Eclipse RoadMap v5

Introduction

As required by the Eclipse Development Process, this document describes the 2010 Eclipse Roadmap.

There are three main sections to this document:

1. This Preamble provides some background on Eclipse and the Foundation, and identifies the strategic goals of Eclipse. It also provides a brief overview of the scope of future projects
2. The Themes and Priorities which has been developed by the Eclipse Councils.
3. The Platform Release Plan which has been developed by the Eclipse Planning Council.

The Roadmap is intended to be a living document which will see future iterations. This document is the fifth version of the Eclipse Roadmap, and is labeled as version 5.0. In order to preserve this document while the underlying information evolves, the pages have been frozen by copying them from their original project hosted locations.

The goal of the Roadmap is to provide the Eclipse ecosystem with guidance and visibility on the future directions of the Eclipse open source community. An important element in this visibility is that the Roadmap help the EMO and the Board of Directors in determining which projects will be accepted by Eclipse during the life of this revision of the Roadmap. In other words, new projects must be consistent with the Roadmap. This does not mean that every new project must be explicitly envisaged by the Roadmap. It does mean that new projects cannot be inconsistent with the stated directions of Eclipse. In particular, Eclipse expects that incubator projects created in the Technology PMC will cover areas not explicitly described in the Roadmap.

New in the version 5.0 iteration of the Road Map is a focus on projects that are part of the next planned release train code named "Helios" with a release date of June 2010. For example, project plan listings are plans for Helios from the projects that are part of that release train.

Background

As defined on our website, the role of the Foundation is:

> *Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. The Eclipse Foundation is a not-for-profit, member supported corporation that hosts the Eclipse projects and*

helps cultivate both an open source community and an ecosystem of complementary products and services.

As defined in our Bylaws the Purposes of the Eclipse Foundation are:

> *The Eclipse technology is a vendor-neutral, open development platform supplying frameworks and exemplary, extensible tools (the "Eclipse Platform"). Eclipse Platform tools are exemplary in that they verify the utility of the Eclipse frameworks, illustrate the appropriate use of those frameworks, and support the development and maintenance of the Eclipse Platform itself; Eclipse Platform tools are extensible in that their functionality is accessible via documented programmatic interfaces. The purpose of Eclipse Foundation Inc., (the "Eclipse Foundation"), is to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services.*

Strategic Goals

The following are the strategic goals of the Eclipse Foundation.

1. Establish Eclipse as a leading provider of open source runtime technologies. At least since 2004, Eclipse projects have been shipping innovative runtime technologies such as Equinox and the Rich Client Platform. The last several years have seen steady growth in runtime technologies at Eclipse. At the same time, there has been rapid growth in interest in OSGi, which is the standard upon which the Eclipse plug-in model is based. Moving forward, we expect to see rapid growth in both the projects building and the adoption of Eclipse runtime technologies.

2. Maintain global leadership in open source tools platforms. As an open development platform, Eclipse provides support for multiple operating environments and multiple programming languages. The goal of Eclipse is to define for the industry a development platform which is freely licensed, open source and provides support for the full breadth of the application lifecycle, in many disparate problem domains, across the development and deployment platforms of choice. In particular, as Rich Internet Application development becomes more mainstream, we anticipate new projects at Eclipse to address the needs of that community.

3. Create value for all its membership classes. The Eclipse Foundation serves many members whose primary interest in leveraging Eclipse technologies in commercial offerings such as products and services. The Eclipse Foundation will focus its energies to ensure that commercial opportunity exists within the Eclipse ecosystem. Look for continuous improvements to Eclipse Marketplace, and other initiatives for the benefits of members.

   Committers are also members of the Eclipse Foundation and are in many ways its

backbone. The Eclipse Foundation and its staff will continue to look for opportunities to continually improve services to its project community throughout 2010. Look for enhancements to our web, download, code management, build and other key components of project infrastructure in 2010.

4. Foster growth of the ecosystem, particularly in verticals. The creation of a large community of commercial and open source organizations which rely on and/or complement Eclipse technology has been a major factor in the success of Eclipse. Each time Eclipse technology is used in the development of a product, service or application the Eclipse community is strengthened. Our goal in 2010 is to focus our attention on the creation of industry working groups and new Eclipse projects which focus on particular industry segments such as mobile, automotive, insurance and finance.

5. Run a good ship. This Eclipse Foundation employs several staff and represents hundreds of stakeholders. It is important that the Foundation be a well run organization internally and externally.

6. Continue to grow a diversified revenue model. Reliance on a single source of revenue to fund the Foundation puts at greater risk of being negatively impacted by industry specific business cycles. It is a goal of the Eclipse Foundation to ensure revenue sources from multiple types of organizations, and seek other sources such as events and sponsorships.

7. Ensure adequate resources are invested in the core technology platform. Although Eclipse hosts well over 100 different projects, many depend on a core set of projects to continue to evolve and innovate. It is important the new entants be encouraged to participate in these core projects and be informed of the importance of and need for their support.

Future Directions

The goal of the Roadmap is to provide the Eclipse ecosystem with guidance and visibility on the future directions of the Eclipse open source community, and to involve the Eclipse membership in a dialog about those future directions. In that vein, this section discusses our current vision of the future as a set of future projects that expand the value of the ecosystem for all of its members.

The Themes and Priorities document prepared by the Requirements Council describes a number of requirements and focus areas for the existing Eclipse projects.

In addition to the Themes and Priorities requirements on existing projects in Helios, we envision future growth in Eclipse projects in the following major areas. These are areas in which we envision starting new projects in 2010-2011, not areas in which we envision having completed Eclipse-quality standards-based frameworks and tooling.

Eclipse 4.0

2010 will see the first major release of the Eclipse Platform since 2004. The major goals of this new release include:

- Making it easier to write plug-ins
- Allowing better control over the look of Eclipse based products
- Providing a uniform, pervasive platform across computing environments (Web / RIA, Desktop, Server, Cloud, Embedded)
- Increasing diversity of contributors to the platform
- Maintaining backward compatibility for API-clean clients

Cloud

Projects such as Amazons AWS tooling build on the Eclipse Platform give instant credibility to Eclipse as a important piece of the strategy for cloud tool providers. Eclipse has a role to play in the entire development lifecycle from development, deployment to testing and QA. In addition to tools, OSGi and Equinox will play an important role in the cloud. The ability to maintain configurations and deployments in large scale applications will be essential to scaling the cloud.

EclipseRT

EclipseRT will continue to grow and evolve. The recent Gemini and Virgo projects at Eclipse demonstrate the community is starting to associate Eclipse as a great place to do runtimes. The key uniter of the various runtime technologies at Eclipse continues to be OSGi as the kernel, but it is plausible for some other complementary technology to become an Eclipse project.

Web Development

Eclipse tools have historically had a very strong correlation with the Java language. However, with the rapid growth of Rich Internet Application languages and technologies such as JavaScript and Ajax, the Eclipse community must support the requirements of these developers as well. Eclipse will begin to invest in re-tooling the Eclipse platform with the needs of Web developers in mind.

The Roadmap Process

The process of creating the Eclipse Roadmap is described in the Eclipse Development Process. The key pieces are:

- The Councils propose a set of Themes and Priorities that realize the purposes and that respond to requirements elicited from the Strategic Developers, Strategic Consumers,

Sustaining Members, and other constituents of the ecosystem. The EMO ensures that the Planning Council and the Development teams have access to all requirements. Updates to the purposes are likely to require updates to the Roadmap and its associated themes and priorities; proposed Roadmap updates may also be motivated by new technologies or opportunities.

- The process of producing or updating the Roadmap is expected to be iterative. An initial set of Themes and Priorities may be infeasible to implement in the desired time frame; subsequent consideration may reveal new implementation alternatives or critical requirements that alter the team's perspective on priorities. The EMO orchestrates interaction among and within the three Councils to drive the Roadmap to convergence.

Eclipse 2010 Themes and Priorities

Theme Categorization

Eclipse themes are described in one of two categories.

- Active themes are those that are ongoing and changing. From time to time, some Active themes will become Persistent and Pervasive.
- Persistent and Pervasive themes are not time or release specific. Persistent and Pervasive themes are not only a signal of importance, but permanence.

Active Themes

Platform Support

While Eclipse has been very successful with Java developers on Windows systems, we endeavour to provide platform support for additional existing and upcoming platforms.

Windows XP, 7 and Beyond

Approximately 80% of Eclipse download requests are for the Windows OS. Windows 7 will be important as both a platform used by developers, as well as one to which the resultant applications and/or products will be deployed.

Given Eclipse's ongoing use in the development of enterprise software, support for use of the Eclipse platform is desirable across the popular Windows OS variants.

Linux

Linux continues to grow in market share as a platform for projects at all levels. We need to offer strong support for Linux on two dimensions:

- As a deployment platform for applications developed using Eclipse technology
- As a platform used by developers as their primary working environment

Mac OS X

Mac OS X is used in many development, open source and end user environments and is a very active community. Throughout recent years there has been a stable percentage of Mac platform downloads of Eclipse packages of about 5% of downloads. Eclipse needs to continue to provide some level of support for users of this platform.

Support for the latest Java(tm) versions

Eclipse endeavors to support next generations of the Java platform in a timely manner. The first stage of support is that end users can build applications that target the latest JDK versions. The next stage of support is that projects themselves are able to take advantage of the latest JDK changes.

Rich Client Platform (RCP)

RCP adoption has been strong by the ecosystem. The goal is for projects to support and use the Eclipse RCP as much as possible.

Aside from general use of RCP, there are two additional dimensions to this theme.

1. Enabling broader use of RCP on smart devices such as PDAs and enhancing the abilities of RCP to work in these environments.
2. Making RCP as easy as possible to use so that it's easier for application developers to adopt.

The Equinox project is an example of the focus on the OSGi component model within Eclipse. The Ecosystem requires Additional PDE enhancements to facilitate developing and deploying RCP-based applications, and for OSGi bundle manifest tooling.

Rich Internet Applications (RIA)

Flex, Ajax and other "Web 2.0" technology (see Wikipedia discussion on Web 2.0) has enabled the development of a new generation of web sites that provide a rich and user-friendly experience in a wide variety of applications. While the initial adopters of this technology have

been social web sites, it's adoption is increasingly seen in business applications such as CRM systems.

As developers shift from the development of traditional web sites to Web 2.0-style sites, the role of Eclipse as a development framework for these applications must be considered. In order to retain these developers, the Eclipse platform could provide strong support for developing applications that leverage Web 2.0 technologies such as Flex, Ajax and Web Services APIs.

Embedded Device Software

This theme describes additions to Eclipse to provide standardization and extensibility to enable embedded tools providers, real-time operating system providers, semiconductor vendors, and hardware developers to create embedded-specific capabilities on top of standard Eclipse projects such as the Platform, JDT, eRCP, CDT, and TPTP. These capabilities could include the following.

- Drive consistency in the workflow for embedded development tools and projects.
- Continue evolving the debug platform API's to support embedded debugging. This debug model will enable integration of debug engines from multiple vendors for debugging bare metal hardware, bringing up operating systems, and developing applications on single and multi-core hardware. This implementation will also enable vendors to integrate target simulation and emulation environments with Eclipse.
- Build a remote target launching, exploring, and management framework with extensible real-time operating system visibility. This framework will provide complex launching capabilities for deploying multiple target images to multiple devices.
- Enable C++ GUI application design, build, and deployment for mobile devices running any operating system. Also enable vendors to customize run-time libraries for their operating systems.
- Provide mobile Java application development support for J2ME mobile profiles, including extensible frameworks for devices and emulators and capabilities for application build and deployment, code obfuscation, code optimization, image signing, and localization.
- Complete the 1.0 release of the embedded Rich Client Platform.
- Enable mobile Linux application development, including design, development, debug, and deployment of cross-compiled applications.
- Provide embedded testing capabilities - monitoring, profiling, and unit testing.

Ease Of Use

The Eclipse components need to not only provide features that advanced users demand, but also be something that users find simple to use. The goal of this theme is to ensure that Eclipse-based products are simple to use for users with widely-varying backgrounds and skill sets performing a variety of tasks. Examples include:

- Provide Eclipse User Experience Guidelines to ensure consistency and usability (including Accessibility) across projects.
- Usability reviews and updates to new and existing user interfaces to streamline common processes and clarify concepts and terminology.
- Improving support for Cheat Sheets to assist users in performing tasks.
- User personas/roles to streamline the user interface to adapt to specific user needs.
- Continue improvements on the Java editor towards tolerating broken code.
- Enhanced user documentation, tutorials, white papers, demonstrations.

For example, if a user interface wizard provides a short path to performing a task, make sure that usability studies have identified the most common task performed by the target users.

Improving the "Out of the Box" Experience

All projects should consider improved packaging, installation and "out-of-the-box" experience to be a critical objective.

Through efforts such as the Eclipse Packaging Project, Eclipse should continue to expand out-of-the-box role-based offerings.

Maturity and Project Readiness

Every project has the capability to damage the reputation and brand of Eclipse if it is claimed to be "release" quality, but clearly is not. Quality refers not only to the reliable execution of typical use cases, but also to documentation, feature completeness, etc. Moreover, when new functionality or architecture replaces old between versions, it is important that features be maintained (or a plan for doing so made available) to not give the appearance of eliminating features.

Technology Trends

Existing and new Eclipse projects need to consider key technology trends in the market to ensure that the Eclipse platform continues to retain it's leadership as the framework and tool of choice for developers.

Cloud Computing

More and more applications are being built to handle bursts of high volume traffic and data by making use of "cloud computing" architectures. By deploying to the "cloud", application developers are more frequently building applications to run on servers where they have little control or knowledge of the underlying technology infrastructure. In many cases this also means relying on non traditional data formats, planning for redundency and fail-over and keeping interactions low-state.

Having tools and projects that support a tools ecosystem for Cloud Computing application developers would be an enormous benefit to Eclipse.

Multi-Core CPU

Due to power constraints, there is a trend towards multiple cores on a CPU instead of merely increasing the CPU frequency. Eclipse could enable developers to write multi-threaded programs to take advantage of the increasing miltiple cores. Moreover, Eclipse itself could be optimized where possible for running on multiple cores.

64-bit CPU

Diverse application software such as payroll, datawarehousing, and reporting now routinely manipulate large amounts of data that exceed 2GB. Using 64-bit CPUs enables these applications to manipulate large data in memory rather than having to write and read intermediate results to much slower disks.

The availability of 64-bit CPUs and matching 64-bit versions of supported OSs is growing, not only on the server but on the desktop. As these 64-bit environments become more popular and Eclipse technology-based server applications become more prevalent, Eclipse could be optimized to run within these environments and aide developers who building on, and/or targeting to, 64-bit solutions.

Scaling Up

This refers to the need for Eclipse to deal with development and deployment on a larger and more complex scale. Increasing complexities arise from:

- Large development teams distributed in different locations,
- Large source code bases, large amounts of data, multiple scripting and programming languages, complex build environments that have been developed incrementally over time the dynamic nature of new source code bases and their interaction with configuration management, and build environments involving many different tool chains and build rules.
- Large volumes of data

Possibilities:

- Performance improvements in memory footprint, user perceived response times, and start-up times as the complexity and number of projects, files, users, and plug-ins grow (10X-100X over the next two years). This is particularly important in client/server environments where a single Solaris, AIX, Linux or HP-UX server must support dozens of concurrent Eclipse users and where Eclipse competes mostly with command line tools.
- Improve support for and performance with Motif based window managers on Solaris (drag and drop, etc)

- Improve remote X window performance
- Improve performance when creating, loading, importing and closing projects with slow file systems (networks)
- All Eclipse projects could identify common use cases and publish performance benchmarks on every milestone.
- Ability to deal with extremely large projects and workspaces where there is a large number of developers working on different, and sometimes overlapping parts of the source tree simultaneously. This may include a more efficient way to manage multiple workspaces. Examples of large projects include Mozilla and Open Office.

Enterprise Ready

Large organisations have a need to support large numbers of users have and maintain similar Eclipse set-ups. This could involve various aspects of the system, eg. what Eclipse components are installed, what preferences and other values are set etc. On one level this could be a convenience thing so that this would enable central management to help developer workstations be up-to-date, on a different level some organizations might want a policy of strict control where the maintenance of the environment is also about enforcing a development policy and toolset, this would need more work in that it would require Eclipse internal policy management.

Provisioning

Ease of installation, deployment, of pre-canned packages (or products) while allowing easy discovery and mixing-in of additional functionality. This includes not only traditional Eclipse features and plugins but could also include the ability to pick from and install various runtimes, servers, libraries, or databases. I also could include not only the initial installation by one end-user, but could also include maintenance, remote installation or management of several installations, etc.

Ease of Deployment, Servicability & Managability of Large Scale Installations

- Zero-conf discovery of Update Site & Pref/Config Repositories
- Shared Hierarchical Configuration (First-class Preferences, Perspective Configurations, Team Share Preferences, Update Site Preferences, etc.)
- Simplified Update of the Platform Bits
- Plug-in Cross-Dependency Awareness / Version Incompatibility
- Improve the Eclipse project and workspace concept to allow overlapping environments
- Ability to fit into an existing environment of source files, build artifacts and version control repositories with minimal disruption to let developers complete a full edit-compile-debug cycle in the shortest possible time. This may include better support for multiple programming languages across language toolkits for improved usability. This would also include a more flexible project model.

Facilitated On-Boarding

Features to enable a developer to get started as part of a (new or existing) team. This could include:

1. Making sure that the person has the correct software set-up,
2. That the software settings are appropriate for the team and then finally (which falls outside perhaps of the above management) that the projects and the project content can be easily "bootstrapped" to the new workstation.

Example Story: Here the ultimate goal could be that once a "team manager" has been told the IP address of a new member's PC, he would have 10 minutes later a fully configured Eclipse workstation with all the project's Eclipse project and all related settings on his/her machine.

Design for Extensibility

Within the Eclipse community, many development projects are defining new development platforms on top of other Eclipse projects. Concrete examples include the Business Intelligence Reporting Tools, the Data Tools, and the Device Software Development Platform projects. It is recognized, however, that some function is not strictly required by the underlying projects but are important to enable other platforms to succeed. This theme also includes effort to assure platform integrity.

Some identified key requirements for this theme are:

- Robust API documentation
- API tools to detect use of internal interfaces
- Expose meaningful building block APIs
- Open the internal JDT (UI) interfaces to enable tools to seamlessly facilitate and interact with the JDT core and UI layers. For example the parsing and AST functionality.
- Enhance the Debug API to enable seamless debugging across mixed (Java+native) languages
- Provide a more flexible mechanism that can be used to debug non-Java programs. This is both in the debug model and presentation
- Provide for debugging a system comprised of multiple languages
- Enable task automation
    - Provide access to Eclipse APIs and resources from scripting languages
    - Provide the capability to record, edit, playback macros, representing a set of user interface actions.
- Provide a better experience for the co-existence of offerings from multiple vendors in a single Eclipse installation
- Permit offering brand/identity to show through (e.g. On the splash screen)
- Allow for license management of "products" (i.e. Aggregations of features)
- Integrated diagnostic capabilities - e.g. When a user encounters a problem, providing assistance on the where the problem originated, which product

- Loosen the strong file orientation by providing an abstraction layer of logical objects to allow one to extend Eclipse functionality tools working at a higher abstraction level.
- Authoring, deploying and managing components/features/etc.
    - Bolster OSGi Adoption (via authoring assistance, etc.)
    - Headless Execution
    - Server-side Runtime Infrastructure
    - Core & UI Split

Consistent Multi Programming Language Support

The original vision of Eclipse was to accelerate the creation of IDEs. There is a lot of work to do to make it simpler to create language-specific IDEs. Our vision is to:

- Make it easier to create language specific tools in a consistent way
- Enabling source files written in multiple languages within the same project.

Persistent & Pervasive Themes

Accessibility Compliance

Every project could support and make a statement on their accessibility compliance. In the U.S., this means Section 508 compliance; in the European Union, this is the Web Accessibility Initiative of the World Wide Web Consortium (W3C).

Internationalization & Localization

Every project could support both internationalization and localization:

- Internationalization (I18N)
- Each project could be able to work in an international environment, including support for operating in different locales and processing/displaying international data (dates, strings, etc.).
- Localization
- Each project could provide an environment that supports the localization of the technology (i.e. translation). This includes, but is not limited to, ensuring that strings are externalized for easy translation.

Where possible, projects should use an open and transparent process to create, maintain and deliver language packs translated into multiple languages in a timely manner. The primary languages to consider are: English, Simplified Chinese, Traditional Chinese, Japanese, French, German, Spanish.

Upgrade Path

Upward compatibility is a critical aspect of developer satisfaction and community growth. Developers need to be able to adopt the latest release of Eclipse technology without reworking their applications. Extensive rework incurred during a migration will lead to developer frustration and the possibility that they will evaluate and adopt other tools. Smooth upward migration is therefore a core Theme that all projects must consider.

This includes:

- Assuring release-to-release migration is supported (e.g., resources, workspaces, API, as appropriate).
- Assuring API compatibility release-to-release, including testing for upward compatibility
- Clear statements indicating which APIs are intended for internal use only (and are not guaranteed to be upward compatible)
- Providing tools that automate the migration process where possible

Eclipse Platform Release Plans for Helios

The following projects from the Helios release have created Project Plans that you can review at the links below.

- birt
- datatools
- dsdp.mtj
- dsdp.tm
- dsdp.tml
- eclipse
- eclipse.jdt
- eclipse.pde
- eclipse.platform
- modeling.amalgam
- modeling.emf
- modeling.emf.cdo
- modeling.emf.compare
- modeling.emf.emf
- modeling.emf.net4j
- modeling.emf.teneo
- modeling.emft.eef
- modeling.emft.mint
- modeling.emft.mtf
- modeling.emft.mwe
- modeling.gmf

- modeling.gmt.modisco
- modeling.m2m.atl
- modeling.m2m.qvt-oml
- modeling.m2t
- modeling.m2t.acceleo
- modeling.m2t.jet
- modeling.m2t.xpand
- modeling.mdt.ocl
- modeling.mdt.uml2
- modeling.mdt.xsd
- modeling.tmf.xtext
- rt.ecf
- rt.eclipselink
- rt.equinox
- rt.jetty
- rt.rap
- soa.swordfish
- stp
- stp.bpmnmodeler
- stp.sca
- technology.actf
- technology.dltk
- technology.egit
- technology.jgit
- technology.jwt
- technology.linux-distros
- technology.mat
- technology.packaging
- technology.subversive
- tools.buckminster
- tools.cdt
- tools.gef
- tools.mylyn
- tools.pdt
- tools.ptp
- tptp
- webtools
- webtools.common
- webtools.dali
- webtools.ejbtools
- webtools.jeetools
- webtools.jsdt
- webtools.jsf
- webtools.servertools

- webtools.sourceediting
- webtools.webservices