the past few years. It could be considered in maintainence mode however this year, there has been some new development. There was the addition of Zest into GEF and a revamp of the Palette, both farily large features however the rest of the functionality is pretty much stable

In terms of openness, the project does run all operations via newsgroup, bugzilla and mailing list. In the past, the project was IBM dominated and this can somewhat hinder efforts to join the project however in this past cycle, a new non IBM committer was added as the owner of the new Zest component.

The PMC sees the project running fairly well however as with other projects, the level of resources on the project is a concern.

The self assessment for GEF is here.

- Performance as an Eclipse open source project

  i. Openness
     Very open, everything in the newsgroup and bugzilla
  ii. Transparency
     Good
  iii. Meritocracy
     Excellent, a new non IBM committer was added to take care of Zest.
  iv. Diversity
     Average – there are three part time IBM committers, one part time independent committer for Zest.
  v. Compliance with the Purposes
     Very good, project is creating the technology it was chartered to create.
- End user community and adoption
  There is a broad set of consumers of GEF including other projects within Eclipse (GMF).

- Commercial community and adoption.
  IBM is a significant consumer of this technology having 67 products across all 5 software group brands consuming this technol

Orbit

Orbit provides a repository of bundled versions of third party libraries that are approved for use in one or more Eclipse projects. The repository will maintain old versions of such libraries to facilitate rebuilding historical output. It will also clearly indicate the status of the library (i.e., the approved scope of use). The repository will be structured such that the contained bundles are easily obtained and added to a developer's workspace or target platform.

The PMC sees no issues to discuss with respect to Orbits openness. It is a collaboration of many projects within the Eclipse ecosystem and has significant participation from many of these projects.

The self assessment for Orbit is here.

- Performance as an Eclipse open source project

  i. Openness
  Orbit readily receives contributions and committers from a wide range of projects within Eclipse.

  ii. Transparency
  All aspects of Orbit take place in mailing lists, newsgroups, and public conference calls for which agendas and minutes are posted in the wiki.

  iii. Meritocracy
  All Orbit committers are committers on other Eclipse projects and have proven that they have the time and energy to put into managing third party code in Orbit.

  iv. Diversity
  The Orbit committers come from a wide range of companies (including independents) and Eclipse projects.

  v. Compliance with the Purposes
  Orbit is a little different in that its sole mission is to bundle third party code as a service to the rest of the Eclipse community. In this task Orbit is very successful and is saving large amounts of time and making the Eclipse deliverables more coherent.

- End user community and adoption
  Most mature Eclipse projects consume some number of Orbit bundles.

- Commercial community and adoption.
  This particular question does not really apply to the Orbit project however pretty much every Eclipse-based product ships some Orbit bundles.

PDT

The PDT project is working towards providing a PHP Development Tools framework for the Eclipse platform. This project will encompass all development components necessary to develop PHP and will facilitate extensibility. It will leverage the existing Web Tools Project in providing developers with PHP capabilities.

The PDT is a project which is struggling with growing its committer membership. The majority of the committers in the project are from one company (Zend) and as such, there is a tendency to run the project as an internal project. This has manifested itself in such ways as committer additions without the necessary contributions proving the committers bona fides. The PMC has been able to contain this but needs to be vigilant.

The PDT has some issues in it's transparency due to it's single company majority and needs to increase it's efforts to communicate plans and design decisions to the community. The recent discussions on the dev mailing lists indicate that the planning process is becoming more transparent than in the past.

The project does have a large base of users, PHP being a very popular language and significant numbers of bugs and questions come from this community. It is difficult to try to grow the community from this group is that they are not Java programmers and as such cannot help in the development of the too themselves.

The self assessment for PDT is here.

- Performance as an Eclipse open source project

i. Openness

PDT has a large set of API in the form of extensions which allows the usage of PDTdata model, AST, debugger and many other infrastructure functionality that can be extended.

ii. Transparency

PDT project keeps its transparency and open communication with its community of users using various channels:

1. Bugzilla to accumulate bugs and enhancement requests.
2. [pdt-dev] mailing list that serves as an open channel between the developers and different contributors and adopters.
3. PDT web site at eclipse.org: contains the project plan, schedule and latest news.

iii. Meritocracy

All PDT committers (except me, Gadi, Keren and Robert Goodman) are contributing code to the CVS on regular basis, solving bugs and adding new features or improving the PDT infrastructure.

Gadi Goldbarg is the QA team leader and although he is not developing a lot of code, he is developing some Unit testing and is one of the most active persons in this project which is reflected in both the Bugzilla and the pdt-dev mailing list answering questions and providing support.

Keren Stern is in charge of the documentation and Help documents and is constantly working to improve the PDT documentation and is committing it to the CVS.

Robert Goodman has been a very active committer at the begging of the project about 2 years ago and has helped the rest of the committers on their first steps in the eclipse eco-system. In the last 16 months he is no longer active in the PDT project and is using his time to contribute to the webtools ATP project.

iv. Diversity

The PDT project is has 2 major goals – to create an extendable PHP development toolkit that has all the basic functionalities of an IDE (syntax coloring, code assist, annotations, problems view and so on) and to provide a comprehensive infrastructure for developing new advanced features on top of the PDT. This includes (but is not limited to):

- Abstract Syntax Tree (AST) for PHP & AST rewrite - that can be used for enhanced code formatting and source generation (getter/setter, override/implements) .
- PHP Model that includes indexing (of occurrences) that is used for code inspection (outline view, type hierarchy) and can be used for Refactoring
- Debugger infrastructure – that was initially used to integrate the ZendDebugger and later on the XDebug (also available in PDT 1.0)

v. Compliance with the Purposes

Absolutely yes, I believe that we are successfully supplying a comprehensive PHP framework that can be extended by adopters to provide a large set of enhanced functionality as described in section (iv) above

- End user community and adoption

PDT is ranked 2nd on the most popular Eclipse projects for a very long time, I think that speaks for its community involvement and massive adoption. The PHP community is very active by nature (PHP being an open source) and has adopted the PDT project and are sending us bugs, feature requests, having discussions in the IRC and sending us feedbacks and questions through the [pdt-dev] mailing list (that is used not only by the PDT developers but also by the community).

- Commercial community and adoption.

There have been few adopters of the PDT, they are described at: http://www.eclipse.org/pdt/whois.php

- Project Zero (IBM REST development framework)
- APDT (PHP Aspect support)
- Zend studio for Eclipse (commercial product based on PDT)
- Smarty for PDT (Google code)

PTP

The Parallel Tools Platform (PTP) project aims to produce an open-source, robust, commercial quality platform that provides a highly integrated development environment specifically designed for parallel and high performance computing application development. The project will provide a standardized and portable IDE that supports a large number of parallel and high performance computing architectures and runtime systems, a scalable parallel debugger, and a range of parallel language development tools that will improve the productivity of scientific application developers.

PTP is a working in an area that is becoming more important in the computer industry. As processors hit the megahertz wall, the trend in the silicon industry is to drive performance via parallelism specifically multicore chips and specialized co-processors so it is somewhat surprising that there is not more interest in this project. It is possible that the focus being "classical HPC" (Fortran and C/C+) had made the project seem less relevant to the needs of Java programmers. The PMC will be haiving discussions

The PTP project is having a release review on the 12th of March for it's pending release on March 31st.

The self assessment for PTP is here.

- Performance as an Eclipse open source project

i. Openness

All information relating to the ongoing development of PTP is maintained publicly. Contributions are also welcomed from any interested parties.

ii. Transparency
All PTP developer discussions, meetings, workshops, and planning activities are conducted using public forums

iii. Meritocracy
Participation from committers is always sought for design & planning decisions, new committers, and other major issues.

iv. Diversity
PTP strives to encourage participation from as diverse a community as possible.

v. Compliance with the Purposes
PTP developers aim to comply with the Purposes, although resource issues may impact on successfully achieving all aims. Contributers are always encouraged to provide generic frameworks where specific solutions would suffice. Emphasis is also placed on the extensibility of tools.

- End user community and adoption

There is only a small user community at this stage. The main issues impacting adoption are:

- o Community resistance to IDE adoption
- o Eclipse development model is sufficiently different from existing practice (e.g. remote development)
- o No/poor support for Fortran
- o Scalability issues with CDT

This will hopefully improve after the next major release of PTP and improvements to CDT in Callisto.

- Commercial community and adoption.
  IBM is adopting PTP technology. Unknown if additional vendors adopting the technology.

VE

The Eclipse Visual Editor project is a framework for creating GUI builders for Eclipse. It will include reference implementations of Swing/JFC and SWT GUI builders, but intends to be useful for creating GUI builders for other languages such as C/C++ and alternate widget sets, including those that are not supported under Java.

The VE project is under somewhat of a transition. The project evolution had stagnated due to resource issues and VE was essentially in maintenance mode for a period of time. The project lead is still IBM but most of the effort since Europa and forward has been non-IBM committers. The new committers who have done the majority of development has been by Philippe Ombredanne and Steve Robenalt. These are two dedicated inviduals who want to take broaden support for visual programming and improve the GUI builder for SWT to make it more competitive.

The self assessment for VE is here.

- Performance as an Eclipse open source project

i. Openness

Relatively open. Successful demonstration of new committers taking over key aspects of the project.

ii. Transparency

Good. Newgroups, bugs and mailing lists all used to manage the project.

iii. Meritocracy

Project is in transition and decisions are being made based on resources and willingness to work on features.

iv. Diversity

Poor but improving. Project was exclusively IBM but has now attracted new developers. Project still needs more developers.

v. Compliance with the Purposes

The VE project is in full compliance with it's goals.

- End user community and adoption
  TBD

- Commercial community and adoption.
  TBD

Buckminster

Buckminster is a set of frameworks and tools for automating build, assemble & deploy (BA&D) development processes in complex or distributed component-based development. Buckminster allows development organizations to define fine-grained "production lines" for the various stages in software production - unit testing, integration testing, field deployment, staged migration, etc. - and to automate the execution of corresponding processes.

The self assessment for Buckminster is here.

- Performance as an Eclipse open source project

i. Openness

Our process is open in all respects. Our code resides in the tools SVN at Eclipse.org, We maintain two Eclipse.org news groups, one for end users and one for developers. The latter is also mirrored as a mailing list. Ideas and proposals for improvements are posted there for input from the community. Bugzilla is also used as a tool for this.

Our documentation is maintained http://wiki.eclipse.org/Buckminster_Project and anyone is welcome to improve on it.

Downloads, update site, archived site, and headless product are always available on our download page http://www.eclipse.org/buckminster/downloads.html

Most links can be found at our webpage http://www.eclipse.org/buckminster

ii. Transparency

Our aim to solve real life problems is our major driving force when moving forward. Anyone in the community that has an opinion can influence the future of Buckminster. We are all ears and we take every posting to our lists very seriously. At present our main objective is to reach stability, improve usability, and give good support to our current user base in preparation for our final 1.0 release. This release is scheduled to be simultaneously with the Ganymede release. Whatever we plan in addition to that will be published on our wiki. I intend to make more use of my blog http://thhal.blogspot.com/ too to ventilate ideas but time has been short the last couple of weeks.

More info can be found here: http://www.eclipse.org/buckminster and of course here: http://www.eclipse.org/projects/project_summary.php?projectid=tools.buckminster

The Active/Inactive status is a bit misleading in my opinion. Michal recently made a significant contribution with a test framework. Stefan is continuously working on the documentation on our wiki and he was also the driving force between our Webinar http://live.eclipse.org/node/427. Henrik is currently very busy with the Spaces project (which has close ties to Buckminster). Mich takes on a heavy burden with social networking. Not everything is about committing code.

iii. Meritocracy

Aside from the initial committers, we added two committers from the Maven camp (Exist (former Devzus)) and four committers from Cloudsmith Inc. All were added based on merits. We have also kicked of two committers that left the projects.

Carlos and Erle (Devzus) were added because they submitted a plug-in for Buckminster that uses the Maven embedder technology. Rather then using Buckminsters home brewed Maven support, we will gain access to the true Maven logic. Unfortunately, this has been blocked so far due to IP related issues. The Maven embedder is not yet formally released so Carlos and Erle has remained passive since they were added.

Filip, Karel, and Michal, all made significant code contributions to Buckminster that they continuously maintain.
Stefan has written 80% of our documentation and is also driving our test effort.

iv. Diversity

We have accepted contributions in the form of patches from various Buckminster users. We have committers from several different places in the world (Sweden, Czech Republic, Germany, USA) but by now we have managed to employ all except two (the Exist guys) in the same company, Cloudsmith Inc.

Buckminster also take a very active part in other projects such as Ganymatic and Spaces and we have made some recent contributions to ECF. Our download mechanism are moving over to use ECF filetransfer.

v. Compliance with the Purposes

Ganymatic and Spaces can serve as good examples of this. Several projects, both within Eclipse (STP for instance) and in the external community is using Buckminster to orchestrate their build process.

Buckminster is of course completely built using Buckminster. We also provide a simple demo app and we plan to do much more work in this area after EclipseCon. One project that I find particulary interesting in

this respect is the proposed EEP project
http://www.eclipse.org/proposals/eep and I already have a running
example of Buckminster building the Sudoku Web-app.

- End user community and adoption

We recently left incubation. Since that happened, we perceive a much higher interest for Buckminster. Our webinar had about 70 active listeners and currently over 500 people has downloaded and viewed it (since February 6). The number of bugzillas has gone up and we receive an increased amount of postings on our mailing lists.

My own perception is that it is starting to really take off now. My worries are definitely not stagnation, rather that our resources will be insufficient to deal with all future support and feature requests. I don't see that as a big problem though. If the interest increases, so will the patch rate. That in turn will make it easier to recruit new committers based on merits.

- Commercial community and adoption.

This project is in it's early stages and as such, there has not been any significant adoption at this point nor do the project leads really expect that to happen with this product yet. We do know that a number of commercial projects are using Buckminster to orchestrate their builds but Buckminster isn't part of the final product (neither is Ant, Maven, or CruiseControl).

Well, there is one exception. The Cloudsmith Inc. website www.cloudsmith.com, is utilizing a lot of Buckminster technology behind the scenes

Mylyn

Mylyn is a Task-Focused Interface for Eclipse that reduces information overload and makes multi-tasking easy. It does this by making tasks a first class part of Eclipse, and integrating rich and offline editing for repositories such as Bugzilla, Trac, and JIRA. Once your tasks are integrated, Mylyn monitors your work activity to identify information relevant to the task-at-hand, and uses this task context to focus the Eclipse UI on the interesting information, hide the uninteresting, and automatically find what's related. This puts the information you need to get work done at your fingertips and improves productivity by reducing searching, scrolling, and navigation. By making task context explicit Mylyn also facilitates multitasking, planning, reusing past efforts, and sharing expertise.

Mylyn is one of the better run projects in the Tools PMC. It has a very active project lead who is making very serious efforts at building a community around Mylyn. The project is very serious about being inclusive and providing users a forum for participating in its development process. With the mantra that the "user is always right", Mylyn sets a very high bar for itself. Mylyn has developed communications guidelines to ensure that all ideas are welcome and evaluated fairly and to ensure a professional level interaction is maintained on the mailing lists, newsgroups and bug reports. Mylyn's bread and butter is user interactions so the balance is to focus on users.

The PMC points out these committer guidelines as exemplary and other projects should consider adopting these types of guidelines, especially if they want to grow their developer and user communities.

The self assessment for Buckminster is here.

- Performance as an Eclipse open source project

i. Openness

The Mylyn development process very open and inclusive. We facilitate contributors getting up to speed by sharing task contexts via Bugzilla (147 bugs with activity from the last 30 days had contexts). Mylyn's current best indicator of openness is the IP Log, which demonstrates dozens of community patches applied with each quarterly release: http://www.eclipse.org/mylyn/doc/mylyn-iplog.csv

ii. Transparency

All significant project communication happens via Bugzilla. 311 enhancement requests and 147 bugs have been active in the past 30 days. For design discussions and online prioritization of bugs we have a weekly call that's one hour long and open to all interested. We also rely on the newsgroup, mylyn-dev and mylyn-integrators mailing lists.

iii. Meritocracy

The project has a clear meritocracy and process for becoming a committer that follows Eclipse conventions. Three of the current 4 active committers have become committers through this process (i.e. all but the project lead). There are current three active contributors working on a part time basis who are close to earning commit rights.

iv. Diversity

The project has high diversity due to the large number of community contributions. The committer diversity is currently lower due to the fact that one of the community contributors was hired by Tasktop Technologies, which means that 3/4 committers are employed by Tasktop technologies. The project is actively seeking adopters and integrators to become committers. It has suffered from some unprofessionalism in its community channels that has driven commercial developers away but has been attempting to improve the channels.

v. Compliance with the Purposes

We are delivering exemplary tools. Our focus for the past year has shifted to improving the Mylyn frameworks in order to support a growing ecosystem of extensions.

- End user community and adoption
  Growing steadily, in particular since the June 2007 release of Mylyn 2.0.

- Commercial community and adoption.

Tasktop Technologies provides the resources for the framework and references implementations. Atlassian has been contributing to the JIRA connector. CollabNet has been contributing via patches from their connectors. CodeGear has contributed the XPlanner connector. Rally Software has been contributing via patches for their connector. SpringSource has been contributing via their extensions to the Task-Focused Interface. The number of Mylyn extensions is continuing to grow at a fairly rapid pace: http://wiki.eclipse.org/index.php/Mylyn_Extensions

Summary

The Tools PMC has learned a few lessons in writing this report the first one being, allow more time for the report writing phase. That said, the annual report is a good idea and the PMC will encourage its projects to keep the "self assessment" part of the scorecard updated regularly but at a minimum annually as input to this written report.

Recommendations

1. IP Process – Some projects have indicated that they would like to use the incubator IP process for some components of the project. Currently you need to create a new Incubator project to be allowed to do so. In tools, this muddles the water as most of the projects are separate and combining all the incubators into a single project is unwieldy. The board should consider allowing components within a project to be incubated as longs as these components provide clearly marked downloads to indicate the IP has not been cleared.
2. The Eclipse Development process continues to evolve and improve but it has been difficult to keep up with the changes and projects often find themselves behind or in non-compliance with some new features. Even well run projects like CDT and Mylyn seem to have incomplete or incorrect meta-data from the portal page. The reminder tool however is quite handy.