UNIVERSITY OF EMBU

Software development using Open Source methodologies and tools.

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 way of managing its 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 into 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 the 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 the 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 a further discussion about the need for the feature, the next step is to design and implement it.

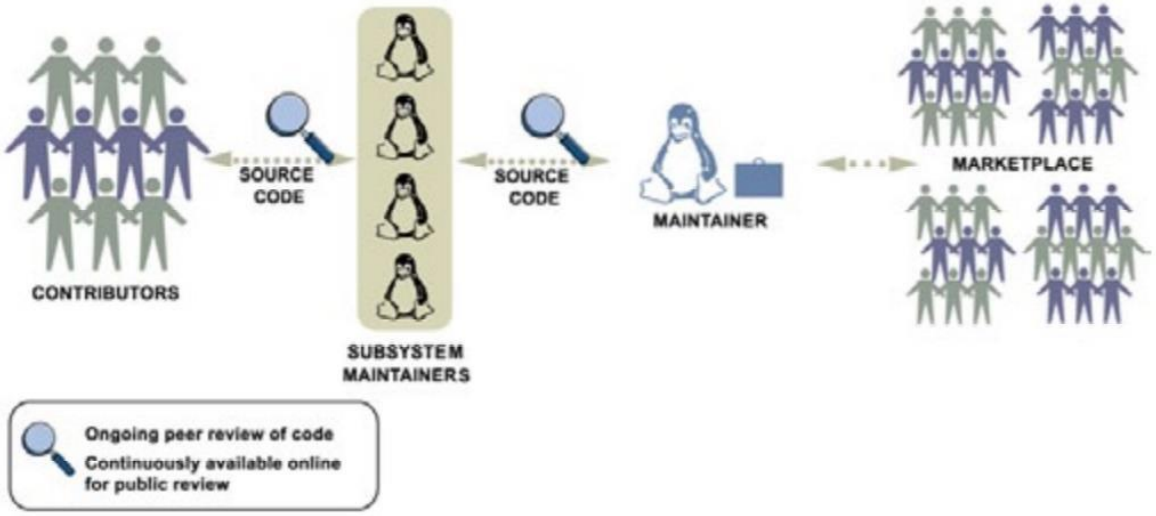

Figure 1: Flow of source code from contributors to the mainline version to the marketplace

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 in 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.

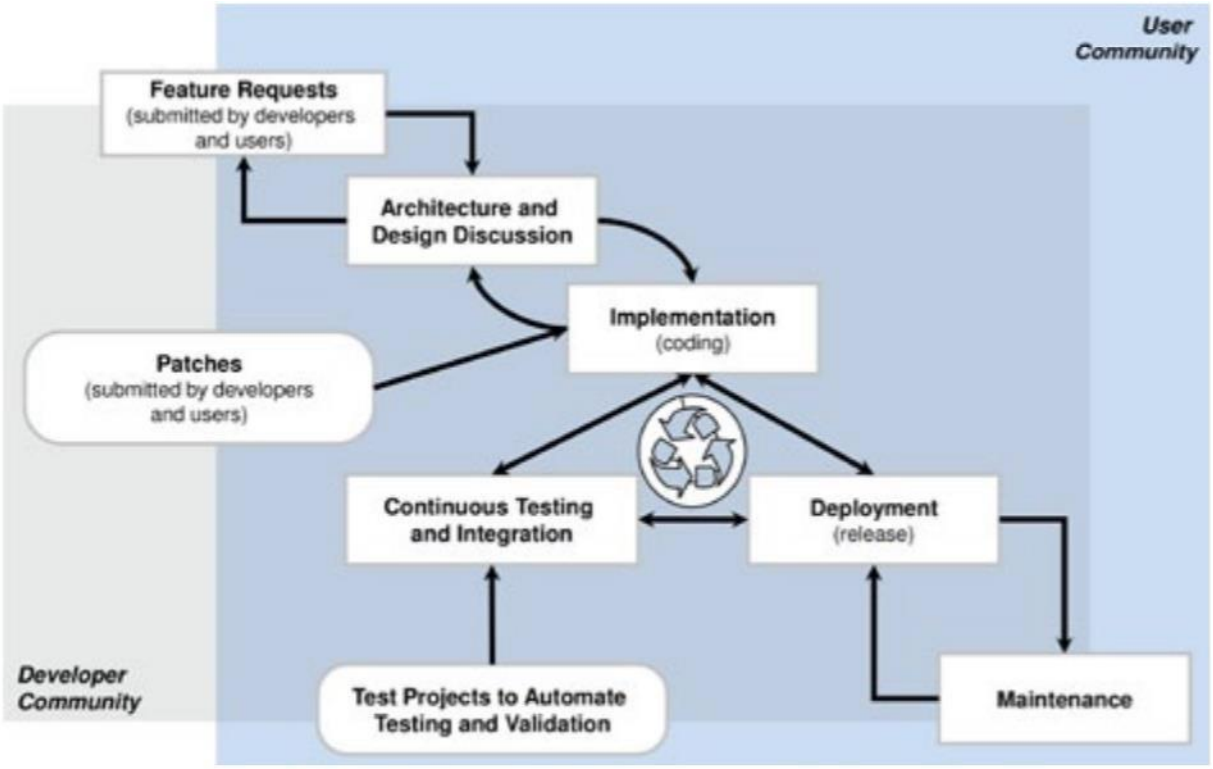

Figure 2: Feature life-cycle in the open source development model

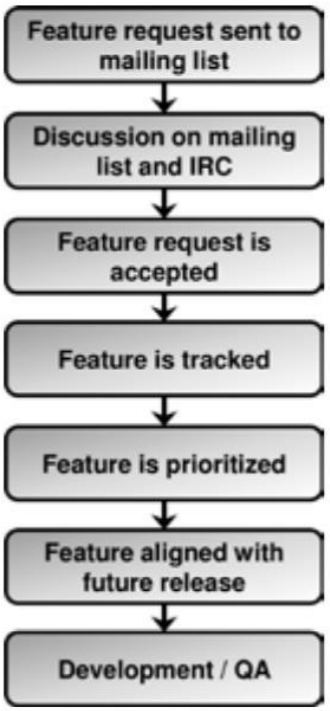

Figure 3: Flow of source code from contributors to the mainline version to the marketplace

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 the 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. The open-source development model places strong emphasis upon collaborative development and peer review, from the 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 communicating their plans. Besides, 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 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 of 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 to submit changes in the smallest increments possible. Smaller patches are less likely to have unintended consequences, and if they do, getting to the root cause of an issue is much easier.

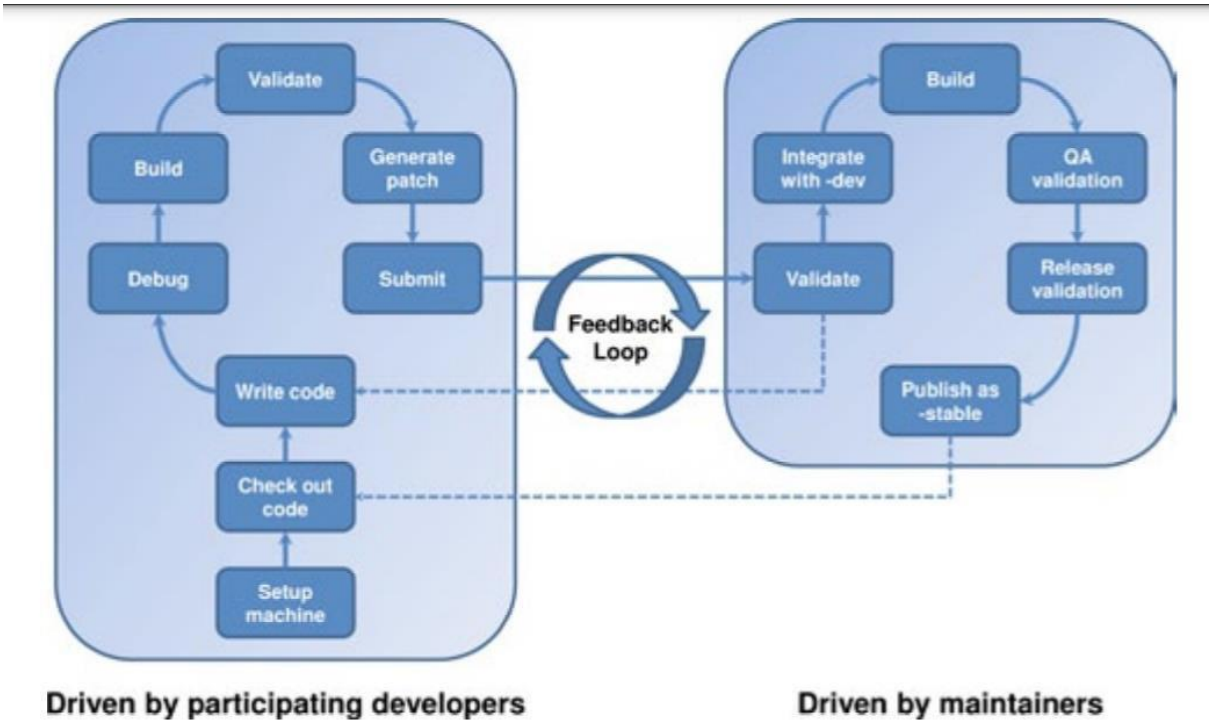

Figure 4: Typical flow of code from individual contributor to mainline version

When the maintainer accepts the 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 emphasizes detecting issues early and fixing them quickly. Many larger projects create nightly and 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.
Open Source and Open Standards.

Open source and open standards are not the same things. Open source refers to software whose source code is freely available to users for reference, debugging, modification, and/or extension.Open standards are, typically, specifications: formal descriptions of software or software

interfaces. Open standards may have reference implementations, but the description in the formal standard typically takes precedence over the behavior of a reference implementation.

Interestingly, the two phrases use the word "open" so differently: For open source, openmeans that the source code must be distributed with every copy of an executable application, and every recipient must be allowed to modify and distribute the source code freely to subsequent users. In open standards, open signifies that the standards process is open to participation and that the completed standards are available to everyone. Working documents and drafts are typically kept private to the issuing organization's members, and there may be reasonable conditions for participation such as membership fees, but any person or company may participate as a member at a meaningful level. Many standards organizations give copies of their standards away for free and the right to implement a standard is typically also free and, if not, is available on fair and equitable terms.

In computing, standards enable portability and interoperability. Portability includes.

i) Application code ports between operating systems or compilers.

ii) Middleware architecture ports between systems--its source code may also port butnot nearly to the same extent.

iii) A developer's skills porting from one platform to another. Each time something ports,

Communication means “transmitting or exchanging through a common system of symbols, signs or behavior.” Standards are a pre-requisite for communication because standardization means “agreeing on a common system.” Geospatial software vendors, developers, and users collaborate in the OGC's consensus process to develop and agree on standards that enable information systems to exchange geospatial information and instructions for geoprocessing. The results of these efforts are Open Standards. The OGC defines Open Standards as standards that are:

- Freely and publicly available – They are available free of charge andunencumbered patents and other intellectual property.
- Nondiscriminatory – They are available to anyone, any organization, anytime,anywhere with no restrictions.
- No license fees - There are no charges at any time for their use.
- Vendor neutral - They are vendor-neutral in terms of their content and implementation concept and do not favor any vendor over another.
- Data neutral – The standards are independent of any data storage model

The consensus group remains in charge of changes and no single entity controls the standard. The OGC's Open Standards are specifications for interfaces and encodings that enable interoperability between geoprocessing systems from different developers, whether employed by

proprietary product vendors, independent integrators, application developers, or active in Open Source projects.

- ✓ A standard is like a blueprint that guides people who build things. A standard documents the use of rules, conditions, guidelines, or characteristics for products or related processes and production methods.

- ✓ Standards can arise from a single company whose successful products become "de facto" standards. Standards can also be set by agreement among two or more software producers, or by government edict or a government-run process, or by representatives from multiple governments.

- ✓ Standards developed in governmental processes, such as those of the International Organisation for Standardisation (ISO) are called "de jure" standards. Alternatively, standards can be developed, as in the OGC, through an inclusive consensus process of the members ordered by well-defined policies and procedures. These are agreed upon by the public and private sector participants in the consensus process, and they typically agree to make their standards freely available in a non-discriminatory manner. Such standards are known as "open" standards.

- ✓ Why do OGC members participate in this process? Some technology providers and users participate because their geoprocessing and geospatial information products and resources are worth more when those products and resources are part of a larger network. Some users participate because they want more choice in their selection of product vendors and better data sharing with other users. They also want to reduce their technology risk. That is, they want to ensure that the systems they buy or build will continue to provide value in years to come. Some solution provider members participate because they want more choice in their selection of platform partners. Some software producers (including Open Source developers) participate because it reduces their development costs and helps them better understand their customers' needs. There are many reasons to participate in a standards consortium like the OGC. This is substantiated by the fact that, with the growth of the Internet and Web, there are more standards consortia than ever before, and a higher percentage of IT stakeholders than ever before are participating in them. Additionally, the OGC's global membership has grown every year since the OGC was founded in 1994 and is currently (May 2011) more than 410 members. The OGC's Open Standards are free, publicly available specifications for interfaces, encodings, and best practices. They are not software.