Adopting an Open Source Approach

to Software Development, Distribution, and Licensing

Advantages, Risks, Culture Change, and Strategies

by Ibrahim Haddad

About the Author

Dr. Ibrahim Haddad is the Director of Portfolio Management for the Embedded Systems, Open Source and Linux software technology group at Motorola. In this role he is responsible for defining and developing the requirements for Motorola's open source initiatives. He represents Motorola in Linux and open source forums driving Motorola's interests and overall strategy. He is also involved with customers and partners on matters related to embedded systems and open source software technologies.

ibrahim.haddad@motorola.com

Since the beginning of the software industry, nearly every software company in the world has followed the same business model: its own employees develop the software, which is closely held intellectual property, the software is delivered to clients in binary form, and users run the licensed software on their own computers. Today, this model has been challenged by a new paradigm: open source. Developed and maintained by volunteers across the world, distributed to users at no cost, and available as source code, open source software is radically different from its proprietary counterpart. Each of the new characteristics of open source software forces organizations to develop new ways of thinking about how they procure, implement, test, and deploy it.

Accessible without cost, open source software is created under conditions very different from commercial software and is distributed to users under very different licensing terms from commercial software. Open source software developers take responsibility for the quality of the software. This responsibility demands a new model of software procurement, one where the organization is an active participant in creating the complete software, rather than a passive recipient of what the vendor delivers. The new model demands new working methods and practices. In this article, we'll discuss the open source development model and practices. We'll explore the various benefits and risks the open source model brings to development practices, and possible strategies to support and get involved in open source.

Open Source versus Freeware

It's important to note that open source differs significantly from freeware. Freeware is software distributed without a fee, but without source code access. Freeware creators restrict intellectual property rights to the software slightly and offer the software on a "as is" basis in contrast to open source, which carries less restrictive licensing terms and lets users modify the software product if they want. Freeware is often distributed on a "time-restricted/limited" basis, meaning that it's only free for a while. When time is up, the software stops working. If the user wants to continue using it, he has to pay a licensing fee to disable the time restriction.

Open Source Software

Open source software is software whose source code is freely available to its users. It can be downloaded from the project's Web site and used or modified as desired, as long as its license requirements are observed. Open source software and the open source community have unique characteristics:

- **Zero-price software:** Open source software is distributed at no cost. There's no charge for the source code and there are no licensing fees.
- **A different licensing model:** Like any other software, open source software is distributed under a license that controls how it can be used. However, open source licenses are less restrictive than the licenses on proprietary software in terms of how the software can be used. Typical license conditions include contributing any source code changes back to the main source base and distributing source changes to any customers of the organization that modified the code. The specific conditions depend on the kind of open source license used. It's important to state that open source licenses are written to encourage wide use, with few restrictions put on the software's use. For a list of approved open source licenses, see http://www.opensource.org/licenses.
- **Open source software developers:** Open source developers come from different backgrounds. They are volunteers who donate their time to work on open source projects. The fact that open source software is written by volunteers affects how open source teams are formed and how they work together. Because individuals participate based on their interest in the software, open source management practices are also very different from those in commercial software companies. Open source development teams work together in a decentralized fashion with little hierarchy. The project leader is usually the person who originated the project; he or she must manage by consensus with a lead-by-example approach. The project leader is responsible for developing a common understanding of what functionally the upcoming release will contain, encouraging new developers to join the project, helping developers select a piece of the project to work on, and solving any conflicts that arise between team members. The hierarchy is loose. The best way to describe the hierarchy and leadership is to say, he who writes the code and contributes to the project makes and influences the decisions.
- **The philosophy of community:** The practice of frequent releases to gather user feedback underscores a distinguishing characteristic of the open source world – the community. The open source community refers to groups of contributors and users participating in and organized into a community based on their field of interest. There are no formal requirements for joining and no formal rules for participating. However, the lack of formality doesn't mean that there are no standards for participating or behaving. Unwritten rules govern all community interactions. A community member is expected to interact respectfully, make reasoned arguments about why a particular course of action is right, and, above all, contribute.
- **Development practices:** Open source projects tend to make releases available early to the user community and then update the releases quickly 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 the software, accessing the source code, reporting bugs, and contributing fixes. A side benefit of having many people look at the code is that it's reviewed for adherence to coding standards; fragile or inflexible code can be improved because of this review.

Open Source Development Model

The open source development model is distinctly different from the corporate development model, which follows the traditional waterfall software model of collecting requirements, designing, implementing, testing, and then releasing.

The open source development model, as illustrated in Figure 1, starts with an idea for a new project, new functionality, or new 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 to translate it from an idea into running software. Following the spirit of release early and often, the moment the software

> Organizations that embrace the open source model increase their chance of retaining a competitive advantage

runs, it's released as a development release, even though it may contain known and unknown bugs. The software will be tested by the community, which discusses the software and provides feedback, bug reports and fixes through the project mailing list. The feedback is recorded and taken into consideration to improve the implementation, and then a new development release will be made available. This cycle happens as many times as needed until project members feel the implementation is stable. 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.

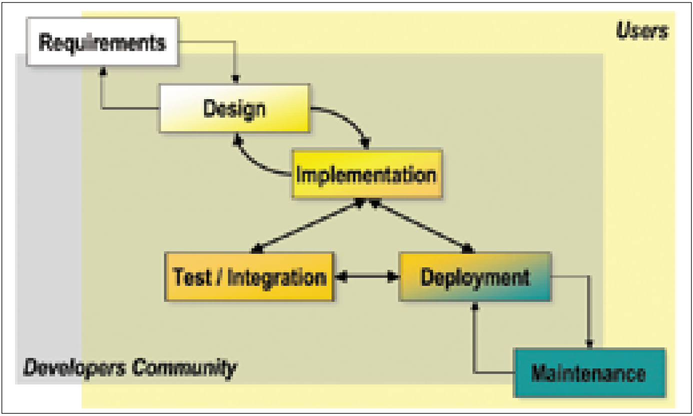

Figure 1: Open source development model (Source: Bill Weinberg, OSDL)

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 in design and implementation decisions.
- **"Release early, release often":** This release philosophy allows for peer review, with all members of the community commenting and offering suggestions and bug fixes. It also allows for small incremental changes that are easier to understand and test.

Open Source: Advantages & Risks

Let's turn briefly to the advantages and risks of using open source software components, and the cultural changes wrought by this model:

Advantages to using open source software components in commercial products

- **Do more with less:** Using open source components, your organization can reduce development costs and time-to-market and focus on the added value it brings to the product. Following this model, you should therefore source enablers and tools as much as possible and focus on your core business.
- **Open source is an additional sourcing channel:** Typically, you develop software yourself within your organization or subcontract it to third party. With open source, you gain an additional sourcing channel.
- **Improved quality by peer review:** The open source community has a special development cycle characterized by early and frequent releases, which facilitates extensive peer review.
- **Using open source for standardization reference implementations:** Open source is a way to launch reference implementations for standardization projects in parallel with our standardization efforts. When it benefits your company, you should promote the development of reference implementations as open source software in various standardization bodies. Open source implementations advance faster than proprietary ones.
- **Access to source code:** Access to the source code lets us modify it and build on top of it to suit our specific needs.
- Undercut the competition: Open source can be used as a way to share the cost of developing software components and reset the competition.
- **Faster path to innovation:** Software components continue to be improved with people contributing fixes, patches, and new functionalities.

Risks of using open source components

Along with advantages come the risks of using open source software components:

- **Requirement to disclose source code:** This depends on which license is attached to the open source software component.
- **IPR responsibility:** With open source software, you're responsible for any IPR-related issues. With third-party software, verifying IPR-related issues are the vendor's responsibility. Problems with intellectual property and software patents can surface because the availability of the source code simplifies the detection of infringements by IP owners.
- **Open source software evaluation:** Like any other third-party software, open source software must go through an evaluation process. If it's open source, its quality isn't necessarily superior. You still need to evaluate the quality and the functionalities provided. There are some quantitative approaches available or in development to assess the maturity of open source software components, including the Open Source Maturity Model (OSMM) and the Business Readiness Rating (a new model for rating open source software).
- **Competing open source projects:** There can be competing open source projects, which means you have to evaluate all of them.
- **No control over project deliverables and deadlines:** The community's motivation for improving and developing a piece of software is unpredictable; it might vanish or at least decrease. Release schedules are uncertain.

Once a company adopts an open source strategy, the next task is to communicate it to the development organization, business units, and technology groups

- **Many developers are working on their own time:** Open source developers will only work on features that interest them, which vary a lot. As such, they don't work to deadlines, but work on a project as long as they're interested in it.

Cultural Changes

Working with the open source community is different from the traditional corporate development environment. It has a process and set of values that is different from traditional proprietary development model.

Corporate Development	Open Source Development
People work to meet project or client requirements	People work on what they find interesting (but, they bring tremendous energy to the project)
People work to meet project deadlines	People work when and as much as they feel like (which can turn out to be quite a lot)
People work on a project until it's completed	People work on a project until they are satisfied with it
People work to meet specific quality goals	Quality goals are sometimes negotiable (but many eyes can make few bugs and the release early and often process helps improve quality)

Furthermore, working with open source and following the open source method will have an impact on the work methods and culture of an organization.

- **Changing working methods:** Working with open source and following the open source development mode impacts the way an organization develops software, moving it from the "Not Invented Here" (NIH) syndrome to using open processes and accepting code written by others.
- **Distributed development:** The open source community doesn't assume that development is done in a single location. Using the same development tools as the open source community will encourage corporate software engineers to work more closely together.
- **Investment versus cost:** Working with the open source community should be perceived as a strategic investment with long-term benefits rather than a cost.
- **Contributing code:** Moving from writing propriety code to contributing source code to open source is a new way of doing things and people should be open to it, accepting criticism and implementing suggestions.
- **Staying competent:** Working on open source projects and being exposed to open technologies is new way of working, being educated and staying competent.

Open Source Strategies

There are many open source strategies that can be adopted and customized to your organization. Below we list three major strategies that are very common.

1) Promote and encourage company developers to use open source software and tools in their development environment when they meet their needs.
2) Include open source software components in commercial products based on a set of criteria such as technical, merit, time-to-market (TTM) advantage, and avoiding vendor lock-in.
3) Contribute source code, initiate projects, and be leader in the open source community. Contributing to open source must be business-driven and decisions to participate and contribute are taken on a case-by-case. Examples of business-driven contributions include providing a reference implementation to establish a project as a de facto industry standard, commoditizing a market, ensuring that new features are of interest to your organization, contributing enablers and software that aren't valuable enough to keep proprietary or as no-margin products.

Once a company adopts an open source strategy, the next task is to communicate it to the development organization, business units, and technology groups.

Conclusions

Open source software is shifting the software industry to a new paradigm, moving from developing code behind closed doors and delivering binaries to customer, to developing code that's shared, modified, and redistributed openly. Key benefits associated with this shift are the reduction in development cost and software component complexity and developing reusable, common off-the-shelf software assets, while increasing flexibility and using common enablers. Organizations that embrace the open source model and follow it when it positively influences their way of building software will increase their chance of retaining a competitive advantage.

Further Readings

- *Eric S. Raymond. The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly. ISBN 1565927249. The book is also available as free download from the author's Web site at http://www.catb.org/~esr/writings/cathedral-bazaar/.*
- *Martin Fink. Business and Economics of Linux and Open Source. Prentice Hall, ISBN 0130476773. http://www.hp.com/hpbooks/strategic/strategic_0130476773.html.*
- *Bernard Golden. Succeeding with Open Source. Addison Wesley Professional. ISBN: 0321268539. http://www.awprofessional.com/title/0321268539.*