» The Linux Foundation

Understanding the Open Source Development Model

November 2011

By Ibrahim Haddad (Ph.D.) and Brian Warner, The Linux Foundation

A White Paper By The Linux Foundation
http://www.linuxfoundation.org

This paper presents an overview of the open source development model. It discusses the typical progression from an idea to an implemented feature, and highlights some of the key characteristics of open source development.

Introduction

The open source software development model is characterized by processes and values that set it apart from the traditional proprietary development model.

The software development model practiced by many organizations generally consists of discrete periods of development activity that cascade towards a project's release. The open source development model takes a different approach, favoring a more fluid development process characterized by increased intra-team collaboration, continuous integration and testing, and greater end-user involvement.

The open source development model is being increasingly adopted within traditional development organizations as a means of producing higher quality software, even within companies that are not producing an open source product. This is generally due to the increased efficiencies the open source development model offers to large, distributed teams working on major software projects.

This paper examines the open source development model and describes typical processes for managing feature requests, source code submissions, and architectural decisions. It will also discuss foundational characteristics of the open source life-cycle such as peer review, the "release early and often" mentality, and continuous testing and integration.

Please note that every open source project has its own way of managing their development process. The description in this article is not specific to any one project, but rather describes a process and characteristics that would apply to most open source projects.

The Open Source Development Model

The open source development model presumes that development is distributed among multiple teams, working in different locations, in a fluid structure that is resilient to new arrivals or departures. Successful open source communities have developed processes where code can be submitted and integrated asynchronously, communication is well documented, and features are integrated in small increments to catch issues early in the development cycle.

One of the core characteristics of the open source development model is that individuals or small teams of contributors are responsible for development and maintenance of code, illustrated in Figure 1. Contributed features are integrated into a single body of code by one or more maintainers, who ensure newly submitted code meets the overall vision and standards set for the project.

The feature development life-cycle, illustrated in Figure 2, begins with an idea for a new project, feature or enhancement, which is proposed to other project developers. Following further discussion about the need for the feature, the next step is to design and implement it.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

THE LINUX FOUNDATION

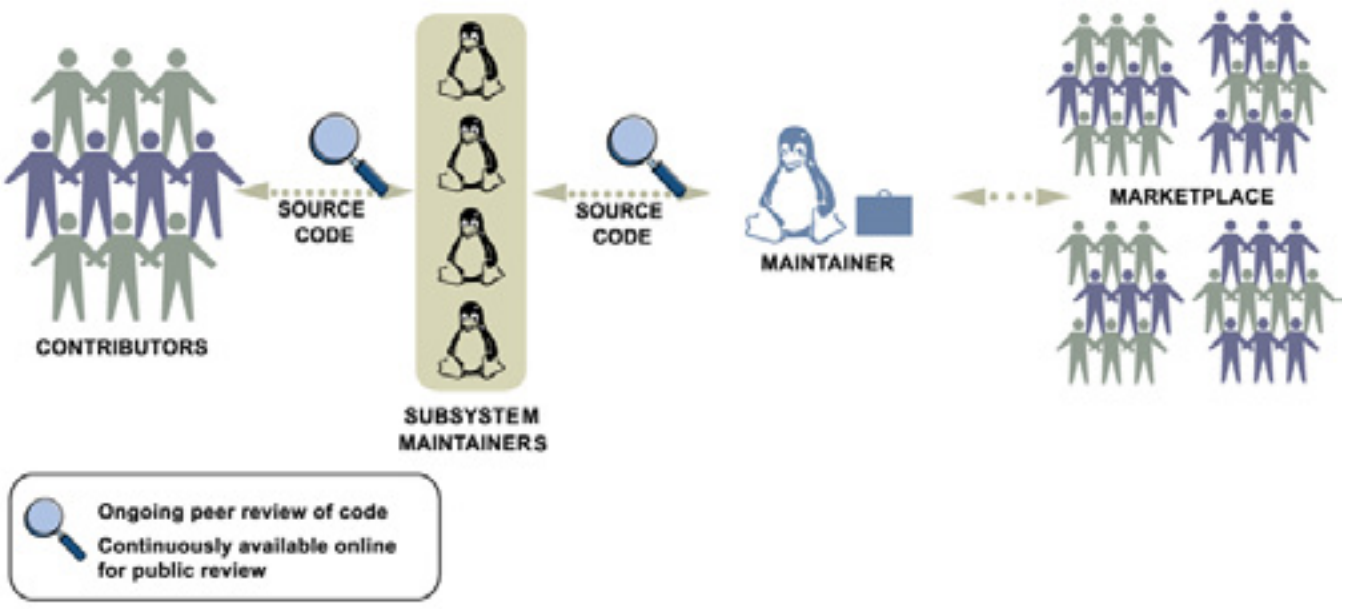

Figure 1: Flow of source code from contributors to the mainline version to the marketplace

When the new feature compiles and runs, it is typically distributed within the development community as an alpha release, even though it may contain known and unknown bugs. The purpose of early distribution is to collect feedback and allow users to test and provide input. This is commonly called "release early and often." Users may provide feedback, bug reports, and fixes, which are integrated into the next development release. This cycle repeats, until the developers feel that the implementation is stable enough to submit for inclusion into the main project.

The author of the code then submits the code to a project maintainer over the project mailing list. The maintainer determines whether the code should be accepted into the development tree, or returned for revision. Some projects may have multiple layers of maintainers, depending on the complexity of the code and the size of the project.

When the code has been signed off by all relevant maintainers, it is then included into the project's main source tree for publishing in the next release.

The Open Source Feature Life-Cycle

Feature Request Process

Feature requests are generally tracked and prioritized using processes that are visible to the rest of the development community. This ensures a common understanding of which features have been requested, their relative priority, their development status, associated bugs and blockers, and when they are planned for release.

Figure 3 shows the typical process used to ensure that feature requests are accurately tracked, prioritized, developed, and released.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

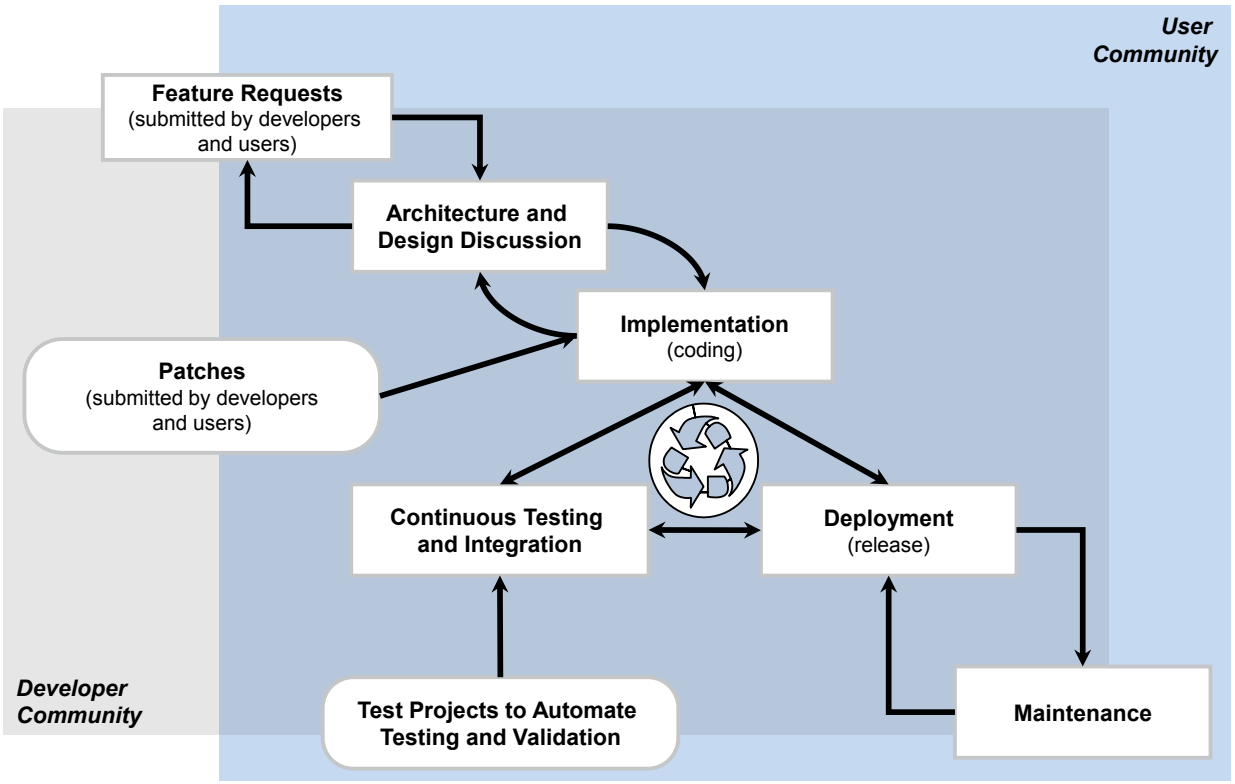

Figure 2: Feature life-cycle in the open source development model

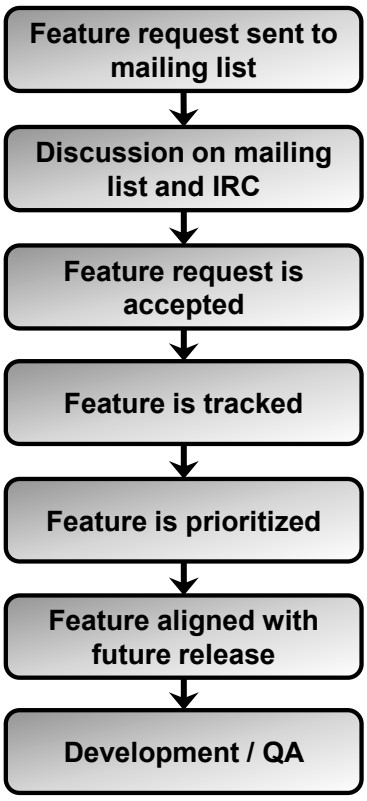

Figure 3: Flow of source code from contributors to the mainline version to the marketplace

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

Feature requests may begin with a proposal to a mailing list, a discussion on IRC, or as a feature request in a project's bugzilla. The request is typically made by whoever is likely to lead the implementation work. The purpose of the request is to notify others of the need, solicit feedback, gain acceptance for the idea, and come to a consensus on next steps.

Project contributors and maintainers then evaluate the request, and determine whether it should be a candidate for a future release. There will often be discussions between the requester and the development community to clarify needs and requirements. If the request is approved, a target release will be set, and development begins.

Architecture and Design Discussion

One of the major contributing factors to the success of the open source development model is its transparency, and ability to accommodate distributed collaboration among project teams. This is accomplished using communication methods that are accessible to all within the project community for strategic decision making, architecture discussions, and code reviews.

Mailing lists are one of the most commonly used communication channels because they are self-documenting, transparent, and typically anyone involved in the project can participate. This includes end users, who may be monitoring the lists to understand future features as they evolve or to provide practical feedback.

In addition to project mailing lists, many distributed teams use IRC for live discussion and meetings. Because of its text-only nature, IRC is useful for design meetings and user support, especially when English is not the primary spoken language of all participants.

Collaboration on Implementation

The open source development model places strong emphasis upon collaborative development and peer review, from first idea to final acceptance. Because it evolved to support highly decentralized teams where submitters were not all personally known to the maintainers, the model favors those who work with others on design and implementation while clearly communicating their plans.

In addition, most open source projects use tools that have evolved to support code contribution from many simultaneous and distributed collaborators. For example, the open source git repository system was created specifically to support Linux development, where thousands of contributors are submitting code for any given release. Each developer works to develop, debug, build, and validate their own code against the current code-base, so that when the time comes to integrate into the mainline project, their changes apply cleanly and with a minimum amount of merging effort. If there is an unforeseen problem with the code, any individual submission can be easily reverted.

Source Code Submission

The life-cycle of a new code submission, illustrated in Figure 4, is often quite iterative. The process begins with collaborative development among a subset of developers who have taken ownership for delivering the feature. When the code is functional and applies cleanly against the mainline project, the project team submits the code to a project maintainer over the project mailing list. The maintainer and other project participants may provide feedback on the submission and decline to accept it, in which case the implementation team would revise and resubmit the code.

Because smaller patches are easier to understand and test, submitters are generally encouraged

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

to submit changes in the smallest increments possible. Smaller patches are less likely to have unintended consequences, and if they do, getting to root cause of an issue is much easier.

Establishing Ownership

Because code submissions can come from anyone, most projects have formal procedures in place to track ownership of code when a patch is submitted.

The "Signed-off-by" line provides the real name and email of the person who is responsible for the code. It is also an agreement to the Developer's Certificate of Origin, which requires that the submitter have the rights to contribute the code. At least one signed-off-by line is typically required, as it enables others to quickly determine who submitted code if there is ever a question over origin, license, or maintenance.

In a similar manner, some projects also track reviewers. The "Acked-by" line indicates when someone other than the author or the maintainer has provided a thorough review and believes it is ready for integration.

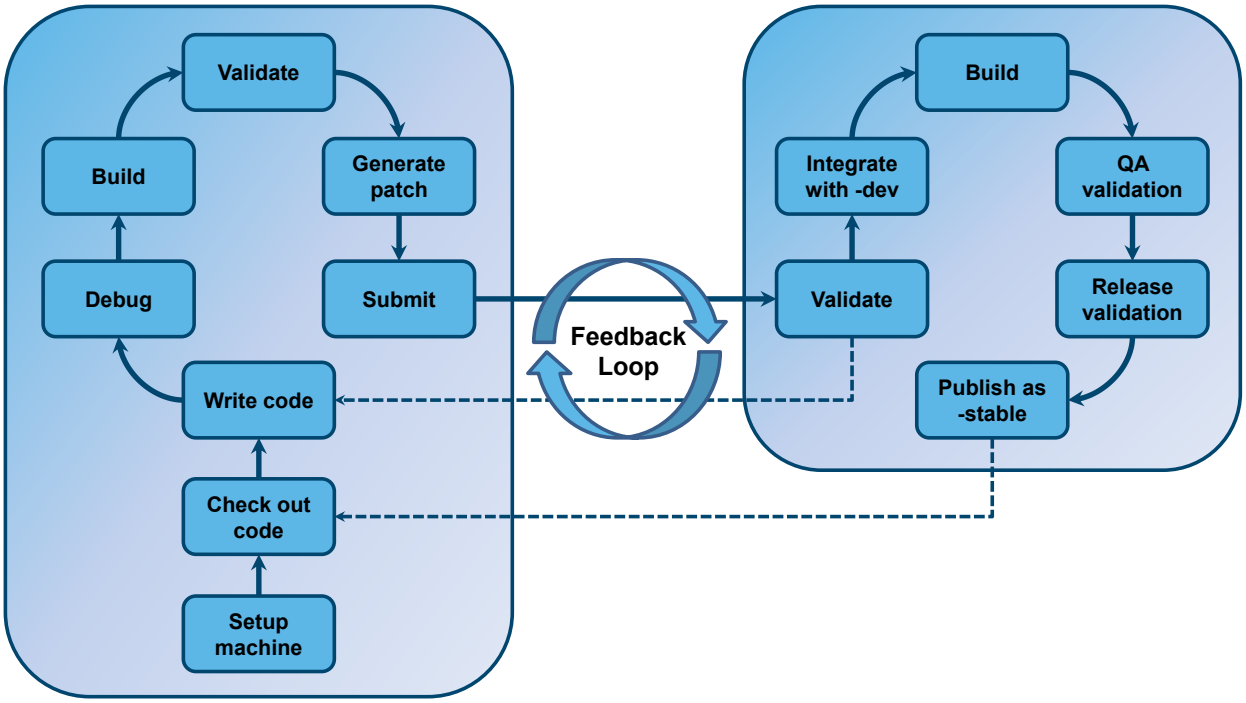

Figure 4: Typical flow of code from individual contributor to mainline version

When the maintainer accepts submitted code, it will then be integrated into his or her development tree. In a large, multi-layer project, the maintainer may then be responsible for submitting it to additional maintainers further up the tree. When the code has been approved by the top-most maintainer, it is integrated for distribution in the mainline release.

Continuous Testing and Integration

Because work may be highly distributed, the open source development model places emphasis upon detecting issues early and fixing them quickly. Many larger projects create nightly and

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

weekly builds using an automated build suite, evaluating new code as soon as possible after integration.

In addition to automated build suites, some projects also create custom test suites to detect functional issues as they occur during active development. These test suites are typically open source as well.

The open source development model favors small, incremental changes, which can make diagnosing build issues, bugs, security holes, and regressions much easier. This ensures that new code does not impact the project's overall focus upon high quality and secure code.

Source Code Release

With few exceptions, projects that use the open source model make both a stable snapshot of the last release and the current development tree available. This helps ensure that users can retrieve the most recent stable release, while developers can work from the most current code.

Release management practices vary from project to project, but most nominate an individual or team to evaluate the maturity of features in the development tree, and monitor QA metrics. When the release criteria are met, this team declares the release to be complete and branches the development tree.

Characteristics of the Development Model

An interwoven development cycle

The open source development model is characterized by a series of interwoven processes that continually improve code quality, instead of a strictly linear progression to a release. Unlike the "big reveal" that typically accompanies the traditional software development model, the open source model encourages continuous and independent feature development. This enables new features to be integrated as they are ready, which in turn allows other developers to build upon them more quickly and produce a more competitive product.

Release Early and Often

"Release early and often" refers to the development practice of publishing alpha code to the development community for review well in advance of the final release. This results in highly iterative development, and minimizes the amount of change between development releases, making regressions and breakages easier to diagnose.

This release philosophy allows for continuous peer review, where all members of the community have the opportunity to comment and offer suggestions and bug fixes. It also encourages small, incremental changes that are easier to understand and test while developers are actively engaged, rather than being discovered during a separate final test cycle.

A side benefit is that the code is frequently reviewed for adherence to coding style, and fragile or inflexible code can be found and improved early in the development cycle.

Peer Review

The open source development process emphasizes peer review throughout the entire development life-cycle. Developers are expected to submit their code to project mailing lists for periodic public peer review, particularly when a feature achieves a development milestone. This helps to ensure that others outside of the development team are aware of the changes, and can

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

provide feedback before the design is final and implementation complete. Other members of the open source project review the code, provide comments and feedback to improve the quality and functionality, and test to catch bugs and provide enhancements as early as possible in the development cycle.

When a feature is complete and ready to be considered for integration, the project maintainer also provides a level of review prior to accepting the code. By the time code is integrated into the main product, it has undergone a number of detailed inspections by others outside of the development team. The result is improved, higher quality code.

Conclusion

This paper discussed major elements of the open source development life-cycle, and described typical characteristics of open source development. The open source development model has proved to be very successful, with hundreds of success stories. This development model has special characteristics that allow faster development by broadly distributed teams, continual and thorough testing, faster innovation, multiple layers of peer review, and total openness and transparency throughout the project.

For more information on getting started, please see the resources below, and learn more on http://www.linuxfoundation.org.

Linux Foundation Resources

Linux Training

The Linux Foundation offers two training courses to enable organizations effectively work with open source developers:

- **LF 205: How to Participate in the Linux Community:** Working with the kernel development community is not particularly hard, but it does require an understanding of how that community works. This course is intended to bring attendees up to speed quickly on how kernel development is done and how to be a part of the process with a minimum of pain and frustration.
- **LF 271: Practical Guide to Open Source Development:** This course prepares organizations to maximize their effectiveness and shorten the time to value when participating in open source development projects. This course builds upon years of best practices and extensive experience in commercial participation in open source projects to help organizations approach the open development model in a structured and methodical manner, maximizing the likelihood of success. The course provides extensive examples from the Linux kernel community, and includes specific best practices for working with upstream.

Linux Foundation Labs

If you have a collaborative software project you need hosted at a neutral party, the Linux Foundation may be able to help. The Linux Foundation assists companies and communities by hosting collaborative software projects.

The Linux Foundation provides three main services to Lab projects:

- The technical, operational and legal infrastructure so that project leaders can focus on technological innovation.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

- Guidance and consulting on open source best practices gleaned from the two decades of experience of Linux and the ability to collaborate and network with the large and growing Linux Foundation community.
- By providing these services to companies and developers, the Linux Foundation provides a much needed framework for advancing and accelerating technology that allows project hosts to focus on innovation.

There are two main criteria that must be met in order for the Linux Foundation to host a lab project:

- Use of open source governance best practices including license and contribution agreement choices in keeping with the ideals of Linux
- Project must either use Linux or have the potential to enhance the Linux ecosystem

If you have a project that may fit this criteria, please contact us:

http://www.linuxfoundation.org/labs.

Open Compliance Program

The Linux Foundation's Open Compliance Program was established to boost adoption of Linux and other open source by making license compliance ever-easier to achieve, to increase awareness and understanding of open source compliance responsibilities, and to make available free resources that can help companies establish their compliance programs. The program offers comprehensive training, compliance educational materials (white papers, compliance blog, webinars), compliance tools, an online compliance community (FOSSBazaar), a best practices checklist, a rapid alert directory of company compliance officers, and SPDXTM, a standard to help companies uniformly tag and report software used in their products.

Events

The Linux Foundation produces a number of technical events around the world that provide a venue to bring together developers to solve problems in a real-time environment.

Publications

The Linux Foundation produces a wide range of publications that are available for free download. These publications are divided into three categories: Open Source Compliance, Workgroups (such as Tizen, OpenMAMA, LSB, SPDX, FOSSology, etc.) and Community. The Linux Foundation publications are available from http://www.linuxfoundation.org/publications.

About the Authors

Ibrahim Haddad, Ph.D., is the Director of Technology and Alliances at The Linux Foundation and Contributing Editor for the Linux Journal.

Brian Warner is Operations Manager at the Linux Foundation.

1796 18th Street, Suite C
San Francisco, CA 94107
+1 415 723 9709
http://www.linuxfoundation.org

THE LINUX FOUNDATION

The Linux Foundation promotes, protects, and advances Linux by providing unified resources and services needed for open source to successfully compete with closed platforms.

To learn more about The Linux Foundation, or any of our other initiatives please visit us at http://www.linuxfoundation.org/.