

## How Is Open Source Special?

**A** Google search for the phrase *open source* in mid-January 2005 returned approximately 28.8 million Web page hits and 11,000 news articles! A term with such widespread usage can mean a million different things to a million different people, and in this case, most of these people may not be familiar with the software development processes. The technically unsophisticated may, understandably, think of open source as a kind of magical pixie dust that gets sprinkled over software to imbue it with various kinds of goodness. Clearly, this is not the case.

### Commonalities and Differences

Open source software projects involve the production of goods, but in software projects, the “goods” consist of information. The open source model is an alternative to the conventional centralized, command-and-control way in which things are usually made. In contrast, open source projects are genuinely decentralized and transparent. *Transparent* means that all of the information pertaining to what is being made—from the source code to the bug list, the design documents, and the mailing lists in which discussions are held and decisions are often made—is available to anyone who cares to look. It’s this transparency that lowers the barriers to entry and participation.

Years ago, Bill Joy, one of the founders of Sun Microsystems, made a statement that is still true today—that no matter what your company is, most of the smart people in the world work somewhere else. Open source is a way of leveraging those

untapped resources—those smart people. There are “a ha!” moments in open source projects when somebody from Albania finds a bug, submits a patch that makes an enormous difference to the project, and becomes a rising star. Why is the person in Albania participating? The work may be taking a lot of time, and he or she is not getting paid to contribute to the project, but in fact, the person is building reputation capital, and reputation is a significant resource. The interval between emerging as a major contributor on an open source project and receiving a really good job offer is shrinking rapidly. As we increasingly manage software projects on a global basis, we are able to mobilize the smart people of the world in a much more effective fashion. There is enormous leverage in this approach. It means that over time, total costs, including support costs, should become a lot lower.

### The Good, the Bad, and the Community

For me, the good things about open source are summed up in the two-word mantra “Anyone can . . .” In open source projects, anyone can participate, anyone can play a role, anyone can shape a proj-

**The beauty of open source is that should something need to be fixed, replaced, or created, you are not only permitted but encouraged to do that yourself.**

ect. If people don’t like the way the code is going, they can fork the code (create a variant), or they can start their own project. By saying that “anyone can,” I don’t mean that literally. Obviously, one needs to have appropriate technical skills, and there is a whole set of subtle barriers to participation in open source communities. But especially when compared with the way proprietary software projects work, open source is indeed an “anyone can” universe. That’s the big upside. That’s why it’s worth investing in and getting involved in open source.

What’s the downside to open source? Well, there are a bunch of hidden costs. Open source is still a bit like the old Wild

West—few fences, spotty law enforcement, boomtowns, ghost towns, and lots of folks on the move looking for greener pastures. Just because you hear about a project that does something interesting and has a place to download code doesn’t necessarily mean that the software is worthwhile. You have to find out for yourself: Is it finished? Does it work? Is anybody still working on maintaining and improving it? Is it something that has a future?

Another drawback of open source is that there are no guarantees. With pro-

proprietary software there are also no guarantees, but for different reasons. In open source, guarantees are a matter of reputation; however, this can also be a problem. For example, most of the thousands of open source projects listed on SourceForge.net don't yet have a reputation one way or the other. The absence of reputation creates unknowns and increased risks for the users. In the proprietary world, when you find a problem in the software or want something different, you are at the mercy of the vendor to get it fixed. Even though the success of open source is forcing some proprietary vendors to be more open—for example, with shared source code—users generally don't have the option of saying, "Thank you very much, but we want to take this in our own, different direction."

Investing in an open source solution always involves some risk. You may discover that some critical requirement isn't supported and that therefore the cost of your decision could turn out to be expensive in the long run. But the beauty of open source is that should something need to be fixed, replaced, or created, you are not only permitted but encouraged to do that yourself. And you might find out that there are others out there who would like to help. What makes open source software work is the community—the combined and coordinated efforts of dozens, or sometimes hundreds, of independent individuals who come together over the Internet with a common interest and intent. On the other hand, sometimes it is this very same group dynamic that can cause a project to fail.

### **What Open Source Needs to Succeed**

For open source projects to succeed, they need to have the right technical infrastructure and the right community infrastructure, as well as shared values.

#### *Technical Infrastructure*

The technical infrastructure is a bit easier to describe than the community infrastructure. What does it take to support a myriad of geographically separated individuals trying to work collaboratively on a single project?

Some of the basic project infrastructure issues to consider are the following:

- Is there a publicly available source code repository?
- Is there a public bug-tracking system?
- What are the communication vehicles that the project uses—a mailing list, a wiki, IRC? Do these help people who want to participate?
- Are the tools used in the project suitable for getting the project done?

Good projects have at least an adequate technical infrastructure, because that is what the work depends on. Without an adequate technical infrastructure, the work doesn't get done.

#### *Community Infrastructure*

As important as, or more important than, the technical infrastructure is the community infrastructure. There is generally some sort of formal governance of projects: some are foundations, some are not incorporated at all, and some fall in the wide variety in between. Then come the details of how the project is managed on a day-to-day basis:

- How does the community work?
- How do decisions get made?
- What are the power relationships?

Finally, an important aspect of the community infrastructure is the particular licensing strategy for the project: the terms by which the software is made available and the intellectual rights of the contributors to the project are managed. The spectrum of licensing options is so broad that a whole conference could be held just on open source licensing.

#### *Open Source Values*

In my experience, there are two very important issues that have received less attention but that are key to what makes these communities successful. First, does the project community have clear values and principles? And second, does the community have practices that integrate those principles through the use of tools?

Simply saying, "We're going to do this open source" can get you in the door, but

there is a lot you have to do and think about to make it happen. In successful projects, the participants share a set of values. How those values are reflected in the day-to-day operation of the project varies widely from project to project, but within a project is a kind of coherence. It is necessary to have a good match between the values and principles of a project and the tools used to get the work done.

When I look at open source projects, I ask several questions: How easy is it for new people to get involved with this project? Are there big barriers? Is there too much to learn? Is the project unfriendly, or are there easier ways to get started? Is there a strategy for community growth? And most important, are project decisions made in a way that people regard as being fair? Groups reach decisions in many ways. It doesn't seem to matter if the project is run as a dictatorship or a commune: the good projects are the ones in which, most of the time, most of the people think the decisions are fair. And, by the way, decisions *do* have to be made. Projects in which decisions aren't made don't produce anything.

### **Conclusion**

Open source is a fertile area for study, both from a practical understanding of how to make these projects work well and from a more formal, academic consideration. I believe the leverage of open source, being fundamentally a more efficient as well as democratic way of developing software, can offer great advantages. But the academic world needs to get more involved in open source, to get more familiar with its mechanisms—how it works and how it doesn't work. I think this kind of research will benefit not only the academic world but open source in general.

Mitchell Kapor, the founder of Lotus Development Corp. and co-founder of the Electronic Frontier Foundation, is President and Chair of the Open Source Applications Foundation (<http://www.osafoundation.org>), a non-profit organization he founded in 2001 to promote the development and acceptance of high-quality application software developed and distributed using open source methods and licenses.

