Eclipse project update

by the Eclipse project PMC – March 11th 2008

This document provides an update of the developments accomplished within the Eclipse Project, how they align with the Ganymede simultaneous release, and beyond.

1. High-level review

The Eclipse project is composed of five sub-projects: Platform, Equinox, JDT, PDE and the Eclipse Project Incubator We will present our current development strategy for the upcoming releases, our release plan and our technical progress so far.

1.1. Strategy

The Eclipse project has delivered seven releases in a row, on an annual basis. Each of these releases needed to both innovate and establish a stable platform. Today, the Eclipse platform is the basis for thousands of solutions in the broader Eclipse community. Hence, stability is becoming the predominant requirement, over innovation for our clients. Therefore, the Eclipse PMC made the decision to split streams into a 3.x stream and a 4.0 stream. The 3.x stream will be dedicated to consolidation, where the focus is on stability, performance, scalability, and controlled innovation; it will keep delivering on an annual basis, and Eclipse 3.4 will be the Eclipse Project's contribution to the Ganymede effort. The 4.0 stream is a longer term effort, intended to capture the future of Eclipse, focus more on pure innovation, and provide a catalyst for increasing participation in the Eclipse Project. The Eclipse PMC is currently defining what a 4.0 plan might look like, and will be presenting ideas at EclipseCon'08.

In the 3.x stream, we intend to preserve our ability to run on Java 1.4 VMs, with optional functionalities requiring Java5 or Java6 VMs (e.g. JSR-199, JSR-269, etc.). We will continue to leverage newer versions of our reference platforms, and provide access to new platforms (e.g. Windows WPF). For Mac OS X, Eclipse 3.4 will still use the Carbon API. In a subsequent release (3.5), we are considering deprecating Carbon and using the Cocoa API instead.

The Eclipse project provides one incubator subproject to host new initiatives, until they can be graduated into the mainstream (Platform, Equinox, JDT or PDE sub-projects). We intend to define additional incubators as needed, and in particular one dedicated to the Eclipse 4.0 work, named "e4 incubator". Note that an incubating component may not graduate in the 3.x stream. For instance, the 4.0 work in the e4 incubator will only graduate once 4.0 becomes the mainstream. Reciprocally, code developed in an incubator may graduate sooner than 4.0, we had several instances of this occurring during 3.4 (e.g. API tooling, provisioning).

Finally, Eclipse has been evolving a runtime community for quite some time with Equinox, RCP, RAP, eRCP, ECF, EMF and others. New projects like Swordfish, Riena and EclipseLink are also emerging in the runtime community at Eclipse. Eclipse runtime efforts have suffered from the view that Eclipse is for tooling. The Eclipse RT top-level project has been proposed to focus on fostering, promoting and housing such runtime work at Eclipse. The Equinox Framework and OSGi provide a common component model used to build runtimes in Eclipse and therefore will move to the Eclipse RT project. Equinox will likely move to the Eclipse RT project after the Ganymede release to minimize disruption.

1.2. Release plan

To ensure the planning process is transparent and open to the entire Eclipse community, we post plans in an embryonic form and revise them throughout the release cycle. The current detailed plan is publicly available at
http://www.eclipse.org/eclipse/development/eclipse_project_plan_3_4.html.

The first part of the plan deals with the important matters of release deliverables, release milestones, target operating environments, and release-to-release compatibility. Eclipse 3.4 will be compatible with Eclipse 3.3 (and, hence, with 3.2, 3.1 and 3.0).

The plan is organized in sections based on areas of focus. For the 3.4 release, the major work areas are:

- **Platforms**: This work is focused on ensuring that Eclipse takes full advantage of the capabilities of the underlying technologies that it is based on, be they operating systems, window systems, Java or others.
- **Consumability**: This work will make it easier for users to get Eclipse, install it on their systems, and configure it for their use. It will also enhance the error handling and reporting mechanisms to make it easier to service Eclipse in the field. Finally, it will improve the scalability and performance of Eclipse, to provide a better experience for users working with many plug-ins and large data sets.
- **Reliability**: As the basis for the entire Eclipse eco-system, the Eclipse SDK must be robust, flexible and secure. This work will address those issues by providing API for missing or currently internal functionality, and focusing on the issues that effect the stability of the platform.
- **The Future**: Eclipse is well-established as the cross-platform IDE of choice, but it has become much more than that. The extensive and diverse range of applications that are being built on the Eclipse code base, and the constantly changing capabilities of the underlying systems on which it runs, are driving us to push the limits of our technology in almost every dimension. This work area marks the start of a new, multi-year focus on innovation, to ensure that the Eclipse SDK continues to be a vibrant, powerful, dynamic basis for our community's use.

Currently all the planned items are marked as committed or proposed. None are marked as deferred until next release.

1.3. Technical progress

The highlights in our 3.4 plan are: SWT 64-bit, SWT WPF port, Linux Bidi, Mac Carbon Internalization & accessibility, product level configurability, provisioning, serviceability, API tooling, security, concurrent compiler and Eclipse 4.0 planning. Here is a high-level review of our progress in some of these areas.

- **SWT 64-bit** : Prior to Eclipse 3.4, 64-bit support was available on Linux GTK only. Support for 64-bit Windows was added using a similar implementation strategy. The result is that SWT is customized for the platform and architecture. For example, Eclipse running on 32-bit Windows is completely unaffected in terms of performance and code size, while Eclipse on 64-bit Windows is a first class citizen on that architecture.

- **Linux BiDi** : Mirroring is a technique that is used to minimize code changes for custom controls when dealing with translation into right-to-left languages like Arabic and Hebrew. In order to support right-to-left locales, mirroring moves the origin from the top left of a control to the top right. All graphics and widget operations become relative to this new point. Prior to Eclipse 3.4, mirroring was supported on Windows only. In 3.4, Eclipse applications can now be mirrored on Linux GTK.
- **Mac Internationalization and Accessibility** : In Eclipse 3.4, the keyboard input mechanism for the Mac and other platforms has been overhauled, giving Eclipse and applications built with SWT unparalleled native behavior. For example, custom controls such as StyledText respect native IME (Input Method Editor) colors and settings, edit text in-line and are indistinguishable from their native counterparts. For Eclipse 3.4, accessibility was implemented for the Mac, providing access to both native and custom controls to differently enabled individuals.
- **API Tooling** : In Eclipse 3.4, PDE has been augmented with API tooling. The integrated toolset assists developers in API maintenance by reporting API problems such as binary incompatibilities relative to a previous release, incorrect plug-in version numbers, missing or incorrect @since tags, and all illegal use of APIs between plug-ins. Quick fixes are provided to correct problems where possible. The tooling is also designed to be used in (and will eventually be integrated with) the automated build process to create API problem reports.
- **Provisioning (aka. p2)** : Eclipse 3.4 delivers a complete replacement for Update Manager based on the new Equinox provisioning platform (p2). For end users, p2 provides simplified workflows, improved download technology, and the ability to share software components across multiple Eclipse applications. A small stand-alone SWT-based installer allows for provisioning entire applications rather than just augmenting or updating an existing application. p2 provides backwards compatibility by supporting installation from update sites designed for Update Manager, and the Update Manager concept of features.
- **Equinox Transforms** : Eclipse 3.4 will deliver org.eclipse.equinox.transform* bundles to the Equinox component, that allow to provide transformations of bundle resources at the OSGi level. Various example transformers exist (XSLT, sed, replacement) that can be used to transform any resource in a bundle including but not limited to plugin.xml, MANIFEST.MF, class files, etc. These bundles provide a powerful tool for RCP and product developers to achieve customization of consumed bundles in whatever way they see fit without actually altering those bundles directly.

2. Self-assessment

In this section, we will assess our performance under the headings inspired by the Three Communities section of the Development Process.

2.1. Performance as an Eclipse open source project

2.1.1. Openness and Transparency

This year, Jeff McAffer (Equinox lead) left IBM, but he is still assuming his role on the Eclipse Project PMC; so this results in some diversity on our PMC. As a result of this, we have improved our processes to better separate internal IBM conversations from public

Eclipse-focused ones, and are beginning to use public channels more frequently for PMC level communication.

We also identified increased openness and transparency as being high-priority.

2.1.2. Meritocracy

SWT has voted commit rights to Uttaran Dutta from IBM India and Oleg Krasilnikov from Intel. Scott Kovatch, formerly of Apple, now at Adobe, is also a strong candidate for commit rights. The hold up is that he is in transition from Apple and doesn't have a new email address so the vote can't proceed. Equinox grew 13 new committers since November 2006 (5 outside IBM). Platform/UI grew 2 new committers (1 non-IBM still in progress).

However, we are not doing a good job at revoking commit rights from inactive committers. We need better guidance from the Eclipse Foundation on this.

2.1.3. Diversity

Diversity is a big concern in the Eclipse project. It was originally contributed by IBM and we haven't seen much improvement over time. We get more committers outside IBM, but the volume of contributions is largely IBM, with help from BEA. For instance, here are numbers for actual commits in 2007.

- Equinox: 89% IBM, 8% individuals, 1% ProsystSoftware, 1% compeopleAG
- JDT: 95% IBM, 4% BEA, 1% individuals
- PDE: 82% IBM, 17% individuals
- Platform: 98% IBM, 1% BEA, 1% QNXSoftwareSystemsCo.

The Eclipse 4.0 initiative is paving the way for others to join. One of the main objectives of the e4 initiative is to increase diversity in participation. It is important to realize that Eclipse 4.0 will only come to fruitition with the help and participation of our community. We already see some interest from the RAP team (Innoopract) in working together to construct the basis for Eclipse 4.0 as the "e4" initiative as an incubator in the Eclipse Project. We do expect that they will become committers on the platform.

2.1.4. Compliance with the Purposes

The Eclipse project is very careful at designing its API in the best possible way. In particular, extensibility is an obvious concern for an agnostic platform. It should be noted that even though JDT & PDE are part to the Eclipse SDK, their needs are considered with all others. All platforms enhancements must be useful to the community at large to make it into the next release, and are discussed in the open: our mantra there is that until you have several clients, you do not have an API.

2.2. End user community and adoption

There is no question that the adoption of the Eclipse project is huge. What will be interesting is to watch how much adoption we get in Eclipse 4.0 vs. Eclipse 3.x. This highly depends on how much we succeed at getting participation from multiple companies.

2.3. Commercial community and adoption

The Eclipse platform is the core of the Eclipse universe and is depended on by all other projects. For example, there have been more than 350 distinct products in IBM build on top of Eclipse technologies, and thousands of Eclipse-based solutions across the world. An interesting data point though is how quickly products adapt to newer versions of the platform. It is important that existing Eclipse-based solutions can quickly leverage newer versions of the platform as they become available, and thus gain new capabilities, increased performance, scalability and robustness. Still too often, some platform evolutions are causing some breakage even though these changes are strictly backward compatible according to our API compatibility charter. The truth is that some applications use internal interfaces, not official API, and thus become vulnerable to any change, even in a service release. Our 3.x/4.0 split should improve the situation. Also, our new API tooling, available in Ganymede, will flag non-API dependencies, and thus allow a risk assessment associated with any component.

3. Compliance with the roadmap

The Eclipse project is compliant with the 3.4 roadmap. This means our projected milestones will be honored both in term of schedule and content. We are currently completing milestone-6, which is our API freeze. We will then engage in a consolidation effort with a stronger performance focus.

4. Board Assistance

The Eclipse PMC identified the following areas where the board could help our project to be more effective.

4.1. Improving diversity

Most of the commits in the Eclipse projects are still performed by IBM. There are historical reasons for this, but there needs to an ongoing focus for getting more committers outside of IBM to participate. The Eclipse 4.0 effort is emphasizing this, but even in the 3.x stream, some help would be greatly appreciated. The Update Manager is currently at risk of having zero active committers. Even though the new provisioning functionality is replacing it, there are still many clients that depends on the current Update Manager. Also we would like to see a growing participation of OS manufacturers in our SWT ports. While we are seeing progress, we need to do better. This would allow Eclipse to run better and on a wider range of platforms.

4.2. Establishing Eclipse 4.0

Eclipse 4.0 is a big initiative, which can only succeed if many others engage. We need the board to help in attracting new committers to Eclipse 4.0, and to promote the technology for a wider adoption.

4.3. Java Compliance Testing

Currently the JDT subproject contributes a Java compiler, but the foundation has no license to the official Java Compatibility Kits from Sun Microsystems Inc. As a result, it is impossible to assess compatibility as part of normal open source development activity; and it is quite possible that some regressions would be introduced as a result of fixing bugs in the compiler. If the foundation did own a JCK license, then this issue would get solved, and the compiler development would be safer.