PART I

Overview of Open Source

Golden.book Page 3 Friday, July 16, 2004 11:03 AM

1

The Source of Open Source

Executive Summary

Every good newspaper story starts with these critical questions: who, what, when, where, why, and how. By answering these questions right up front, the reporter enables readers to comprehend the important facts and implications of an issue quickly and incisively. This chapter uses the practices of journalism to provide a quick overview of open source software. It addresses each of the questions and offers a speedy introduction to what is perhaps the biggest sea change in the software industry since its beginnings more than 40 years ago.

This chapter answers the who, what, when, where, why, and how of open source.

Since those beginnings, nearly every software company in the world has followed the same business archetype: closely held intellectual property, developed by the company's own employees, delivered in binary format, licensed to users to run on their own computers. This formula has been responsible for the growth of today's commercial software industry: a $400-billion business behemoth that the United States dominates, and the products of which impact nearly every person on earth.

Software has traditionally followed a consistent business archetype.

Today, however, that archetype is being challenged by a new software formula: open source. Developed and maintained by volunteers, distributed to users at no cost, and available in source form, it is radically different from its commercial counterpart. Open source promises to shift the balance of power from vendors to users: Information technology (IT) organizations can, for the first time, control their own destiny.

Open source differs radically from commercial software and offers users much more control.

Open source's control comes with new responsibilities.

However, this control comes with a price. As the story of Aladdin and the magic lamp illustrates, magic powers carry with them new responsibilities. Each of the new characteristics of open source software forces IT organizations to develop new ways of thinking about how they procure and implement software.

Open source offers IT organizations much more freedom.

Available without cost, open source is distributed to users under different licensing terms from commercial software. Open source licenses offer IT organizations much more freedom in how they use software—freedom to install it wherever they want, modify its source code if they wish, and even redistribute the modified source to anyone they choose. IT organizations have far more power over their software infrastructure now than at any time in the history of computing.

Open source is created much differently from commercial software.

In addition to the different license conditions, open source software is created under different conditions from commercial software. Instead of being developed by a private company that takes responsibility for all aspects of the product, open source is written by a small group of developers, typically unpaid volunteers. For delivery of all other product elements, open source developers rely on the user community or other commercial entities.

Open source causes IT organizations to use new methods to select and assess software.

Because of the differences between open source and commercial software, IT organizations must use new methods to select and assess software. Open source users must take responsibility for locating all the product elements that commercial software companies typically deliver along with their software: support, training, documentation, and the like.

With open source, users must take responsibility for the quality of the complete product.

Locating the elements is just half the battle, however. Unlike the commercial software industry, where the vendor takes responsibility for the quality of each of the product elements, in the open source world that responsibility falls to the user. Each element must be evaluated by the user to determine its quality. This responsibility demands a new model of product procurement—one where the IT organization is an active participant in creating the complete product, rather than a passive recipient of what the vendor delivers.

The new archetype demands new working practices. Just as the software industry has had more than 40 years to perfect its business practices, software users have had more than 40 years to hone their skills in procuring and implementing commercial software. For open source, new skills need to be developed and used.

The new open source archetype demands new working practices.

This chapter begins the process of outlining those skills with an introduction to open source. It addresses the following topics:

- What is open source?
- Who creates open source?
- Who uses open source?
- Where do I get open source software?
- When and how do I use open source?

What Is Open Source?

When most people hear the term *open source,* their initial reaction is, "What is open source? What does it mean?" Simply put, open source is software that has the following characteristics:

Source Code Availability

Open source is software that has source code available to its users. It can be downloaded at will and used or modified as desired, as long as its license requirements are observed. This differs significantly from commercial, or proprietary, software, which is distributed only in binary format to ensure that its intellectual property

Open source means that the product's source code is available to all users.

remains privately held by the software creators. Commercial software is delivered in frozen form: It must be used as delivered.

Open source products are usually available in binary form as well.

Open source products are usually also available in binary form so that they can be used on common operating systems without needing to be compiled first. Of course, not every operating system will have a binary available, but the source code makes it possible for the product to be compiled for any operating system that does not have a binary version available.

Open source licenses impose far fewer restrictions on users.

Open source software licenses are far less restrictive in terms of how the software can be used. This does not mean that there are no conditions imposed by open source licenses. Open source usually allows an organization to use the software in any way it desires, but often requires that any changes made in the source code be shared with the user community and given to any customers of the organization that makes the change.

Zero-Price Software

Open source is usually available without charge.

Open source software is distributed at no cost (this is mostly true; see Chapter 2, “Open Source Business Models,” for a discussion of open source products available for purchase). This makes sense because it reflects the reality of source code availability. There is no way to control distribution of a software product available in source form. If any attempt were made to limit the product’s use by, for example, locking the executable onto a single processor, the source could be modified to take out that portion of the code. Free source implies zero-price software.

Open source is different from freeware.

There is no charge for the source code either. In this way, open source differs significantly from *freeware*, a type of software open source is often confused with. Freeware is software distributed without a fee, but without source code access. Freeware creators tightly restrict the intellectual property rights to the software and

Golden.book Page 7 Friday, July 16, 2004 11:03 AM

offer the software on a "take it as it is" basis, in contrast to open source, which carries far less restrictive licensing terms and allows users to modify the product if they so desire.

Freeware is often distributed on a "time-bombed" basis, meaning it is free for use for a certain period of time. When that period is up, the software stops working. If the user wants to continue using the product, a licensing fee is necessary to defuse the "time-bomb" restriction.

The fact that open source software is zero price offers tremendous benefits to users:

- The common dilemma of wanting more installations than they can afford is avoided. Use of additional copies often increases the benefits of the software to the organization.
- The availability of zero-price software encourages innovation. Expensive software forces IT organizations to purchase software only for proven applications. Lower costs allow organizations to experiment and develop new applications or even new lines of business. This allows an organization to take greater advantage of IT in running their business.
- Zero-price software enables IT organizations to stretch their budgets farther and purchase software that they might not have been otherwise able to afford. The dollars saved by using free software allows funding for additional applications that might not have otherwise made the budgetary cut.
- Zero-price software reduces overall IT costs, allowing an organization to make greater investments in other aspects of its business. Free software reduces the number of capital expenditure trade-offs that companies must make.

Free Software and Zero-Price Software

Readers who wonder why the unusual “zero price” term is used instead of “free” should be aware that some participants in the open source community have a very different meaning for the term “Free Software.” The adherents of Free Software believe that computer software should be widely available with no restrictions placed on its use, study, copy, modification, or redistribution.

Most participants in the open source community do not share these beliefs, but instead feel that intellectual property should be made available according to the motivation of the creator(s). Much more information about Free Software can be found at the Free Software Foundation’s Web site (*www.fsf.org*).

It might seem a bit confusing to use the term zero price to refer to the cost of open source software, but this alternative seems preferable to the potential confusion of using the term free. The term Free Software carries with it much more implication than zero price, which should be kept in mind. In this text, zero price refers explicitly to the fact that most open source products are available at no charge.

Open Source: A Different Licensing Model

Software licenses protect the intellectual property of the creator(s).

All software licenses reflect the rights of the creator to control how the software is distributed. Software is a copyrighted entity that embodies intellectual property, and, as such, enjoys the legal protection of copyright law. Although copyright law is often used to restrict use of a product, the law can be used to enable wide distribution as well. Every piece of software is distributed under some kind of license, which controls the manner in which the product can be used.

Open source licenses are significantly different from commercial software licenses.

Open source licenses differ significantly from commercial software licenses. Commercial licenses restrict the use of the software as much as possible, to enhance the possibility of selling many licenses. In contrast, open source licenses are written with the aim of encouraging wide use, with very few restrictions placed on the use of the software.

One way of viewing the differences in licensing practices between commercial and open source is that commercial software licenses are written to allow the software creators to harvest the value that users receive from the product. By contrast, open source licenses are written to allow software users to harvest the value from the product. As noted earlier, users are allowed to modify the source code if they desire to increase the value they receive from the software. Other benefits of open source licensing include the following:

- Users are not restricted as to which or on how many machines they can install software. Commercial licenses typically control very tightly on how many machines software can be used. Being able to install as many copies as desired is a great benefit to users. As application use grows, it is easy to expand the number of copies installed for load-balancing purposes. Furthermore, organizations can install additional copies of open source software for training, testing, demonstration, and integration purposes. The flexible licensing terms encourage organizations to use software in ways that offer the greatest benefit to them. Strict licensing terms often restrict users from using software in ways that offer them the greatest possible benefit.
- There are no restrictions on access to later versions of the software. Commercial software licenses often require large “maintenance” payments to enable user organizations to access patches, maintenance releases, and upgrade versions of the software. Open source software imposes no such restrictions.
- The user communities for open source products are usually much larger than for commercial products. Because the products are available at zero price, many more organizations use the products. Large user communities offer many benefits to the developers and users of the software.

Open Source: Free Speech, Not Free Beer

"Free speech, not free beer" embodies the beliefs of open source.

Open source devotees often describe the importance of its licensing with the phrase "free speech, not free beer." The point of this phrase is that, although open source software is usually available at zero price, the critical aspect about it is that open source offers real freedom for software creators and users. Specifically, the "free speech" part of the epigram refers to the liberty that the users of open source have to use, modify, and distribute the software. This liberty is tied to the licensing conditions that make source code available to software users.

"Free speech, not free beer" emphasizes the rights that accompany open source.

Source availability means that the uses someone can make of software are nearly unlimited. It can be copied. It can be modified for one's own purposes. The modified version can be distributed as well, if one chooses to do so. So, "free speech, not free beer" emphasizes the rights that accompany open source licenses and points out that these rights are not the same thing as zero-price software.

Not all users are focused on the free speech part of "free speech, not free beer."

"Free speech, not free beer" might overstate the appeal of free speech and understate the appeal of zero-price software to most open source users, just as New Hampshire's motto "Live Free or Die" might be only a small part of the residents' enjoyment of the state—the larger part being the low taxes they pay. However, open source licensing terms make possible the free availability of products as well as underlying the other benefits of open source software. The topic of Free Software is addressed in the "Free Software and Zero-Price Software" sidebar as well as in the licensing section of Chapter 3, "Open Source Risks."

Conditions of Open Source Licensing

Open source does carry licensing restrictions.

Just because open source products are available with an unrestrictive license does not mean that there are no licensing conditions at all. Typical license conditions include contributing any source 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 type of open source license that accompanies a given product. The topic of open source licensing is addressed at greater length in Chapter 3, "Open Source Risks."

Who Creates Open Source?

A consistent question regarding open source is, "Who writes open source software?" A second, often-unasked question is, "Why would anyone work on open source?" Many people don't understand why someone would program without financial compensation, because they view programming as unfulfilling drudgery. Alternatively, many people believe that open source developers must be students or unemployed, with an assumption that they work on open source in place of a real job.

A key question is, "Who creates open source and why do they do it?"

Who creates open source software and how they support their work on open source is, however, key for pragmatic users. IT organizations need to use software that will be available and supported for the long term—their software infrastructure must be "future-proof." Relying on software created by people who are uncommitted for the long term is too risky. After all, no IT organization wants to find that a key piece of technology is suddenly orphaned because the developers lost interest or had to "get a real job."

IT organizations view the ongoing involvement of developers as key to software success.

Of course, the availability of source code makes a product future-proof in some sense. Even if the developers end their involvement with an open source product, users have the source code itself to rely on for use in the future. This really isn't enough for most IT organizations, however. Almost all commercial enterprise software purchases come with source code escrow agreements, which make the product source available if the vendor goes out of business. IT organizations avoid doing business with vendors when they suspect

Source code access is not usually enough for pragmatic IT organizations.

the escrow conditions might come into play, however. IT organizations want working software, not a code base. Source code escrow arrangements are a last resort, not a procurement strategy.

Even with source availability, the question of who creates open source remains.

Therefore, most IT organizations do not perceive the source availability of open source products as their path forward. Even those that work with source code want to contribute to ongoing product development rather than taking on sole responsibility for the product. Therefore, the question of who creates open source software remains key. Who are open source developers? Can they be relied on to create a long-lived product?

A good profile of open source developers and development practices is available.

Fortunately, there is good information available about the open source development community. In 2002, the Boston Consulting Group (BCG)[1] carried out a large survey of the open source community in cooperation with SourceForge, an open source portal. They did this to better understand the potential of open source as well as how much risk is present for open source users. BCG contacted more than 1,500 randomly chosen open source developers with a Web-based survey and received more than 500 responses. The findings of the survey provided a snapshot of the open source development community; more important, the findings contradicted the assumptions many people have about open source developers. (The complete findings of the survey can be found at *www.osdn.com/bcg/*.)

1. “The Boston Consulting Group Hacker Survey,” presented at the O’Reilly Open Source Conference, San Diego, CA, July 24, 2002. Available at *www.bcg.com/opensource/BCGHackerSurveyOSCON24July02v073.pdf*.

Why Do Developers Work on Open Source?

BCG found that open source developers are motivated by intellectual curiosity and a desire to improve their skills. Many of them consider programming to have an aesthetic appeal, like poetry or music. For these developers, working on open source is far from a burden; it is a chance to do something they find personally fulfilling. In fact, a majority of them agreed that "when I program, I lose track of time" and that "with one more hour in the day, I would spend it programming." A large proportion of the respondents also felt a sense of personal accomplishment by working on open source.

Open source developers consider programming a mode of self-expression.

With respect to the issue of risk being posed by the developers of a project abandoning it, a significant percentage believe that one of the requirements of working on open source is finding someone to take on the project if a developer leaves it. Developers begin contributing to an open source product out of a sense of interest and typically develop a personal stake in their work. For that reason, they are unlikely to abandon a product without seeing that someone else is ready to take over their role. Still, there is some risk that one or more members of the development team might walk away from a product, leaving users exposed.

Open source developers consider continuity of their project very important.

What Are Open Source Developers Like?

To a large degree, open source developers reflect a trend that has been noted in many other professions: stronger identification with peer professionals than with organizations. An overwhelming majority—83 percent—agreed strongly or agreed somewhat with the statement, "Hackers are a primary community with which I identify." It should be noted that, in this context, *hackers* refers to very technically oriented individuals and not to people with malicious motivations.

Open source developers identify most strongly with their profession.

The survey respondents were mostly between 20 and 30 years of age and 98 percent were male. They averaged 11 years of professional IT experience.

In terms of geographic location, North America is home to approximately 46 percent of open source developers, Europe accounts for about 42 percent of developers, and the remaining 12 percent are located in other areas of the world.

The geographic distribution of open source developers is different from open source users.

Interestingly, this distribution of developers does not match the distribution of open source users. About 24 percent of all downloads from SourceForge are from Internet domains located in the United States, with the remainder going to international domains. In terms of individual countries, page views (a proxy for downloads) identify the three heaviest user nations of SourceForge other than the United States as Germany, Canada, and the United Kingdom.

How Do Open Source Developers Support Themselves?

This is the question implied in the second question listed at the start of this section: "Why would anyone work on something for free?" Two surprising results came out of the BCG survey:

- A full 30 percent of those surveyed participate in open source development as part of their employment. These developers work in organizations that use open source products and they participate in the project to make the product work better for their employer's needs.
- Well over 50 percent of those surveyed are professionally employed in technology organizations. About 20 percent of those surveyed are students, with 7 percent being academics, and 15 percent identified as "other."

Therefore, most participants in open source development already work on technology. Their involvement in an open source project usually is in addition to their “real” job, motivated by skill development or the opportunity to work on an intellectually stimulating project. By no means are the participants only students or the unemployed.

Although the majority of open source developers already have full-time technology jobs, they devote a significant amount of time to their open source efforts. Volunteer participants (those who do not work on open source as part of their regular employment) contribute almost 6 hours per week to open source work, whereas those who are paid participants contribute a little more than 11 hours per week.

Open source developers devote significant time to their open source activities.

Implications of the BCG Survey

The BCG survey provides an excellent overview of the open source development world. Most open source developers are IT professionals who work on projects to improve their skills or for intellectual stimulation. Far from the stereotype of inexperienced or unemployable engineers, open source project developers have significant IT experience. They are usually employed in technology jobs and are unlikely to abandon a product and leave its users in the lurch. Open source developers have a strong commitment to the product and are reluctant to see its users harmed in any way. Consequently, the risk associated with using a product created by volunteers is probably not as high as many potential open source users believe.

Far from the stereotype of unemployed engineers, most open source developers have significant IT experience and are employed full-time in technology positions.

Who Uses Open Source?

The short answer is “everyone.” If you’ve searched with Google, purchased books from Amazon, or placed a call with MCI, you’ve used open source. Each of these organizations uses open source as part of its core computing infrastructure.

Open source is used by a number of major corporations.

Many other corporations are actively experimenting with open source.

The somewhat longer answer is that many organizations currently use, are actively experimenting with, or are thinking about using open source. To date, users of open source have mostly been early adopters; pragmatic IT organizations are now beginning to consider open source, mostly for the reasons outlined earlier. A detailed discussion of how these early adopters and pragmatic organizations use IT is contained in Chapter 4, "The Open Source Maturity Model."

An open source product's user group is usually called its *community*.

The even longer answer is that open source is used by the product's user community. This might seem redundant, but the term *community* (or *user community*) is one you will hear repeatedly in discussions about open source. One of the key differences between commercial software and open source is captured in this phrase. To understand it, you need to consider the relationship between developer and user.

Commercial software developers are not usually accessible to the product community.

With commercial software, there is practically no interaction between software developers and the people who actually use the product. Most companies seek to shield their developers from users to enable them to focus full time on banging out the code needed for the next release.

Open source developers are usually highly involved with the product community.

In contrast, on open source projects, there is a great deal of interaction between developers and product users. In fact, one might say that there is a very intense relationship among all individuals involved with the product—whether developers or users. E-mails fly back and forth among the development team and product users. Feedback is sought and freely given. Without romanticizing community, it's critical to understand it and recognize how you can interact with it and take advantage of it. It's worth a discussion about community to see how it impacts how you will use open source. The place to begin community lies with the development team and how they work. After that, we can explore how the user community affects and is affected by open source.

Open Source: Two Case Studies

Many people believe that open source software is not used in mission-critical production applications. Here are two examples of companies that rely on open source to run their mission-critical applications.

Charles Schwab

Charles Schwab runs a very large Web site that allows customers to buy and sell financial instruments, check the status of their accounts, and research potential investment opportunities. Traffic to the site varies enormously, depending on what is happening in the stock market on any given day. In the past, maintaining sufficient reserve computing capacity was difficult because of the high cost of the hardware. Over the past year, Schwab has been trading out proprietary UNIX boxes for inexpensive Intel-based machines running Linux. Geoff Penney, Schwab's chief information officer (CIO), states that the company had saved hundreds of thousands of dollars bringing in less-expensive machines. Moreover, it had also increased its ability to respond to traffic spikes, because it was now able to afford to have reserve capacity available.

Sabre

Sabre Holdings is a global travel company that operates the largest airline reservation system in the world. This system is so complex that its original mainframe systems ran an operating system written specifically for their application—the standard operating system couldn't handle the transaction load. Today Sabre not only operates this reservation system, but also offers software and services to travel organizations and airlines throughout the world. To reduce its operations costs, the company extensively uses open source software. Today, its core applications still reside on legacy systems, but they are surrounded by Linux-based servers using the open source database MySQL for storage, and communicate via the open source TAO object request broker. The services themselves are offered as Simple Object Access Protocol (SOAP)-based Web services that are constructed with open source software.

These applications perform hundreds of thousands of transactions each day, all of which execute on open source software systems. Sabre estimates that it has saved millions of dollars using open source software to enable external access to its core applications; in fact, because the airline industry has such thin margins, these applications would not have been economically feasible if the company used commercial software.

Created by Volunteers, Not Employees

Open source software is usually developed by volunteers.

Open source developers are typically IT professionals who donate their work on a voluntary basis (this is not universally true, as will be discussed in Chapter 2, "Open Source Business Models." By and large, however, most participants in an open source product are there by choice on an unpaid basis. In contrast, commercial software is written by paid employees.

The voluntary involvement of developers affects open source management practices.

The fact that open source is written by volunteers affects how open source product teams form and work. Because individuals participate based on their interest in the product, open source management practices are very different from those in commercial software companies.

Open source management practices require very different techniques.

Anyone who has worked in a voluntary organization will recognize the key differences between the two types of organizations. People volunteer for personal reasons, so getting individuals to work on something that doesn't interest them is fruitless. They won't do it—they just drift away. Volunteers must be emotionally engaged to work on a task and "managing" a volunteer is chiefly an exercise in figuring out a good way to motivate him or her. The role of a manager in an open source project involves lots of emotional stroking and personal interaction, which takes time and forfeits urgency. It can all seem very touchy-feely: There is no such thing as a well-disciplined voluntary organization.

Paid employees might not be as committed as one might think.

On the other hand, one might argue that the motivation of paid employees is less than complete. Just because an employer offers a paycheck doesn't mean that an employee is strongly motivated. Employees can accept their pay but disdain their tasks. Anyone who has spent time as a manager knows that some employees find it impossible to be enthusiastic about their work. If they're not enthusiastic, they deliver minimal effort. They might actually deliberately hinder their team's progress if they're unhappy enough. Therefore, it's important not to overstate the power that an employer has because it supports its employees' living standards.

Volunteers' passionate commitment can lead to great products.

There are tremendous benefits to having volunteers write open source software, however. Volunteers bring real passion to what they do. Again, anyone who has worked in a voluntary organization has seen the tremendous effort expended by participants. The inefficiencies in managing open source projects are balanced by the enthusiasm the engineers bring to the tasks they select. Great products are achieved by passion; it might not be missing in software written for pay, but it surely cannot be missing in software written for free.

Open source development teams work in a highly decentralized, consensus fashion.

Because they are voluntary groups, open source development teams work together in a decentralized fashion with little hierarchy. The project leader is usually the individual 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 functionality the upcoming product release will contain, encouraging new developers to join the project, helping developers select a portion of the project to work on, and arbitrating any conflicts that arise between team members.

In summary, egalitarian groups working together as volunteers create open source products. These groups operate with all the glories and frustrations that accompany any voluntary organization.

Development Practices

Open source projects "release early and often."

Open source projects tend to make early releases available for use by the user community and quickly refresh the releases as the product is modified. It's not unusual for new versions of a product under development to be released every few days. This practice is described as “release early and often.” The open source community believes that this practice leads to higher-quality products.

“Release early and often” results in higher product quality.

The reasons for this belief are straightforward. Developers create and test code based on their assumptions about how it will be used. However, actual users use (and misuse) the product in ways that no developer could possibly have imagined. This use or misuse exercises unexpected code paths and stresses the code in unforeseen ways. By making the product widely available, a large pool of users quickly performs this product exercising and more quickly improves the product’s quality.

Source code availability means that bug fixes are higher quality.

Because the source code for the product is available, a second factor comes into play. As the epigram “two heads are better than one” illustrates, additional perspectives about problems lead to better solutions. As many of the people using (or misusing) the product access the source code to create fixes that they then submit to the core development team, the overall solution created as a blend of the different fixes will be of higher quality than any one fix possibly could be. Certainly the use of a large pool of developers beyond the central team enables a broader perspective to be brought to bear on the source code. The benefit of having many people working on the source code is summarized in the open source shibboleth “many eyes make all bugs shallow.” A side benefit of having many people looking at the source is that the code is reviewed for adherence to coding standards; fragile or inflexible code can also be improved as a result of these reviews. Generally speaking, code reviews are considered to be a very positive quality practice in software engineering.

Direct user feedback ensures that the product implements critical functionality.

A third factor that affects open source development practices is that large numbers of real users work with the product and offer feedback directly to the development team. This feedback enables the team to learn what features the product really needs to include to be more useful. By contrast, commercial product companies often suffer from what is known as *feature creep*, focusing on delivering more features in a race to outshine competitors rather than on what product users really need. There is no pithy catchphrase to express this open source practice, but the direct involvement of end users is believed to lead to more useful products.

A Philosophy of Community

The practice of frequent releases to gather user feedback highlights one of the most important aspects of the open source world: the community. The size and activity level of the community carries significant implications for organizations considering a product. Community pervades discussions of open source, and is discussed throughout this book. More particularly, the size and activity level of the community directly affects the maturity of the product. This book extensively addresses all the ways that the user community impacts the maturity of an open source product, and how you or your organization can use the user community to assess the product's maturity.

One of the most important aspects of open source is the community.

What is an open source community? It is all the product developers, any users who are interested in participating, and any other individuals who care to be involved with a product. Essentially, it is a freewheeling organization of everyone who is interested in a particular product for whatever reason. There are no formal requirements for joining and no formal rules for participation.

A product community is open to any interested participant.

However, lack of formality does not mean that there are no standards for participation or behavior. Very strong 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, to contribute to as well as take advantage of the community.

Easy participation does not imply lack of behavioral standards.

Open source product communities are very powerful and offer tremendous benefits:

- As described earlier, the interaction between developers and users makes for a more useful product. The product is better for having more people determining its direction and improving its stability via usage.
- The community offers a real resource to draw on for expertise in the product. It is very common for members of the community to help one another figure out problems or to suggest potential solutions for product usage.

There are some drawbacks, however:

- Because of the need for consensus and the large community size, the pace of work and decision making can seem very slow. There really isn't any way around this, but it usually doesn't cause an enormous problem. In any case, if you feel strongly about something, there is always source access to more quickly make changes that you feel are time sensitive.
- Despite the strong informal rules of participation, immature behavior sometimes (rarely, really) occurs. This can take the form of namecalling, but the more common form is one-upmanship based on technical opinion—the "it is obvious to anyone who really understands how objects interact that it should . . ." sort of thing.
- Occasionally, the community will engage in protracted discussions about off-topic subjects. Depending on your temperament, these discussions might be charming, mildly distracting, or irritatingly time wasting. I tend to the latter point of view, but have learned to grit my teeth because of the community benefits.

One unique aspect of an open source community is that it is essentially anonymous and extremely decentralized. Nearly all interaction is done via e-mail and Web forums. It is common for members of the development team to have never met, but to have worked together quite closely to create a high-quality product. Many members of a product community will cooperate to solve a problem successfully, all while being scattered literally around the world.

The community is anonymous and extremely decentralized.

Can Community Work?

Many IT managers are initially reluctant to use a product that relies so heavily on an informal community organization. They feel open source communities that lack hierarchical leadership and a formal governance structure must be inefficient and chaotic. It's easy to understand this reaction, because so much of corporate and governmental life revolves around hierarchies and formal rules of behavior.

Many IT managers question whether a community-based product can be successful.

At bottom, this reaction is based on a fear that community means no one is truly responsible for the product—that there is no one to turn to about a specific problem or an urgent need. There is a comforting sense that a commercial provider offers a responsible point of contact, sometimes charmingly referred to as "one throat to choke." On the other hand, having a commercial provider standing behind a product is no guarantee of responsible behavior. The software industry has witnessed many instances of companies abruptly declaring products "no longer supported" or defining them as having reached their "end of life plan." Another perspective on "one throat to choke" is "single point of failure," which is usually considered a bad thing.

Commercial practices are no guarantee of success, however.

However, there is a precedent for a technology successfully delivered by an informally structured organization. It is one that every reader of this book likely uses each day: the Internet. Despite its

The Internet itself is an example of a community-based success.

Golden.book Page 24 Friday, July 16, 2004 11:03 AM

lack of formal hierarchy, use of the informal request for comment (RFC) standards mechanism, and reliance on personal reputation for leadership, the Internet has succeeded brilliantly. It is unlikely that a single commercial entity could have created what this informal organization has brought about. Many observers even argue that the lack of formal structure and regulations for the Internet have enabled it to evolve more quickly and be more useful than would otherwise have been possible.

Community is one of the benefits of open source software.

Many discussions about open source denigrate the commercial software world and describe open source methods as far superior and perhaps even better morally. The informal community is portrayed as delivering much better products that will eventually send commercial software to its deserved burial in the elephant's graveyard. My own view is that there are many powerful benefits available from using open source and the community is one of those benefits. I don't believe that commercial software is going to disappear by any means; however, I do believe that open source software will become an important part of every IT organization's infrastructure and that taking advantage of the community is vital to success with open source. Open source succeeds brilliantly at delivering software, and the community is an important part of that success.

Where Do I Get Open Source Software?

The most convenient place to get a product is from one of the open source portals.

Open source software is available from many different places. Individual open source products might have their own Web site to make the product available. There are several open source portals, which act as repositories of open source software. Many open source products are available at these portals, making them convenient for locating products via the portal's search capability. Finally, a few open source products are available for sale, typically made available by companies that have bundled the basic open source product along with some useful utilities and possibly an improved installation mechanism. Much more is said about commercial distributions in Chapter 2, "Open Source Business Models."

Individual Open Source Product Web Sites

Some open source products have their own Web site for distribution.

Some very well established open source products have their own Web sites that act as the main distribution mechanism for the software. The Web sites act as gathering points for developers and the user community to interact. They often have forums for discussions and questions among the community. News about the product will be available as well. These Web sites are the electronic equivalent of an old-fashioned country store in which transactions, friendships, information swapping, and gossip all take place. The sites themselves can easily be found via a Google search on the product name.

Open Source Portals

SourceForge is an excellent open source portal.

Open source portals offer a centralized location for open source products. The portals host open source projects, offering a number of services that make starting and maintaining an open source project much easier. As noted earlier, the fact that many projects are homed in a single portal offers real value to users, as they can easily search and sift through hundreds or even thousands of projects to find the right one. A fuller description of the services offered by the leading open source portal, SourceForge (*www.sourceforge.net*), is contained later in Chapter 5, "The Open Source Product."

SourceForge: An Overview

SourceForge is a tremendous resource for open source users and developers. It offers a full range of services to developers, freeing them from creating project infrastructure. Instead, they can take advantage of what SourceForge provides.

One interesting question is how did SourceForge itself come into existence? Who took it upon themselves to create this community resource?

SourceForge was originally created by employees of VA Linux (now known as VA Software). VA Linux was a hotbed of open source activity centered on Linux, and SourceForge was a fairly informal portal set up as a casual project. However, as the number of projects grew, VA Linux recognized that the site needed to be robustly engineered to handle the traffic it was receiving. They put a team in charge that extended the functionality, implemented a scalable architecture, and planned future enhancements. SourceForge is now one of a number of open-source-oriented portals operated by OSDN, a subsidiary of VA Software.

Commercial Distributions

Some open source products are available on commercial distributions.

A few open source products are available for sale. I know that this sounds like a contradiction of the term open source, but commercial open source products do exist. The commercial product is usually offered along with other product-oriented services, like technical support or training. Even in the companies that offer a commercial version of an open source product, however, usually the product is available at zero price as well. The version sold is merely made available in a more convenient format (e.g., on a CD) or as part of a larger product offering that bundles services along with the software.

The Challenge of Anonymous Distribution

Open source products are available for anonymous download.

One of the most interesting, yet frustrating, aspects of open source is that not only is it available at zero price, but it is available anonymously. You don't have to identify yourself to download the product: no forms to fill in, no credit card information (unless the product is purchased), no nothing.

Anonymous download makes open source acquisition extremely easy.

This is absolutely a delight. Nothing stands between you and the product. You don't have to provide personal information to get the product. There is no need to go through an extended capital request cycle because the product costs nothing. Indeed, the easy availability can pose a problem, which is discussed in Chapter 3, "Open Source Risks."

On the other hand, it can be quite frustrating that the user base for an open source product is essentially faceless and nameless. Many times, the open source developers will have no idea of the identity of most of their users. Companies might be using the product as a key part of their software infrastructure, and no one will know. If you are assessing an open source product, this can pose quite a challenge. With a commercial product, you can ask to see customer references and talk with actual users to hear how the product has worked for them. In the open source world, it can be quite difficult to locate specific users to get the same information. There are ways to address this problem, which are discussed in Chapter 5, "The Open Source Product," but nonetheless the anonymity of open source can seem quite odd.

Anonymous download means that many of a product's users will remain unknown.

When and How Do I Use Open Source?

These are intertwined questions. The right time to use open source is when both you and the product are ready. How to use open source is the subject of this book. The practices you (and the IT industry) have used over the past 40 years won't work with open source products. A whole new method of selecting and evaluating products is required to succeed with open source. The thesis of this book is that successful organizations will recognize that new methods are required and will implement them when they begin to work with open source.

Using open source requires new working practices.

The next two chapters of the book address open source business models and open source risks. Chapter 4 starts off the discussion about how to succeed with open source. It outlines the types of technology users and why the key question users must ask about any open source product is, "How mature is it?"

The key question for open source software is, "How mature is it?"