Eclipse Development Process 2014

Contents

1. Purpose

This document describes the development process for the Eclipse Foundation. In particular, it describes how the membership at large, the board of directors, other constituents of the ecosystem, and the Eclipse Management Organization (EMO) lead, influence, and collaborate with Eclipse projects to achieve these Eclipse purposes:

> *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., 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.*

This document has the following sections:

- *Principles* outlines the basic principles upon which the development process is based.
- *Requirements* describes the requirements that the Eclipse community has for its development process.
- *Structure and Organization* specifies the structure and organization of the projects and project community at Eclipse.
- *Development Process* outlines the lifecycle and processes required of all Eclipse projects.

2. Principles

The following describes the guiding principles used in developing this development process.

2.1 Open Source Rules of Engagement

- **Open** - Eclipse is open to all; Eclipse provides the same opportunity to all. Everyone participates with the same rules; there are no rules to exclude any potential contributors which include, of course, direct competitors in the marketplace.
- **Transparent** - Project discussions, minutes, deliberations, project plans, plans for new features, and other artifacts are open, public, and easily accessible.
- **Meritocracy** - Eclipse is a meritocracy. The more you contribute the more responsibility you will earn. Leadership roles in Eclipse are also merit-based and earned by peer acclaim.

2.2 Eclipse Ecosystem

Eclipse as a brand is the sum of its parts (all of the projects), and projects should strive for the highest possible quality in extensible frameworks, exemplary tools, transparent processes, and project openness.

The Eclipse Foundation has the responsibility to *...cultivate...an ecosystem of complementary products, capabilities, and services...* . It is therefore a key principle that the Eclipse Development Process ensures that the projects are managed for the benefit of both the open source community and the ecosystem members. To this end, all Eclipse projects are required to:

- communicate their project plans and plans for new features (major and minor) in a timely, open and transparent manner;
- create platform quality frameworks capable of supporting the building of commercial grade products on top of them; and
- ship extensible, exemplary tools which help enable a broad community of users

2.3 Three Communities

Essential to the purposes of the Eclipse Foundation is the development of three inter-related communities around each project:

2.3.1 Contributors and Committers

A thriving, diverse, and active community of developers is the key component of any Eclipse project. Ideally, this community should be an open, transparent, inclusive, and diverse community of committers and (non-committer) contributors. Attracting new contributors and committers to an open source project is time consuming and requires active recruiting, not just passive "openness". The project leadership must make reasonable efforts to encourage and nurture promising new contributors.

2.3.2 Users

An active and engaged user community is proof-positive that the project's exemplary tools are useful and needed. Furthermore, a large user community is one of the key factors in creating a viable ecosystem around an Eclipse project, thus encouraging additional open source and commercial organizations to participate. Like all good things, a user community takes time and effort to bring to fruition, but once established is typically self-sustaining.

2.3.3 Adopters

An active and engaged adopter/plug-in developer community is the only way to prove that an Eclipse project is providing extensible frameworks and extensible tools accessible via documented APIs. Reuse of the frameworks within the companies that are contributing to the project is necessary, but not sufficient to demonstrate an adopter community. Again, creating, encouraging, and nurturing an adopter community outside of the project's developers takes time, energy, and creativity by the project leadership, but is essential to the project's long-term open source success.

The Eclipse community considers the absence of any one or more of these communities as proof that the project is not sufficiently open, transparent, and inviting, and/or that it has emphasized tools at the expense of extensible frameworks or vice versa.

2.4 Clear, Concise, and Evolving

It is an explicit goal of the development process to be as clear and concise as possible so as to help the project teams navigate the complexities, avoid the pitfalls, and become successful as quickly as possible.

This document imposes requirements and constraints on the operation of the projects, and it does so on behalf of the larger Eclipse community. It is an explicit goal of the development process to provide as much freedom and autonomy to the projects as possible while ensuring the collective qualities benefit the entire Eclipse community.

Similarly, this document should not place undue constraints on the EMO or the board that prevent them from governing the process as necessary. We cannot foresee all circumstances and as such should be cautious of being overly prescriptive and/or requiring certain fixed metrics.

The frameworks, tools, projects, processes, community, and even the definition of quality continues to, and will continue to, evolve. Creating rules or processes that force a static snapshot of any of these is detrimental to the health, growth, and ecosystem impact of Eclipse.

Part of the strength of this document is in what it does not say, and thus opens for community definition through convention, guidelines, and public consultation. A document with too much structure becomes too rigid and prevents the kind of innovation and change we desire for Eclipse. In areas where this document is vague, we expect the projects and members to engage the community-at-large to clarify the current norms and expectations.

3. Requirements

This document and any additional criteria as established by the EMO contains requirements, recommendations, and suggestions.

Required - Certain responsibilities and behaviors are required of participants in Eclipse open source projects. Projects that fail to perform the required behaviors will be terminated by the EMO. In keeping with the guiding principles, the number of requirements must be kept to an absolute minimum.

Guideline - Other responsibilities and behaviors are recommended best practices. Collectively, we have learned that projects are more likely to be successful if the team members and leaders follow these recommendations. Projects are strongly encouraged to follow these recommendations, but will not be penalized by this process if they do not.

3.1 Requirements and Guidelines

This document is entirely composed of requirements. In addition to the requirements specified in this development process, the EMO is instructed to clarify, expand, and extend this process by creating a set of Eclipse project development guidelines 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 and services.

The EMO is not permitted to override or ignore the requirements listed in this document without the express written endorsement of the board of directors.

4. Project Structure and Organization

A project is the main operational unit at Eclipse. Specifically, all open source software development at Eclipse occurs within the context of a project. Projects have leaders, developers, code, builds, downloads, websites, and more. Projects are more than just the sum of their many parts, they are the means by which open source

work is organized when presented to the communities of developers, adopters, and users. Projects provide structure that helps developers expose their hard work to a broad audience of consumers.

Eclipse projects are organized hierarchically. A special type of project, "top-level" projects, sit at the top of the hierarchy. Each top-level project contains one or more projects. Each project may itself contain zero or more projects. A project that has one or more projects is said to be the "parent" of those projects. A project that has a parent is oftentimes referred to as a "subproject". The term project refers to either a top-level project or a subproject.

The descendants of a project are the project itself and transitive closure of its child projects. The top parent of a project is the top-level project at the top of the hierarchy.

Projects are the unit entity for:

- committers;
- code and releases;
- intellectual property (IP) records; and
- community awareness

As defined by Bylaws of Eclipse Foundation - Article VII, the "Eclipse Management Organization" (EMO) consists of the Eclipse Foundation staff and the councils. The term EMO(ED), when discussing an approval process, refers to the subset of the EMO consisting of the executive director and whomever he or she may delegate that specific approval authority to.

4.1 Committers

Each project has exactly one set of committers. Each project's set of committers is distinct from that of any other project, including subprojects or parent projects. All project committers have equal rights and responsibilities within the project. Partitioning of responsibility within a project is managed using social convention. A project may, for example, divide itself into logical partitions of functionality; it is social convention that prevents committers from one logical partition from doing inappropriate work in another. If finer-grained management of committer responsibilities is required, a project should consider partitioning (via a Restructuring Review) into two or more subprojects.

The committers of a project have the exclusive right to elect new committers to their project; no other group, including a parent project, can force a project to accept a new committer.

There is no roll-up of committers: the set of committers on a project is exactly that set of people who have been explicitly elected into that role for the project (i.e. being a committer on a subproject does not give you any automatic rights on the "parent" project or any child project).

In practical terms, each project has a single UNIX group of its committers that provides write-access to the project's resources. Pictorially below, we see that a project, in addition to the various resources and committers it has, can also have zero or more subprojects. Each of these subprojects has its own distinct set of committers and resources.

4.2 Code and Resources

Each project owns and maintains a collection of resources.

Resources may include source code, a project website, space on the downloads server, access to build resources, and other services provided by the Eclipse Foundation infrastructure. The exact infrastructure provided by the Eclipse Foundation varies over time and is defined outside this process document.

A project is not strictly required to make use of all the resources made available; a project might, for example, opt to *not* maintain a source code repository. Such a project might operate as an organizational unit, or container, for several subprojects. Similarly, a project might opt to provide a consolidated website, build and/or download site for its subprojects (the subprojects would then not require those resources for themselves).

Namespaces are assigned to a project by the EMO. All project source code must be organized in the assigned namespaces and projects can only release code under their own namespace (that is, they cannot infringe on another Eclipse project's namespace). Projects should work with their PMCs and the EMO to request exceptions to this rule, and with their mentors and PMC if there are questions regarding the use of the namespace.

4.3 Intellectual Property (IP) Records

A project at any level may receive IP clearance for contributions and third-party libraries. IP approval will often include the same approval for all descendant projects. However, IP clearance will only be granted at the most appropriate technical level.

4.4 Community Awareness

Projects are the level of communication with the larger Eclipse community and ecosystem. Projects may either have their own communications (website, mailing lists, forums/newsgroups, etc) or they may be part of a parent project's communications (website, mailing list, forums/newsgroups, etc). In either case, the project is required to maintain an open and public communication channel with the Eclipse community including, but not limited to, project plans, schedules, and design discussions.

All projects must make the communication channels easy to find. Projects are further required to make the separate communication channels of their child projects (if any) easy to find.

Any project in the incubation phase must correctly identify its website and releases. A project with at least one descendant project in incubation phase must correctly annotate its own website so as to notify the Eclipse community that incubating projects exist in its hierarchy. Any release containing code from an incubation phase project must be correctly labeled, i.e., the incubation phase is viral and expands to cover all releases in which it is included.

4.5 Scope

Each top-Level project has a charter which describes the purpose, scope, and operational rules for the top-level project. The charter should refer to, and describe any refinements to, the provisions of this development process. The board of directors approves the charter of each top-level project.

Subprojects do not have separate charters; subprojects operate under the charter of their parent top-Level project.

All projects have a defined scope and all initiatives within that project are required to reside within that scope. Initiatives and code that is found to be outside the scope of a project may result in the termination of the project. The scope of top-level projects is part of the charter, as approved by the board of directors of the Eclipse Foundation.

The scope of a subproject is defined by the initial project proposal as reviewed and approved by the Project Management Committee (PMC) (as further defined below) of the project's top parent and by the EMO. A project's scope must be a subset of its parent's scope.

4.6 Leaders

There are two different types of project leadership at Eclipse: The Project Management Committee (PMC) and project leads. Both forms of leadership are required to:

- ensure that their project is operating effectively by guiding the overall direction and by removing obstacles, solving problems, and resolving conflicts;
- operate using open source rules of engagement: meritocracy, transparency, and open participation; and
- ensure that the project and its subprojects (if any) conform to the Eclipse Foundation IP policy and procedures.

The leadership chain for a project is composed of the project's project lead(s), the leadership of the parent project (if any), the PMC leads and PMC members for the top-level project, the EMO, and the EMO(ED).

In exceptional situations—such as projects with zero active committers, disruptive committers, or no effective project leads—the project leadership chain has the authority to make changes (add, remove) to the set of committers and/or project leads of that project, and otherwise act on behalf of the project lead.

4.6.1 Project Management Committee (PMC)

Top-level projects are managed by a Project Management Committee (PMC). A PMC has one or more PMC leads and zero or more PMC Members. Together the PMC provides oversight and overall leadership for the projects that fall under their top-level project. The PMC as a whole, and the PMC leads in particular, are ultimately responsible for ensuring that the Eclipse Development Process is understood and followed by their projects. The PMC is additionally responsible for maintaining the top-level project's charter.

PMC leads are approved by the board of directors; PMC members are elected by the existing PMC leads and members, and approved by the EMO(ED).

4.6.2 Project Lead

Eclipse projects are managed by one or more project leads. Project leads are responsible for ensuring that their project's committers are following the Eclipse Development Process, and that the project is engaging in the right sorts of activities to develop vibrant communities of users, adopters, and contributors. The initial project leads are appointed and approved in the creation review. Subsequently, additional project leads must be elected by the project's committers and approved by the project's PMC and the EMO(ED).

In the unlikely event that a project lead becomes disruptive to the process or ceases to contribute for an extended period, the individual may be removed by the unanimous vote of the remaining project leads (if there are at least two other project leads), or unanimous vote of the project's PMC.

4.7 Committers and Contributors

Each project has a development team, led by the project leaders. The development team is composed of committers and contributors. Contributors are individuals who contribute code, fixes, tests, documentation, or other work that is part of the project. Committers have write access to the project's resources (source code repository, bug tracking system, website, build server, downloads, etc.) and are expected to influence the project's development.

Contributors who have the trust of the project's committers can, through election, be promoted committer for that project. The breadth of a committer's influence corresponds to the breadth of their contribution. A development team's contributors and committers may (and should) come from a diverse set of organizations. A committer gains voting rights allowing them to affect the future of the project. Becoming a committer is a privilege that is earned by contributing and showing discipline and good judgment. It is a responsibility that should be neither given nor taken lightly, nor is it a right based on employment by an Eclipse member company or any company employing existing committers.

The election process begins with an existing committer on the same project nominating the contributor. The project's committers will vote for a period of no less than one week of standard business days. If there are at least three (3) positive votes and no negative votes within the voting period, the contributor is recommended to the project's PMC for commit privileges. If there are three (3) or fewer committers on the project, a unanimous positive vote of all committers is substituted. If the PMC approves, and the contributor signs the appropriate committer legal agreements established by the EMO (wherein, at the very least, the developer agrees to abide by the Eclipse Intellectual Property Policy), the contributor becomes a committer and is given write access to the source code for that project.

At times, committers may become inactive for a variety of reasons. The decision making process of the project relies on active committers who respond to discussions and vote in a constructive and timely manner. The project leads are responsible for ensuring the smooth operation of the project. A committer who is disruptive, does not participate actively, or has been inactive for an extended period may have his or her commit status revoked by the project leads. Unless otherwise specified, "an extended period" is defined as "no activity for more than six months".

Active participation in the user communication channels and the appropriate developer mailing lists is a responsibility of all committers, and is critical to the success of the project. Committers are required to monitor and contribute to the user communication channels.

Committers are required to monitor the mailing lists associated with the project. This is a condition of being granted commit rights to the project. It is mandatory because committers must participate in votes (which in some cases require a certain minimum number of votes) and must respond to the mailing list in a timely fashion in order to facilitate the smooth operation of the project. When a committer is granted commit rights they will be added to the appropriate mailing lists. A committer must not be unsubscribed from a developer mailing list unless their associated commit privileges are also revoked.

Committers are required to track, participate in, and vote on, relevant discussions in their associated projects. There are three voting responses: +1 (yes), -1 (no, or veto), and 0 (abstain).

Committers are responsible for proactively reporting problems in the bug tracking system, and annotating problem reports with status information, explanations, clarifications, or requests for more information from the submitter. Committers are responsible for updating problem reports when they have done work related to the problem.

Committer, PMC lead or member, project lead, and council representative(s) are roles; an individual may take on more than one of these roles simultaneously.

4.8 Councils

The councils defined in the bylaws, section VII are comprised of strategic members and PMC representatives. The councils help guide the projects as follows:

- The Planning Council is responsible for establishing a coordinated simultaneous release (a.k.a, "the release train"). The Planning Council is further responsible for cross-project planning, architectural issues, user interface conflicts, and all other coordination and integration issues. The Planning Council discharges its responsibility via collaborative evaluation, prioritization, and compromise.
- The Architecture Council is responsible for (i) monitoring, guiding, and influencing the software architectures used by projects, (ii) new project mentoring, and (iii) maintaining and revising the Eclipse Development Process. Membership in the Architecture Council is per the bylaws through strategic membership, PMCs, and by appointment. The Architecture Council will, at least annually, recommend to the EMO(ED), Eclipse Members who have sufficient experience, wisdom, and time to be appointed to the Architecture Council and serve as mentors. Election as a mentor is a highly visible confirmation of the Eclipse community's respect for the candidate's technical vision, good judgement, software development skills, past and future contributions to Eclipse. It is a role that should be neither given nor taken lightly. Appointed members of the Architecture Council are appointed to two year renewable terms.

4.9 Permanent Incubator Projects

A project may designate a subproject as a "permanent incubator". A permanent incubator is a project that is intended to perpetually remain in the incubation phase. Permanent incubators are an excellent place to innovate, test new ideas, grow functionality that may one day be moved into another project, and develop new committers.

Permanent incubator projects never have releases; they cannot participate in the annual simultaneous release. Permanent incubators may have builds, and downloads. They conform to the standard incubation branding requirements and are subject to the IP due diligence rules outlined for incubating projects. Permanent incubators do not graduate.

The scope of a permanent incubator project must fall within the scope of its parent project. The committer group of the permanent incubator project must overlap with that of the parent project (at least one committer from the parent project must be a committer for the incubator). Permanent incubator projects do not require Architecture Council mentors (the parent project's committers are responsible for ensuring that the incubator project conforms to the rules set forth by the Eclipse Development Process).

A permanent incubator project must be designated as such by including the word "incubator" in its name (e.g. "Eclipse Incubator"). To do otherwise is considered exceptional and requires approval from the PMC and EMO(ED).

Only top-level projects and projects in the mature phase may create a permanent incubator. Permanent incubator projects are created upon request; a creation review is not required.

5. [Reserved]

6. Development Process

Projects must work within their scope. Projects that desire to expand beyond their current scope must seek an enlargement of their scope using a public review as described below. Further, projects must fit within the scope defined by their containing projects and the scope defined in the charter of their top-level project.

Projects must provide advanced notification of upcoming features and frameworks via their project plan.

6.1 Mentors

New project proposals are required to have at least two mentors. Mentors must be members of the Architecture Council. The mentors must be listed in the proposal. Mentors are required to monitor and advise the new project during its incubation phase; they are released from that duty once the project graduates to the mature phase.

6.2 Project Lifecycle

Projects go through distinct phases. The transitions from phase to phase are open and transparent public reviews.

6.2.1 Pre-proposal

An individual or group of individuals declares their interest in, and rationale for, establishing a project. The EMO will assist such groups in the preparation of a project proposal.

The pre-proposal phase ends when the proposal is published by EMO and announced to the membership by the EMO.

6.2.2 Proposal

The proposers, in conjunction with the destination PMC and the community, collaborate in public to enhance, refine, and clarify the proposal. Mentors for the project must be identified during this phase.

The proposal phase ends with a creation review, or withdrawal. The proposal may be withdrawn by the proposers at any point before the start of a creation review. The EMO will withdraw a proposal that has been inactive for more than six months.

6.2.3 Incubation

The purpose of the incubation phase is to establish a fully-functioning open-source project. In this context, incubation is about developing the process, the community, and the technology. Incubation is a phase rather than a place: new projects may be incubated under any existing project.

- A project in the incubation phase can (and should) make releases;
- Top-level projects skip incubation and are immediately put into the mature phase;
- The incubation phase ends with a graduation review or a termination review.
- Designated permanent incubator projects remain perpetually in the incubation phase; they do not create releases, so no reviews are required.

Many Eclipse projects are proposed and initiated by individuals with extensive and successful software development experience. This document attempts to define a process that is sufficiently flexible to learn from all its participants. At the same time, however, the incubation phase is useful for new projects to learn the community-defined Eclipse-centric open source processes.

Only projects that are properly identified as being in the incubation phase (including designated permanent incubator projects) may use the Parallel IP Process to reduce IP clearance process for new contributions.

6.2.4 Mature

The project team has demonstrated that they are an open-source project with an open and transparent process; an actively involved and growing community; and Eclipse-quality technology. The project is now a mature member of the Eclipse community. Major releases continue to go through release reviews.

6.2.5 [Reserved]

6.2.6 Archived

Projects that become inactive, either through dwindling resources or by reaching their natural conclusion, are archived. Projects can reach their natural conclusion in a number of ways: for example, a project might become so popular that it is absorbed into one of the other major frameworks. Projects are moved to archived status through a termination review.

If there is sufficient community interest in reactivating an archived project, the project can start again with a creation review. As there must be good reasons to have terminated a project, the creation review provides a sufficiently high bar to prove that those reasons are no longer valid.

6.3 Reviews

The Eclipse Development Process is predicated on open and transparent behavior. All major changes to Eclipse projects must be announced and reviewed by the membership-at-large. Major changes include the project phase transitions as well as the introduction or exclusion of significant new technology or capability. It is a clear requirement of this document that members who are monitoring the appropriate media channels (e.g., mailing lists or RSS feeds) not be surprised by the post-facto actions of the projects.

For each review, the project leads prepare documentation for, and receive feedback from, the Eclipse membership.

A review is a fairly comprehensive process. Gathering the material for a review and preparing the documentation is a non-trivial effort, but the introspection offered by this exercise is useful for the project and results are very useful for the entire Eclipse community. In addition, reviews have a specific relationship to the requirements of the Eclipse IP Policy.

All reviews have the same general process:

1. Projects are responsible for initiating the appropriate reviews. However, if a project does not do so and the EMO believes a review is necessary, the EMO may initiate a review on the project's behalf.
2. A review then continues with a project lead requesting that the EMO(ED) schedule the review.
3. Prior to the start of the review period, a project lead provides the EMO with review documentation.
    - The review documentation material always includes a document that describes the review. The minimum contents of the document are specified by the individual review types.
    - The review documentation must be available in a format that anyone in the Eclipse membership can review. PDF and HTML are acceptable single formats.
    - The review documentation must have a correct copyright statement and license.
    - The review documentation must be *archival quality* . This means that the materials must be comprehensible and complete on their own without requiring explanation by a human presenter, reference to a wiki, or to other non-archived web pages.
4. The EMO announces the review schedule and makes the documentation available to the membership-at-large.

The criteria for the successful completion of each type of review will be documented in writing by the EMO in guidelines made available via an Eclipse Foundation website. Such guidelines will include, but are not limited to the following:

1. Clear evidence that the project has vibrant committer, adopter and user communities as appropriate for the type of review.
2. Reasonable diversity in its committer population as appropriate for the type of review. Diversity status must be provided not only as number of people/companies, but also in terms of effort provided by those people/companies.
3. Documented completion of all required due diligence under the Eclipse IP Policy.
4. For graduation and release reviews, the project must have a current project plan, in the format specified by the EMO, available to the community.
5. Balanced progress in creating both frameworks and extensible, exemplary tools.
6. Showcase the project's quality through project-team chosen metrics and measures, e.g., coupling, cyclomatic complexity, test/code coverage, documentation of extensions points, etc.

The review period is open for no less than one week and usually no more than two weeks of generally accepted business days.

1. The review begins with the EMO's posting of the review materials at the start of the review period
2. The proper functioning of the Eclipse Development Process is contingent on the active participation of the Eclipse members and committers, especially in reviews, thus each review has an EMO-designated discussion and feedback communication channel: a forum/newgroup, a mailing list, or some other public forum.
3. If a committer election is required for a review (for example, for a creation review), then it is held simultaneously with the review period. Thus the election and the review will end at the same time, allowing quick and efficient provisioning of the resulting project.
4. The EMO(ED) approves or fails the review based on the public comments, the scope of the project, and the purposes of the Eclipse Foundation as defined in the bylaws.
5. The review ends with the announcement of the results in the defined review communication channel (the EMO(ED) will request that the project lead make this announcement).

If any member believes that the EMO has acted incorrectly in approving or failing a review may appeal to the board of directors to review the EMO's decision.

6.3.1 Creation Review

The purpose of the creation review is to assess the community and membership response to the proposal, to verify that appropriate resources are available for the project to achieve its plan, and to serve as a committer election for the project's initial committers. The Eclipse Foundation strives not to be a repository of "code dumps" and thus projects must be sufficiently staffed for forward progress.

The creation review documents must include short nomination bios of the proposed initial committers. These bios should discuss their relationship to, and history with, the incoming code and/or their involvement with the area/technologies covered by the proposal. The goal is to help keep any legacy contributors connected to new project and explain that connection to the current and future Eclipse membership, as well as justify the initial committers' participation in a meritocracy.

6.3.2 Graduation Review

The purpose of the graduation review is to mark a project's change from the incubation phase to the mature phase.

The graduation review confirms that the project is/has:

- A working and demonstrable code base of sufficiently high quality.
- Active and sufficiently diverse communities appropriate to the size of the graduating code base: adopters, developers, and users.
- Operating fully in the open following the principles and purposes of Eclipse.
- A credit to Eclipse and is functioning well within the larger Eclipse community.

A graduation review is generally combined with a release review.

6.3.3 Release Review

The purposes of a release review are: to summarize the accomplishments of the release, to verify that the IP Policy has been followed and all approvals have been received, to highlight any remaining quality and/or architectural issues, and to verify that the project is continuing to operate according to the principles and purposes of Eclipse.

6.3.4 [Reserved]

6.3.5 [Reserved]

6.3.6 Termination Review

The purpose of a termination review is to provide a final opportunity for the committers and/or Eclipse membership to discuss the proposed archiving of a Project. The desired outcome is to find sufficient evidence of renewed interest and resources in keeping the project active.

6.3.7 [Reserved]

6.3.8 Restructuring Review

The purpose of a restructuring review is to notify the community of significant changes to one or more projects. Examples of "significant changes" include:

- Movement of significant chunks of functionality from one project to another.
- Modification of the project structure, e.g. combining multiple projects into a single project, or decomposing a single project into multiple projects.
- Change of project scope.

6.3.9 Combining Reviews

Reviews can be combined at the discretion of the PMC and EMO. Multiple projects may participate in a single review. Similarly, multiple review types can be engaged in simultaneously. A parent project may, for example, engage in an aggregated release review involving itself and some or all of its child projects; a consolidated restructuring review may move the code for several projects; or a release review may be combined with a graduation review. When multiple reviews are combined, the review documentation must explicitly state all of the projects and types of reviews involved, and include the required information about each.

It should be noted that the purpose of combining reviews is to better serve the community, rather than to reduce effort on the part of the project (though it is fortunate when it does both). Combining a release and graduation review, or aggregating a release review of a project and several of it's child projects generally makes sense. Combining release reviews for multiple unrelated projects most likely does not.

6.4 Releases

Any project, with exception of permanent incubators, may make a release. A release may include the code from any subset of the project's descendants.

(Most of this section is borrowed and paraphrased from the excellent Apache Software Foundation Releases FAQ. The Eclipse community has many of the same beliefs about Releases as does the Apache community and their words were already excellent. The Apache Software Foundation Releases FAQ is distributed under the Apache License, Version 2.0.)

Releases are, by definition, anything that is distributed outside of the committers of a project. If users are being directed to download a build, then that build has been released (modulo the exceptions below). All projects and committers must obey the Eclipse Foundation requirements on approving any release.

(Exception 1: nightly and integration builds) During the process of developing software and preparing a release, various nightly and integration builds are made available to the developer community for testing purposes. Do not include any links on the project website, blogs, wikis, etc. that might encourage non-early-adopters to download and use nightly builds, release candidates, or any other similar package (links aimed at early-adopters and the project's developers are both permitted and encouraged). The only people who are supposed to know about such packages are the people following the developer mailing list and thus are aware of the limitations of such builds.

(Exception 2: milestone and release candidate builds) Projects are encouraged to use an agile development process including regular milestones (for example, six week milestones). Milestones and release candidates are "almost releases" intended for adoption and testing by early adopters. Projects are allowed to have links on the project website, blogs, wikis, etc. to encourage these outside-the-committer-circle early adopters to download and test the milestones and release candidates, but such communications must include caveats explaining that these are not official releases.

- Milestones are to be labeled `x.yMz` , e.g., 2.3M1 (milestone 1 towards version 2.3), 2.3M2

(milestone 2 towards version 2.3), etc.
- Release candidates are to be labeled `x.yRCz` , e.g., 2.3RC1 (release candidate 1 towards version 2.3).
- Official releases are the only downloads allowed to be labeled with `x.y` , e.g., 0.5, 1.0, 2.3, etc.

All official releases must have a successful release review before being made available for download.

(Exception 3: bug fix releases with no new features) Bug fix releases (x.y.z, e.g., 2.3.1) with no new features over the base release (e.g., 2.3) are allowed to be released without an additional release review. If a bug fix release contains new features, then the project must have a full release review.

Under no circumstances are builds and milestones to be used as a substitute for doing proper official releases. Proper release management and reviews is a key aspect of Eclipse quality.

Releases for projects in the incubation phase must be labeled to indicate the incubation status of the project.

6.5 Grievance Handling

When a member has a concern about a project, the member will raise that concern with the project's leadership. If the member is not satisfied with the result, the member can raise the concern with the parent project's leadership. The member can continue appeals up the project leadership chain and, if still not satisfied, thence to the EMO, then the executive director, and finally to the board of directors. All appeals and discussions will abide by the guiding principles of being open, transparent, and public.

Member concerns may include:

- Out of scope. It is alleged that a project is exceeding its approved scope.
- Dysfunctional. It is alleged that a project is not functioning correctly or is in violation of one or more requirements of the Eclipse Development Process.
- Contributor appeal. It is alleged that a contributor who desires to be a committer is not being treated fairly.
- Invalid veto. It is alleged that a -1 vote on a review is not in the interests of the project and/or of Eclipse.

A variety of grievance resolutions are available to the EMO up to, and including, rebooting or restarting a project with new Committers and leadership.

7. Precedence

In the event of a conflict between this document and a board of directors-approved project charter, the most recently approved document will take precedence.

8. Revisions

As specified in the bylaws, the EMO is responsible for maintaining this document and all changes must be approved by the board of directors.

Due to the continued evolution of the Eclipse technology, the Eclipse community, and the software marketplace, it is expected that the Eclipse Development Process (this document) will be reviewed and revised on at least an annual basis. The timeline for that review should be chosen so as to incorporate the

lessons of the previous annual coordinate release and to be applied to the next annual coordinated release.

The EMO is further responsible for ensuring that all plans, documents and reports produced in accordance with this development process be made available to the membership at large via an appropriate mechanism in a timely, effective manner.

8.1 Revision 2.7

This document was approved by the Eclipse Foundation Board of Directors in its meeting on [TBD]. It takes effect (replacing all previous versions) on [TBD].

8.2 History

Changes made in this document:

- (Bug 344041) Yearly reviews are no longer required
- (Bug 367235) Confusing language regarding termination of project lead role
- (Bug 368196) A graduation review is generally combined with a release review
- (Bug 415626) Restructuring review section too verbose
- (Bug 415629) Permanent Incubator Projects do not require a creation review
- (Bug 415636) Ensure that social coding is adequately addressed by the EDP
- (Bug 415696) Move paragraph concerning releases from section 4.2 to section 6.4
- (Bug 415714) Remove reference to "Bugzilla"
- (Bug 415715) Fix capitalization
- (Bug 416636) "Top-Level" is not a phase
- (Bug 417883) Termination reviews not required for proposals
- (Bug 418346) Reduce redundancy in 2.3 "Three Communities"
- (Bug 418468) Rename Incubators to "Permanent Incubators"
- (Bug 419717) Complete section 8.2 "History"
- (Bug 419721) Confusing use of "project leadership" vs. "project lead"
- (Bug 419724) Include the EMO(ED) in the project leadership chain
- (Bug 325004) Eliminate Continuation, Promotion, and Move Reviews
- (Bug 331397) EDP should say more about project namespaces
- (Bug 345755) Eliminate the pre-1.0 versioning requirement for incubating projects