Open Source: Beyond the Fairytales

Richard P. Gabriel & Ron Goldman

Open source is a software development model that to many is synonymous with a free and free-wheeling operating system juggernaut destined to upend current computer markets. IBM is embracing Linux to unify its hardware product line, and Microsoft is screaming that open source is "anti-American and destructive of intellectual property." Analysts and executive have been gathering around open source to find out what it's all about, and people like Eric Raymond and Linus Torvalds—once indistinguishable from society's stereotype of the elfin programmers working in the dark on the software that powers the digital age—have become heroes, spokesmen, and leaders in the new age of anti-monopolistic software.

But is this narrow view adequate? Many other companies such as Sun Microsystems Inc. have embraced open source but offer neither Linux nor Apache nor any other of the open-source icons in their product lines. What are they doing? Open source, for corporations, has more to do with a particular business strategy than it does with any specific technology. In fact, technology is just another piece of the puzzle for a successful technology company—certainly it's a necessary piece, but often it's not the distinguishing piece. Apple Computer, for example, is viewed as an innovative technology company but it actually competes on industrial design, habitable human interfaces, image, and lifestyle while its technology is largely borrowed (Power PC CPUs, BSD Unix, Sony and Samsung flat screens, etc) or a bit old-fashioned (Mac OS 9.x, for example).

Innovation Happens Elsewhere by Richard P. Gabriel

We call the strategy "Innovation Happens Elsewhere." But before we talk about that, let's look at some of the realities of doing open source as a company, and that way we'll be able to better understand the strategy behind it.

Everyone together is smarter than your group alone

A myth of open source is that by developing software this way, a company can reduce its costs because outsiders will be writing some of the code, and the code will be developed faster because of the increased workforce. In fact, a typical open-source project attracts relatively few outside developers.

Inviting people to look at source code puts them in a frame of mind to help out. Not only will a project gain a lot of testers, but these testers will have the source code and many of them will be able to locate the source of a problem in the code itself. The sum total of people running the software will encompass a wide variety of experience and expertise, and it is not unreasonable to assume that every type of bug in the code has been seen by someone in the pool at one time or another. It's like having a thousand people proofread a book for spelling and typographic errors.

These will be mostly highly informed and dedicated testers. The fact the source code is available means that many of them will be at least able to develop their own workarounds to problems, and so they will stick with the software longer. And some of them will want to contribute, sometimes substantially, to the core source code.

Open source is a gift economy

To understand open source it helps to make a distinction between a *commodity economy,* to which we are accustomed in a capitalist society, and a *gift economy.* In a gift economy, gifts are exchanged, forming a bond based on mutual obligation: In the simplest form of gift exchange, when one person gives a gift to another, the receiver becomes obligated to the giver, but not in a purely mercenary way—rather, the recipient becomes very much like a member of the giver's family where mutual obligations are many, varied, and long lasting. A person may give a gift with the realistic expectation that someday a gift of equal or greater use value will be received, or that the recipient will pass on a further gift. In an open-source project, the gift of source code is reciprocated by suggestions, bug reports, debugging, hard work, praise, and more source code.

The commodity economy depends on scarcity. Its most famous law is that of "diminishing returns," whose working requires a fixed supply. Scarcity of material or scarcity of competitors makes high profit margins. It works through competition.

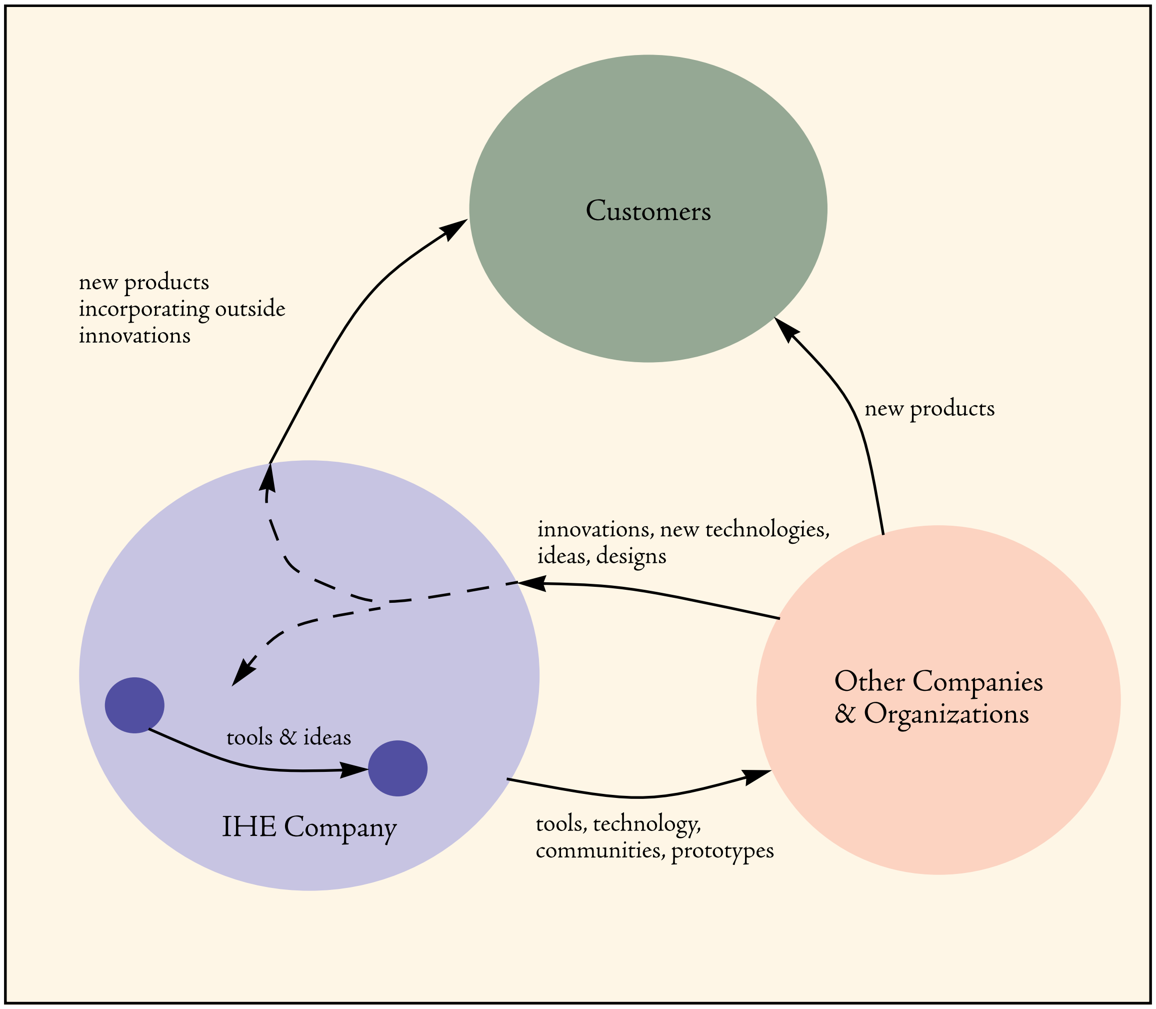
The gift economy is an economy of abundance—the gifts exchanged are inexhaustible. Gift economies are embedded within non-economic institutions like kinship, marriage, hospitality, artistic patronage, and ritual friendship. A healthy Western family operates on a gift economy. In an open source project an individual's status and reputation depends on the quality of the gifts they contribute.

Unless a company realizes and is able to work with the gift-nature of open source, any adventure in building an open source community will likely end badly.

Innovation Happens Elsewhere

Most companies are not large enough to influence the direction of their own markets, and few companies are able to design products that truly serve their customers. Most companies find they need to fine-tune designs and products over a series of releases. Some companies use time to introduce products: the Saturn car company started by introducing inexpensive but high-value cars which appealed to young adults just starting out in their careers, and while learning the tastes and values of that generation, Saturn has introduced new models that reflect the increasing affluence of its core customer base. This is an example of using the world outside the corporation as a source of innovation. Software companies using open source can perhaps exploit this strategy in its most pure form. So, what is the strategy?

To simplify the discussion, let's consider only companies that produce technology products. The Innovation Happens Elsewhere (IHE) strategy begins by recognizing where the company's proprietary value lies. Everything outside this inner circle of protected ideas and technology is available for instigating outside innovation beneficial to the company. The primary goal of the strategy is to increase the number of potential customers—that is, the size of the market available to the company.


Customers
new products incorporating outside innovations
new products
innovations, new technologies, ideas, designs
tools & ideas
IHE Company
Other Companies & Organizations
tools, technology, communities, prototypes

To do this, the IHE company tries to create more products in the market that either are enablers for the products the IHE company sells or form an aftermarket for them. Rather than trying to accomplish this alone, the IHE company tries to encourage other companies or organizations to do this work—but for their own purposes.

The impetus for companies to do this comes from tools, technology, communities, and prototypes that the IHE company provides. By opening up part of itself to the outside, the IHE company can provide gifts that trigger the gift-economy effect, technology; tools, and prototypes that are of high value to outside companies and organizations and that trigger them to work in areas important to the IHE company; and communities where others can work within a culture that supports the vision of the IHE company. Let's break this down a little.

Giving gifts of technology and tools initially spurs outside individuals to work on or with those gifts for their own benefit—perhaps one of the tools is useful to a software developer for one of his or her home projects. Later, that developer might bring the tool into his or her company, which starts to use it for their own purposes. Because the tool was a gift (and perhaps its source is available), individuals and their companies send bug fixes, extensions, modules, and ideas back to the IHE company. The gift has been picked up—for selfish purposes—but it stimulates a gift in turn of work that is of value to the IHE company directly, which takes those tools, technologies, prototypes, and ideas back and uses them to enhance its own products. In the open-source realm, this is the fundamental expected payback for opening up software source. For the IHE company, this is only the beginning,

Next, an outside company using gifts from the IHE company, recognizes that these tools, technologies, and prototypes can be combined with their own company's technology to produce products at lower cost, less risk, and of great value to their customer base. Again, this is done for selfish reasons, but such outside companies are, by doing this, contributing to expanding the customer base for the IHE company. Sometimes a company or organization will think of an application or variation of the gift that will open it up to entirely and unexpected markets. This is the sort of return gift the IHE strategy can provide.

Even further, the IHE company, by creating, building, and maintaining a set of communities around these gifts, can engage in serious conversations with outside companies, organizations, and individuals. Such conversations not only can improve the IHE company's products, designs, and directions, but also provides the IHE company with an opportunity to demonstrate leadership and vision, thereby putting it in a position to strongly influence the direction and structure of the competitive landscape. Because this is done in a context of gift-giving, culture exchange, and conversations, this is leadership and seduction, not control and power. If the IHE company exhibits learning in response to these communities, other members will continue to give gifts, work with the company, and help it maneuver through difficult competitive and market situations.

Communities are also a way to tell stories to the community while minimizing the effects of cynical write off. When a company puts out white papers, advertising, and other public relations materials, the audience is likely to discount as hype statements made there. But with a community, where developers and engineers are talking to each other, the messages can be both smaller—too small for press releases—and larger: global visions and proposals for new directions couched in design and other engineering statements. By valuing the comments and ideas from outside community members—by showing respect—the IHE company can create a channel unlike any other.

The ability to communicate through communities like this means that the IHE company can nudge the market in directions that play to its strengths. And by demonstrating both great innovation and respecting the innovation of others and by providing just the right gifts, the company can increase the size of its market and tune itself and its informal "partners" to that market, which is also tuned through the vision of the company. That is, the market, the IHE company, and companies in the innovation virtuous cycle co-evolve under the partial direction or influence from the IHE company.

The IHE company takes advantage directly of the returned gifts from others, both by using them in its own products or by using them to help its internal operations. The IHE company puts itself in the best position by embracing at all levels the philosophy of "Innovation Happens Elsewhere," and when people in the company are always looking inside and outside the company for innovation they and their groups can use.

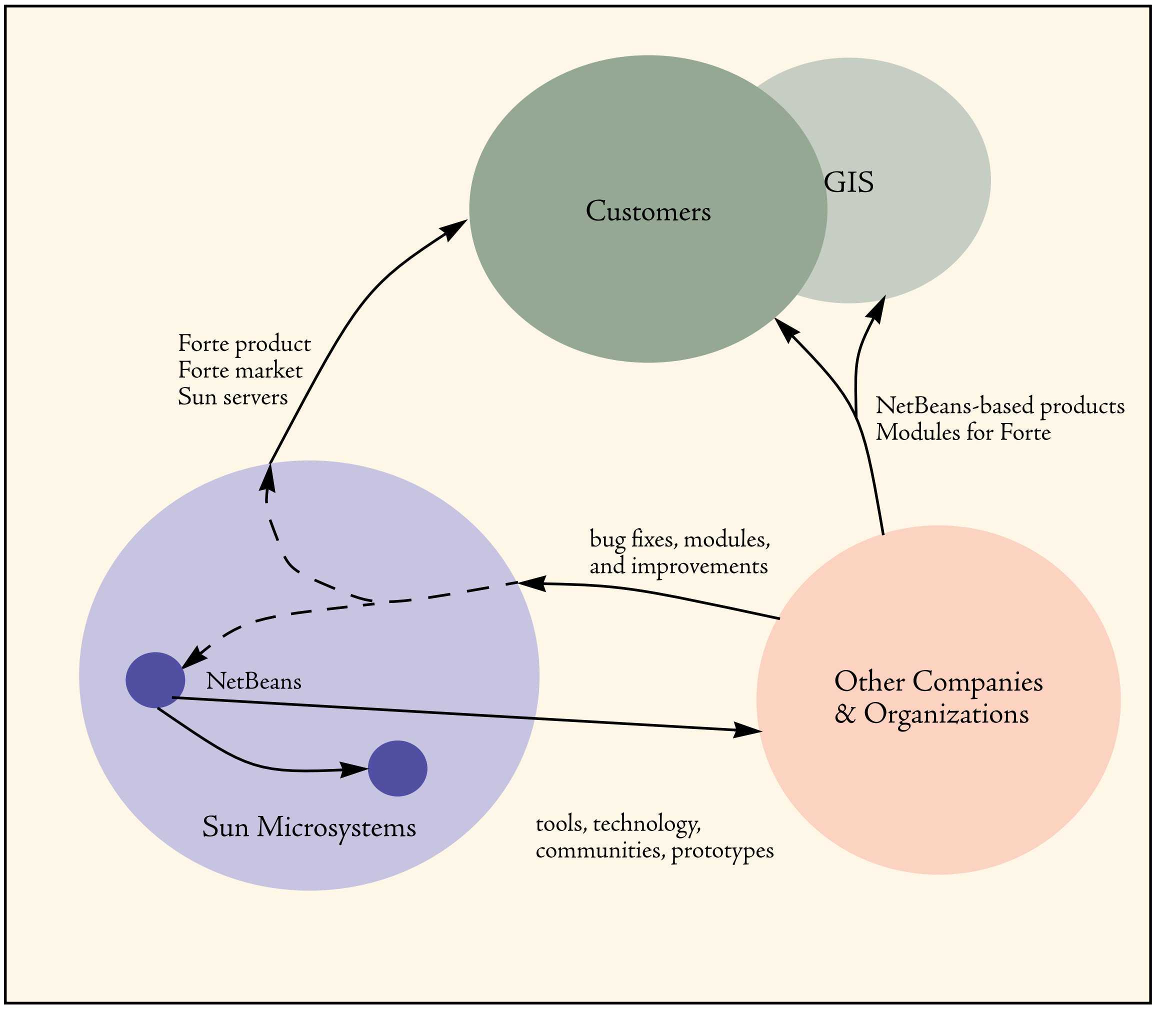
There are several keys to using this strategy. The first key is understanding and isolating the true proprietary value and technologies of the company. The more accurately this is done, the better able is the company to use its gifts and thrive. If what's proprietary and valuable to customers is too small, the company will have a hard time surviving as others crowd into its space. If what's judged proprietary and of value is too large—or too much of what the company makes or works on—then there will be few gifts available to get the cycle of innovation going.

The second key is the confidence that your organization can engineer products quickly and well enough to stay ahead of competition. Microsoft, for example, does not believe it has enough of an engineering edge over competitors to do other than hold proprietary all its source code. This is why it tries to combat open source rather than embrace it.

The third key is a company culture that can embrace and celebrate innovations wherever they occur. In a sense, this is confidence and enthusiasm for ideas, but it is also respect and the right balance of pride and humility. Some organizations seem to fear ideas that originate from outside. Such organizations cannot use Innovation Happens Elsewhere.

Open Source and "Innovation Happens Elsewhere"

Companies that truck in software can use open source as a basis for sharing tools, technology, and prototypes, and communities can be built around open source projects. In this case, it's important to recognize both that such communities need to succeed as software development efforts and that the goal of such communities is to fuel the IHE feedback cycle.



Sun Microsystems' NetBeans open source project is a good example of a community built to support the IHE strategy. NetBeans is an open-source platform for building an integrated development environment (IDE), specifically for the Java language. Sun gets improvements from the open source community, and builds a proprietary version of NetBeans—called Forte for Java—which is a tested version of NetBeans extended with proprietary modules. Other companies use the NetBeans technology in-house and as part of other products. Forte for Java is additionally positioned as a platform on which other companies can sell plug-in modules—that is, Forte for Java is the basis of a marketplace.

The effect is to increase the population of Java language programmers by both providing tools for those developers directly and through other companies building on the NetBeans platform. This way, not only is Sun Microsystems building the Java community, but so are other companies and individuals in the NetBeans community. Sun sells an IDE derived from NetBeans, but most importantly, Sun sells server hardware that runs Java particularly well. Further, Java developers within Sun use NetBeans and Forte for their own work.

An unexpected innovation happened when a group removed the Java specific modules from the NetBeans IDE and replaced them with mapping, visualization and analysis modules in order to build a modular, environment for spatial analysis and visualization. This potentially opens up a future geographic information systems market to Sun, a market never contemplated.

Lessons

There are other advantages to using open source, but being able to engage the Innovation Happens Elsewhere strategy is certainly the most compelling. Not only does it build the market size, but the tactics to make it work help companies that use it get to better products faster by involving directly its customer base in their design. Through direct conversations with customers, there is reduced guesswork in gathering market requirements. Moreover, there are some operational efficiencies to be gained through better testing, community-based support, and some significant product contributions.

Sometimes—no, actually usually—a surprise will happen, an innovation or application of the ideas and technology will come along that you never dreamed of, never heard of, couldn't imagine—done by a group or individual you never heard of. And it could be a pivotal market for you or could change how you view and develop your original technology.

What is most stunning, though, is the difference in the feel of companies that truly engage with their communities of customers, partners, and competitors. Morale is boosted, progress is constant, and it simply feels like something good is always happening.

Looking Beyond the Code by Ron Goldman

There are many definitions of what constitutes open source or, as it was originally called, free software. The basic idea is very simple: By making the source code for a piece of software available to all, any programmer can modify the software to better suit his or her needs and redistribute the improved version to other users. By working together, a community of both users and developers can improve the functionality and quality of the software.

Thus to be open source requires that anyone can get and modify the source code, and that they can freely redistribute any derived works they create from it. The different licenses have various wrinkles on whether modifications must also be made into open source or if they can be kept proprietary.

The success of a number of highly visible open source projects, such as Linux, Apache and Mozilla, has caused many people to take a more detailed look at open source and how it works. But call it "open source" or "free software" the focus is the same: on the computer source code used to create useful computer applications.

Traditional thinking about open source

If you ask almost anyone connected with open source about how it works, they will draw something like the following figure:

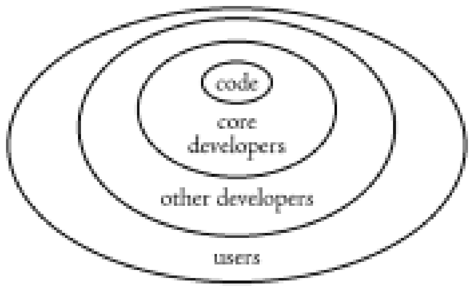


Figure 1: Single community built around source code

They will tell you about how people start as just users and how some will become more involved by reporting bugs. Some may become developers who fix bugs or make minor enhancements. A few developers then get more involved and join the ranks of the core developers, being granted the right to check in changes to the actual project's source code. This is a very code-centric view of the open source process as a hierarchy that has users at the periphery, occasional developers closer in, core developers even closer in, and the code at the center.

And in fact, this view makes the hierarchy into a funnel in which the goal is to convert people from one ring in the diagram to people in the next inner one—almost as if the goal were to turn people into code in the end, the highest form of existence.

A direct result of this perspective is that the actual users of the program are marginalized. While the success of the project is often measured by the number of people that use the computer program being developed, only those people who are willing and able to talk about the code can participate in discussions on the project's mailing lists. Major decisions are made by those writing the code, not by those who will only use it.

Taking a closer look at what is actually going on

If we step back a bit we can see that the people involved in an open source project take on various roles. Over time they may act as a user, a designer or a coder. So a better diagram would look like this:

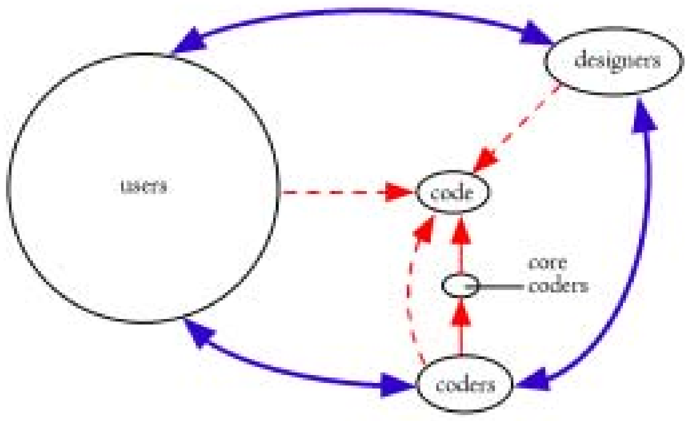


Figure 2: Still focused on code

Here the code is still at the center; users, designers, and coders can look at the code; but only so-called *core coders* deal directly with the code. We use the term *coder* rather than *developer* to emphasize the roles being played: A coder is someone who manipulates code, either by writing new code to implement some design feature or by inspecting and correcting existing code someone else has written.

The thin, solid red lines indicate changes to the source code, the dotted lines indicate viewing the source code and interacting with the program, and the thicker, blue lines indicate communications between people in the different roles. Note that the ovals representing the different roles are not drawn to scale; the users circle should be much, much bigger for any healthy open source project.

This is still a view that focuses on the source code, but it brings out that design is separate from coding and that designers need not be coders. It also reflects the fact that people in the community may not simply advance from user to core developer in a linear progression, but adopt roles that make sense for what they are doing at the moment.

If we step back still further and look at all the ways that people interact in the context of an open source project we see they do so in many different ways, such as:

- writing, modifying and debugging the actual source code
- reporting and commenting on bugs
- writing and reading documents such as user manuals, tutorials, how to's, system documentation, etc.
- online public discussions via mailing lists, newsgroups, IRC chat sessions, etc.
- visiting the project's official web site or other related web sites

- attending meetings and conferences such as user groups, community meetings, working groups, etc.

Many of the above do not involve the source code at all: Users can discuss how best to do their jobs, maybe only just touching on how the project's tools can help them. Other conversations may focus on standards for protocols or APIs that the project uses. Still other conversations may address issues affecting the larger open source community. Each different conversation involves a different group of people. These groups may range in size from small working groups of less than twenty members to full communities of hundreds or thousands of participating individuals. We might represent these multiple communities like this:

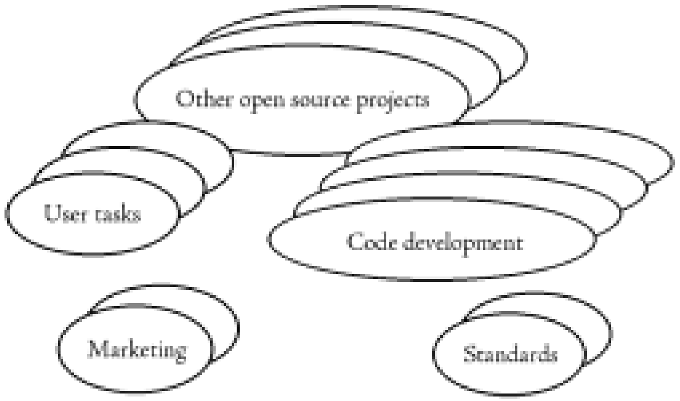


Figure 3: Communities built around common interests

Some people will participate in different discussions, making them a member of several communities. This overlap helps to create the larger community associated with the open source project as a whole.

Each of these communities has its own special interests, for example some communities connected to Linux might include system administrators concerned with installing and configuring the Linux operating system on various types of computers. Another group of people might be concerned with business productivity tools (such as word processors, spreadsheets and presentation tools) that are available for Linux—their focus is on what tools exist and how to use them. A third community might form around computer games available for Linux, with a subcommunity around a specific game like Quake—they would focus on exchanging tips, rumors of new games, and finding opponents to play with.

Each community will flourish or wither depending on how well their interests are met by the community resources. For example a community of newbies asking basic questions about how to use a piece of software will only succeed if more experienced users that can answer those questions are also part of the community.

In the course of a successful open source project different communities will come and go. New ones will spring up, grow and possibly become dormant or die. As long as there are always some thriving communities, the larger open source project can be considered alive and well.

Note that death of a community does not equal failure. Consider a community that arose to develop a new standard. After the standard they developed is accepted by the larger internet community, the community would have achieved its purpose and no longer be necessary. If in the future revisions to the standard are called for the community might be resurrected.

Examples of Multiple Communities

To make this more concrete let's look at some of the different communities that exist within two Sun sponsored projects, Jini and NetBeans.

Jini

Jini technology is a simple distributed computing model based on the Java programming language developed by Sun. Among other things, it was intended as a model for services—small bits of functionality—to discover each other dynamically and create a network of interoperating program parts. These services could be housed within devices—physically separate computing platforms as far as Jini is concerned.

For Jini to succeed, it was clear that the underlying Jini protocols and infrastructure would need to become pervasive, and to accomplish that would require a strong community of participants and partners. Moreover Sun did not have the expertise to define Jini services in areas like printing, digital photography, storage, white goods, or the many other potential markets for products that would benefit from being network enabled.

As a preliminary step the Sun Community Source License (SCSL) was developed to provide a framework where the source code for Jini could be safely shared. SCSL is not true "open source" but has many of the same characteristics.

Within the Jini Community that has developed since the core code was released there are many separate communities focused on creating Jini services in different application areas:

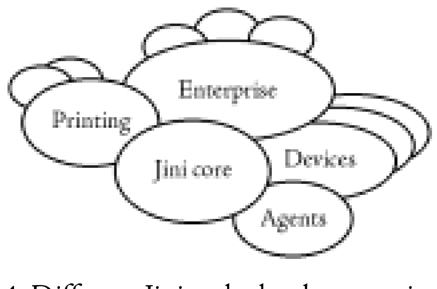


Figure 4: Different Jini code development interests

A small number of developers care about further developing the core Jini code. Others care only about areas like printing or enterprise applications or building networked appliances that use Jini to connect to services. People working in one application area often have little in common with those working in another area. For example, those people working on creating services for appliances in the home have a very different set of concerns from those using Jini to connect legacy enterprise applications.

A similar situation exists in any large open source project such as Linux or Apache. In Apache, for example, there are smaller sub-communities focused on the Apache HTTP Server, the Apache Portable Runtime, the Jakarta Project (which includes major efforts like Tomcat, Struts, and Ant), mod_perl, PHP, TCL, and XML (with subprojects like Xerces, Xalan, Batik and Crimson).

In addition to communities focused on developing code, other Jini related groups have formed around interests like helping beginners learn Jini, Jini in academia, and, even at one point, Jini marketing & business development. A recent check of the jini.org website revealed over 85 current projects, some of which were no longer active.

NetBeans

NetBeans is a modular, standards-based integrated development environment (IDE) that software developers can use to create and modify applications written in Java. It supports a wide range of features from JSP development and debugging, to integrated CVS support and beyond. All of these pieces of functionality are implemented in the form of modules that plug into the NetBeans core. As an open source project, the basic version of NetBeans is available free to anyone wishing to use it. Sun's main interest in NetBeans is to bring developers to the Java platform and make them successful.

Let's look at the various communities involved with the NetBeans project:

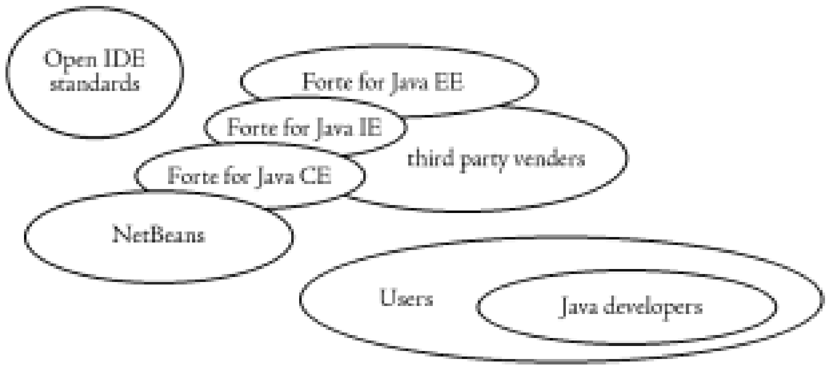


Figure 5: Multiple NetBeans products and uses

Besides those directly involved with developing the NetBeans source code, there are three additional groups at Sun working on various editions of Forte for Java: *Community Edition* (CE), *Internet Edition* (IE) and *Enterprise Edition* (EE). The latter two editions are for sale products that include additional proprietary modules. There are also various third party companies developing NetBeans modules that are also for sale.

There are several distinct *user* communities. First there are those developers using NetBeans as an IDE to create Java applications—their concern is how to best use NetBeans to write Java programs. There is also a geographical information system (GIS) built using the NetBeans framework, so users of that application are not involved with any programming; they want to discuss issues around using

GIS. When the NetBeans IDE is enhanced to handle languages besides Java, like C/C++, then there will be a new user community of C/C++ developers.

Note that in the above diagram the size of the oval labelled *Users* should be several orders of magnitude greater than all of the other ovals; the users number in the tens or hundreds of thousands, while the people modifying the NetBeans code is in the hundreds.

For NetBeans to be a success many of the communities shown above need to be successful. They must grow for NetBeans to prosper.

Other projects

Note that the "users" for both Jini and NetBeans are still mainly programmers. If we look at a project such as OpenOffice that is aimed at creating an office suite for anyone to use, then the diversity of the communities becomes even more clear. OpenOffice has groups focused on creating better user documentation, marketing and promoting OpenOffice, and usability issues. The basic community of people using OpenOffice has split to include new groups for French and German native speakers.

Implications for the Future of Open Source

An open source project has many different communities, each with a different purpose. For a successful project, each of these communities must be nurtured and helped to grow. In a successful community a vocabulary might spring up which is derived from the project's technology, application area, and existing culture. Such a community will come to resemble a long-existing club with its own phrases, in jokes, rituals, and customs—an astute creator of such a community will know this, and will help the community grow these aspects. One less astute will focus on the code, probably leaving out vital potential members of the community.

One way to make the focus go beyond the code is to actively make roles for non-developers like user interface (UI) designers and documentation writers. For example, NetBeans has a community-wide mailing list dedicated to the design and discussion of UI issues. There is also a NetBeans project focusing on UI development and issues, plus a process for other developers to get UI design assistance from the members of the UI project. When the NetBeans project first started there was a hesitancy to add non-developer roles like this because that wasn't something that the high profile open source projects like Apache or Linux did. Now the community values the work of the UI group. A similar example is the recent addition of a Usability group as part of the GNOME project—their work has been welcomed by the larger GNOME community as something long needed.

Learning from Open Source

For companies involved with open source projects, once they realize that all of these other communities exist then they can consider business reasons to interact with them—informal channels to customers, collaborations, etc. By cultivating new communities around user interests such companies can work to ensure the success and growth of the underlying open source project.

Companies can also apply this to their regular product offerings by recognizing and fostering the development of communities around those products. By helping to make these communities successful the company can increase the success of its products. These communities might include other companies, even some with competitive products, but making the community successful should result in growing the size of the market benefiting everyone.