Is the Open Source Development Model Right for Your Organization?

A roadmap to open source adoption

by Ibrahim Haddad

The open source development model has unique characteristics that make it in some instances a superior model for developing software compared to the traditional software engineering cascade model. As with other practices, the open source development model had its advantages and inconveniences. Will adopting the open source development model improve the way your corporate developers work and produce software? What are the best practices from the open source development model that we can use in a corporate environment?

The open source software development model has a different process and set of values than traditional proprietary software development model. The traditional software development process consists of six activities illustrated in Figure 1: collecting and analyzing requirements, designing a solution approach, developing the code, testing, deploying, and maintaining. After each step is finished, the process proceeds to the next step.

The open source development model has key differences compared to the traditional model of developing software (collect requirements, design, implement, test, release, and maintain).

The open source development model, illustrated in Figure 2, starts with the idea for a new project, a new functionality or capability for an existing open source software component. The next step is to provide a design for the implementation and then a prototype of the capability and translate it from an idea into running software. At the moment the software runs, it's released as a development release, even though it may contain known and unknown bugs. This follows the spirit of release early and release often.

The software will be tested by the community, which discusses the software through mailing lists and discussion boards and provide feedback, bug reports, and fixes through the project mailing list. The feedback is recorded and taken into consideration by project members and maintainers to improve the implementation and then a new development release will be available. This cycle repeats as often as needed until project members feel the implementation is stable enough. When the implementation is released as stable, the development cycle continues with the development release (also called the development tree) until a newer stable release is available.

Some of the unique characteristics of the open source development model include:

- **Bottom up development:** Project members who do the most work get the most say when it comes to making design and implementation decisions. Those who do the most work get the most say. Relationships between developers are very important.
- **"Release early, release often":** Don't wait to have a fully working version to make the code public. This release philosophy allows for peer review, where all members of the community can comment and offer suggestions and bug fixes. It also allows for small incremental changes that are easier to understand and test. Open source projects tend to make a release available early to be used by the user community and then update the release as the software is modified. This practice is described as "release early, release often." The open source community believes that this practice leads to higher-quality software because of peer review and the large base of users who are using and testing the software, accessing the source code, reporting bugs, and contributing fixes. A side benefit of having many people looking at the code is that the code is reviewed for adherence to coding style; fragile or inflexible code can be improved because of these reviews.

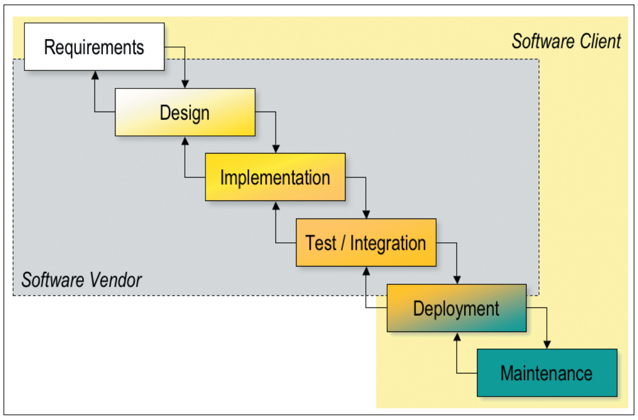

Figure 1: The cascade model of traditional software engineering

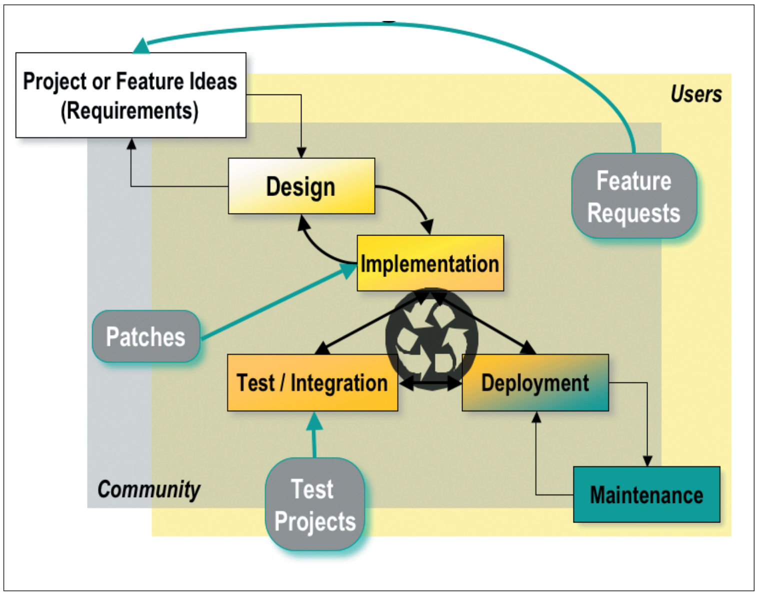

Figure 2: Open source development model

(SOURCE: BILL WEINBERG, OPEN SOURCE DEVELOPMENT LABS, 2006)

- **Peer review:** 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. The result is high-quality code.
- **Small incremental changes:** In open source project development, additional features are often small and non-intrusive and for good reason:
  - It's easier to understand small patches and code changes than big changes in the code or big architectural redesigns.
  - The small changes are important because they help focus the testing phase, which is cyclical and ongoing with every increment of the software.
  - A small change is less like to have unintended consequences.
- **Features that ignore security concerns are flagged:** The open source community takes security very seriously and any development or capability that jeopardizes the security of the software is flagged and not included in the software until the security concern is dealt with.
- **Continuous quality improvement:** This is due to the extensive peer review and quick bug fixes
- **Test projects:** In many cases, test projects are created for large open source projects to create test suites and automate testing.
- **End-user involvement in the entire process:** In Figure 2, we notice that the users are involved in all phases of development in the open source model.

Communication

Open source developers primarily communicate with each other using mailing lists. In the table below, we illustrate some slight differences concerning communication in an open source project compared to a corporate project.

Open Source	Corporate
• Open Source developers are distributed across the world • No face-to-face meetings • No conference calls	• Depending on the size of the company, developers can be in different geographies • Weekly or bi-weekly project reviews to track progress, lead by project managers • High reliance on conference calls and face-to-face meetings
• E-mail is very important as the primary mean of communication between open source project members • Discussions happen on open mailing lists	• E-mail is important • Discussions are mostly face to face and in conference calls • A lot of one-to-one e-mails between project members
• Many open source projects use chat for quick developer and user discussions	• The use of chat software among corporate developers is growing as a cheap way to communicate versus travel for face to face meetings

Many companies are adopting some of the the open source development model has

Project Hierarchy

Open source projects are organized differently than corporate projects. In the table below, we illustrate some key differences between open source projects and corporate projects focusing on project organization and hierarchy.

Open Source	Corporate
• Open source development teams primarily work together in a decentralized fashion with little hierarchy • Hierarchy is loose and flexible • Those who make the most contributions have the most say about the project	• Well structured with defined roles for the project manager, project architect, senior developer, etc.
• There are no formal requirements for joining and no formal rules for participating • The lack of formality doesn't mean that there are no standards for participating or behaving • There are strong unwritten rules that govern all community interactions • Community members are expected to interact respectfully, make reasoned arguments about why a particular course of action is right, and above all, contribute to the community	• There are formal processes to follow when an individual wants to work on a new project • Individuals follow and respect company rules and regulations, and are expected to contribute to the success of the project
• Bottom-up development approach where decisions and power is as close to the bottom as possible (i.e., developers who write the code have a say in the direction of the project)	• Top-down development approach where project management makes the decisions and pushes it down to the implementers
• Meritocracy drives advancement and acceptance • As developers prove their competence and their contributions prove to be valuable to the project, they become more influential	• Corporate adopts specific criteria as part of its performance management
• Open source project members work on a project when, and as much, as they feel like it • Open source project members work on a project until they get bored and loose interest in the project	• Members of a project are fully dedicated to it and must dedicate all their time to the project • Must respect project deadlines and deliverable schedules • Can't stop working on a project without management approval
• Quality levels are often negotiable since the first goal is to provide a working prototype/proof-of-concept, but after several cycles the quality improves tremendously	• Quality is very important and often specific quality goals are request by customers
• The project leader is usually the person who originated the project or the person with the most technical competence and contributions working on the project. • The project leader manages the project by consensus, leading by example • The project leader is responsible for developing a common understanding of what functionally the upcoming release will contain, encourage new developers to join the project, help developers select a portion of the project to work on, and solve any conflicts that arise between team members	• The project leader is usually the manager assigned to the project by management • The project leader is responsible for project requirements, communicating them, assigning developers portion of the work, and resolving conflicts

practices of the open source development; special characteristics that make for faster development, faster testing, higher innovation, peer review, total openness, and transparency

Cultural Differences

Working with the open source community is very different from the traditional corporate development environment and has a different process and set of values from the traditional proprietary development model. In the table below, we illustrate some key cultural differences between an open source development environment and a corporate development environment.

Open Source	Corporate
• Open source developers work on what they find interesting and bring tremendous energy to the project they contribute to	• Corporate developers work on projects they are assigned to
• Open source developers are usually volunteers who donate their time to open source projects that benefit the community as a whole	• Corporate developers are paid to work on company projects
• Motivation for improving and developing a given piece of software is unpredictable. It might vanish or decrease depending on the interest in this piece of software. Release schedules are uncertain.	• Motivation for improving and developing a given piece of software is driven by customer demand
• Open source developers work on features of interest to them. As such, they don't work to meet specific deadlines, but work as long as they're interested in the project.	• Corporate developers are paid by their companies to devote their time to the projects they're assigned to
• Open source developers work in the open with full transparency and extensive peer review of their code • All code developed for the project will be viewed, reviewed, and enhanced	• Development typically takes place in a product group that is often closed and not available to others in the company for cultural reasons and little peer review outside the group that did the development
• Open source developers welcome code contributions written by other developers	• Corporate often suffer from the "not invented here" syndrome in accepting code written by others • Moving from writing propriety code to contributing source code to open source or using code developed by others is a new way of doing things. • Many corporations are developing open source policies and procedures, and creating open source training for their employees
• Open source developers are famous for their code reuse practices and try to avoid doing something twice if it can be automated	• Corporates are encouraging code reuse among their developers in an effort to produce reusable software to help cut their costs
• Open source developers maintain a source code tree that is open and available for all to see and access. They follow the release early release often practice that gives a good estimate of the progress and helps catch bugs early	• Corporate developers follow strict rules when it comes to accessing source code trees and offering stable releases

The Benefits of Adopting Open Source Working Methods

There are several open source development practices that corporates can benefit from adopting in their development environment that can improve code quality, communication, effectiveness, and performance.

- **Using open development methods "à la Sourceforge"**
  - Open source code tree: Make source code available to others to review and offer feedback and suggest improvements (peer review). Inside a company, this lets teams work across organizational lines and lets others add value to the software. Different users tickle different bugs, leading to higher quality. The practice of incrementally adding functions allows for better testing and better chances of capturing bugs. Cooperation is good and benefits all.
  - Open mailing list used for all project-related discussions.
  - Bug tracking systems.
  - Technical support tracking systems.
  - Patch tracking systems.
  - Feature request tracking systems.
- **Fast development cycle with small incremental changes**
  - Adopt the "release early and release often" practice.
  - Go through the cycle several times.
  - Apply small incremental changes in the release to make it easier to understand and test.
  - Faster development builds.
  - Shorter time-to-market.
- **Pay special attention to quality and security**
- **Encourage reuse**
  - Promote and encourage company developers to use open source software and tools in their development environment where it might meet their needs
  - Include open source software in products based on a set of criteria such as technical merits, time-to-market advantage, and avoiding vendor lock-in.
  - Code reuse improves efficiency and increases cost savings.
- **Build reusable software components**
  - Don't keep reinventing the wheel and don't act superior. If someone has already implemented the capability or feature you need, use it, and build on top of it.
  - When you develop from scratch, keep reuse in mind, and develop code in modules that can be used by others and by you for other situations without much modification.
- **Respect and follow community coding style**
  - The open source community follows a strict coding style to make it easier to understand the code, review it, and revise it quickly.
- **Flag problems early and review with the team**
  - Hiding problems or bugs until you come up with a solution isn't encouraged.
  - It's advisable to report bugs or problems when they turn up; the community will help you come up with a workaround or propose and help implement a better solution.
  - Openness and honesty is key.
- **Foster innovation**
  - New ideas have a better chance if engineers can review the source code and experiment with and build proof-of-concept code and test different methods.

Recommended Practices	Description
Increase team communication	Using mailing lists, chat software, wikis
End-user feedback	Involve the end user to get feedback as you proceed
Peer review	Encourage peer review and provide an environment that welcomes feedback and suggestions
Release early and often	Adopt the "release early release often" development practice for the many benefits it offers as compared to the traditional release model, and follow the model of continuous integration and automated test environments
Transparency	Adopt transparency and openness by using open source code trees, bug tracking database, and mailing lists that are open to the whole company.
Good code design	Build a minimal code base and add all the functions and capabilities as separate modules to encourage reuse and ensure easier testing.

Conclusion

The open source development model has proved to be a very successful model with hundreds of open source projects that can be used as a success story. This development model has special characteristics that allow faster development, faster testing, higher innovation, peer review, total openness and transparency. In this article we reviewed the open source development model and compared it to the traditional corporate development model. Many companies are adopting some of the practices of the open source development model for the advantages it offers.

Will these practices be right for your company? You be the judge!

About the Author

Ibrahim Haddad is currently director of embedded & open source technology at Motorola where he is responsible for defining and developing the requirements for Motorola Software Group's open source initiatives. Prior to Motorola, Dr. Haddad managed the carrier grade Linux and mobile Linux initiatives at the Open Source Development Lab (OSDL), which included promoting the development and adoption of Linux and open source software in the communications industry. He is co-author of two books on Red Hat Linux and Fedora, a contributing editor of the Linux Journal, Linux Planet, and Enterprise Open Source Magazine, and a featured speaker and panelist at industry conferences such as Linux World, GlobalComm, Ottawa Linux Symposium, and academic conferences hosted by IEEE, ACM, and USENIX. He got his BSc and MSc in computer science from the Lebanese American University, and his PhD in computer science from Concordia University in Montreal, Canada.