What Makes a Virtual Organization Work?

M. Lynne Markus ▪ Brook Manville ▪ Carole E. Agres

Today's workforce increasingly consists of de facto volunteers. The open-source software movement — propelled in large part by volunteer programmers — suggests ways to motivate and direct knowledge workers.

M. Lynne Markus is professor of electronic business, Department of Information Systems, City University of Hong Kong. Brook Manville, formerly a partner at McKinsey & Co., is chief learning officer at Saba Software in Redwood Shores, California. The late Carole E. Agres was a doctoral student at Claremont Graduate University. Contact the authors at: islynne@cityu.edu.hk and bmanville@saba.com.

In his 1998 article "Management's New Paradigms," Peter F. Drucker argues against the traditional view that the essential managerial task is to tell workers what to do.[1] In fact, managing a workforce increasingly made up of knowledge workers has very different demands. Managers today, Drucker tells us, must direct people as if they were unpaid volunteers, tied to the organization by commitment to its aims and purposes and often expecting to participate in its governance. They must lead workers instead of managing them.

Drucker's view of knowledge workers as volunteers seems to be on target with today's economic, business and workforce trends. A number of industries have seen the breakup of large traditional organizations and the emergence of new, networked organizational forms, in which work is conducted by temporary teams that cross organizational lines. With a booming economy, there is a shortage of skilled labor, exacerbated by an aging population and fewer new workforce entrants. High-tech companies in particular are facing a war for talent, while people increasingly value personal time and autonomy over greater income and advancement. Consequently, companies seek to harness the talents and energies of dispersed "communities of practice." At the same time, a record number of knowl-

edge workers are self-employed freelancers, and more people choose periods of less than full-time work. If those trends continue, managers will increasingly face a workforce of volunteers — at least in spirit if not in fact.

How will the traditional management tasks of motivating and directing employees have to change in the face of these new realities? One way to answer that question is to examine an example of an economic enterprise that acts in many ways like a voluntary organization: the open-source software movement. Open-source software, such as the Linux operating system, is licensed as a public good — in other words, it is essentially given away for free. And many open-source software products are built, at least in part, by people who are neither employees nor contract workers and who receive no direct compensation for their participation. Nevertheless, open sourcing has become an increasingly popular way of doing business in the software world, and many entrepreneurs and investors are making considerable money from companies involved with open-source software. (See "About the Research.")

What motivates people to participate in open-source projects? And how is participation governed in the absence of employment or fee-for-service contracts? The answers to those questions reveal some important lessons for organizations — whether or not they develop software products — about both the challenges of keeping and motivating knowledge workers and the process of managing various types of virtual organizations, such as ad hoc project teams, virtual teams, communities of practice and multicompany collaborations.

First, money is only one, and not always the most important, motivation of open-source volunteers. Although professional contributors may value a possible share in the collective wealth a successful new software project generates, they also are motivated by the personal benefit of using an improved software product and by social values such as altruism, reputation and ideology. In many cases, several motivations operate together and reinforce one another, suggesting that traditional organizations should plan for a broader array of work motivations than they often do today. (See "Why Virtual Organizations Work.")

Second, despite the clear potential for chaos, open-source projects are often surprisingly disciplined and successful through the action of multiple, interacting governance mechanisms. Membership management,

About the Research

We became interested in the open-source movement during the course of our work with a knowledge-based organization (not in the software business) that was facing a crisis of governance. Traditional principles and practices that had served the organization well through its long history no longer seemed to work. People felt distanced from organizational governance, and employee retention began to be a problem. It was generally agreed that a new model of organizational governance was needed. After hearing open-source proponent Eric Raymond speak at a public forum, we began to think that the movement might offer just the model the organization needed. We decided to undertake a case study focusing on the motivation of open-source participants and the coordination of their software-development work. We examined the extensive literature on the open-source phenomenon, filled in a few gaps through e-mail correspondence with a small number of open-source developers and triangulated those sources with the academic literature on management, virtual organizations and "public goods" phenomena, in which people jointly produce a benefit they all can share. The literature suggested the concepts we used to describe our findings, although we sometimes relabeled them.

Why Virtual Organizations Work

- A powerful set of mutually reinforcing motivations, including a share in collective success.
- Self-governance, including:
  - membership management (the ability to ensure that there is a manageable number of high-quality contributors);
  - rules and institutions that members can adapt to their individual needs;
  - the ability to monitor and sanction members' behavior;
  - reputation as a motivator and control mechanism; and
  - shared culture, values and norms of behavior.
- Effective work structures and processes, such as task decomposition and project management in software-development work.
- Technology for communication and coordination — and norms about how to use it.

rules and institutions, monitoring and sanctions, and reputation build on the precondition of shared culture to self-regulate open-source projects. The implication is that traditional organizations should consider ways to shift from the management of knowledge workers — an oxymoron, some say — to the self-governance of knowledge work.

Motivating Contributors to Open-Source Projects

There are many different types of contributors to open-source projects: organizations and individuals; initiators and those who help with other people's projects; hobbyists and professionals. Each has different needs. Our primary focus is on the many participants who are *not* employees or contract laborers of the companies that are directly involved with initiating or managing an open-source project. Such volunteers may play a significant role in open-source software product development, or they may simply participate in maintenance and enhancement work by reporting and fixing the bugs they encounter while using the software. Because they have access to the source code, volunteers can actually make changes to open-source programs, whereas users of proprietary software products (such as the Microsoft Office Suite) cannot.

Most professional volunteers have multiple, reinforcing reasons for participating in open-source projects. And there are both social and economic benefits.

The Social Benefits of Open-Source Participation

The social benefits of participating in open-source projects — altruism and gift giving, reputation and ideology — interact with the sheer joy and challenge of "hacking" to motivate professionals to volunteer their time and skill to open-source projects.

Although open-source software has been around since the 1960s, much of the buzz was created in 1997 by Eric S. Raymond's influential online publication, "The Cathedral and the Bazaar,"[2] which triggered Netscape's decision to open its Navigator browser. In "Cathedral" and subsequent papers, Raymond forcefully argues for the benefits of open-source software development and describes the motivations of participants. Carefully distinguishing between "hackers," who build software, and "crackers," who destroy it, Raymond waxes lyrical about the joys of hacking.[3] Writing software can be challenging and fun. But why would hackers give their software away?

But just as philanthropists can bask in public esteem, open-source volunteers can benefit from enhanced reputations among their peers.

Altruism and reciprocity undoubtedly play a role in open-source participation, as they do with other volunteers. Open-source contributors have told us that they enjoy the sense of "helping others out" and "giving something back." But just as philanthropists can bask in public esteem, open-source volunteers can benefit from enhanced reputations among their peers. Participating in open-source projects can be a highly visible activity: Much of it is coordinated over the Internet, where one's performance can be monitored by other members of a particular open-source community. As an example, the Mozilla public Web site, which actively solicits volunteers, posts the names of individuals responsible for various parts of the project.

Consequently, participating in open-source projects is one way that software workers can develop a name for themselves or enhance their reputations. For example, in "How To Become a Hacker,"[4] Raymond notes that writing open-source software and helping test and debug it are two critical ways to earn the respect of "alpha" hackers. Elsewhere, Raymond explains that gaining a reputation for one's work is an important reward for participating in open-source projects: "We do keep score in the open-source world. . . . Our scoreboard is the 'credit list' or the 'history file' that's attached to every open-source project. . . . If you see somebody's name on several credit lists, then you know that person is doing lots of good work."[5]

Gaining or enhancing reputation through participation in open-source projects can lead to such tangible rewards as employment opportunities or access to venture capital.[6] But in "reputation cultures" such as academia and the open-source world, reputation serves as a coin of the realm in its own right. Raymond describes the hacker culture as a "gift" culture or "potlatch" culture, in which reputation is the only measure of success: "Gift cultures [unlike exchange cultures such as our society] are adaptations not to scarcity but to abundance. . . . In gift cul-

A Brief History of the Open-Source Movement

The open-source movement has been around for a long time. Quite a number of software products have been licensed as a public good. Examples include Sendmail, Apache and Perl. The Internet itself is widely regarded as the largest and best-known example of open-source development.

But the open-source movement as we know it today largely grew out of an influential 1997 online publication by Eric S. Raymond, "The Cathedral and the Bazaar." Raymond contrasted the way in which software products such as the Linux operating system are built with the way in which proprietary software products are developed by companies such as Microsoft. Raymond claimed that opening software development to the efforts of many volunteer developers resulted in products that are technically superior to those produced under closed development conditions.

In recent years, the open-source movement has been gathering momentum. In January 1998, influenced by publication of "The Cathedral and the Bazaar," Netscape announced the launch of its Navigator browser with open-source code. In November 1998, the Open Source Initiative was formed. By late 1999, open-source software had become a household word with the IPO of Red Hat Software, a company that provides support for the Linux operating system and other open-source products. That event was pivotal to the acceptance of open-source software by businesses, which have traditionally shunned so-called freeware. Today some analysts suggest that Linux is poised to rival Microsoft's operating systems.

Date	Event	Description
1969	**Development of Unix was initiated.**	Bell Labs researchers Ken Thompson and Dennis Ritchie completed the Unix operating system in 1973. When AT&T was forbidden to enter the computer business, Thompson and Ritchie offered tapes of source code for Unix to anyone willing to pay a small copying fee.
1981	**Sendmail was developed.**	A University of California, Berkeley, graduate and Unix guru, Eric Allman, developed Sendmail. Sendmail is an e-mail transfer agent that today moves at least 75% of Internet service provider data traffic.
1984	**Free Software Foundation (FSF) was founded, and GNU project was initiated.**	Richard Stallman, recognized as the father of free software and the ultimate hacker, founded the Free Software Foundation (FSF) and started the GNU (GNU's Not Unix) project. Stallman developed FSF in part to fight the injustices associated with proprietary software triggered by the legal battles of AT&T; University of California, Berkeley; and software developers over licensing copyrights for Unix. The goal of the GNU project was to create a comprehensive — and free — Unix-compatible operating system.
1986	**Perl (Practical Extractions & Report Language) was released.**	Larry Wall released the Perl language to the Internet, and collaborative development by interested programmers ensued. Perl's capabilities include the ability to scan text files, create HTML files from text files and navigate the Net.
1989	**Cygnus Solutions was founded.**	The first commercial venture built on open-source or freeware products was founded. Cygnus provides consulting, engineering and support services for open-source software, including GNU development tool kits.
1991	**Development of Linux began.**	Linus Torvalds, then a 21-year-old computer science student at the University of Helsinki, began working on a Unix kernel to run on a PC. The kernel was the one piece of the Unix operating system that the GNU project had not yet successfully produced. When he had the kernel working, he announced his project on the Internet, essentially asking for assistance in debugging it. The response was terrific, and a powerful community evolved as people everywhere began fixing bugs and enhancing the software. Today Linux is considered a stable operating system of higher quality than Windows NT. Linux International, a nonprofit organization, distributes information about Linux and accepts donations to support its work.

A Brief History of the Open-Source Movement, *continued*

Date	Event	Description
1994	**Additional companies joined Cygnus to rally around the open-source business model.**	Red Hat Inc. and Caldera Systems Inc. were formed. The companies use open-source support-revenue models to distribute, brand, train, consult, undertake custom development and provide post-sales support.
1997	**Eric S. Raymond wrote and posted an online version of "The Cathedral and the Bazaar."**	Raymond had been involved in the GNU project, but his experience with the Linux style of development was an epiphany. In his influential article, later published in book form, he contrasts the traditional proprietary model of development (such as that used by Microsoft) with the "bazaar" style of the open-source model, claiming that the latter produces a higher-quality product by exposing the source code to many co-developers who will see bugs and suggest fixes.
Jan. 1998	**Netscape "opened" its Navigator source code.**	In an unprecedented move influenced by Raymond's article, Netscape announced that it would release the source code for Navigator 5.0, thereby establishing a prescription for other software companies to change their business-as-usual revenue model. The software, renamed Mozilla (after the original code name for the Navigator project), was released after a licensing team wrote the software licenses, posted them on the Web, and incorporated the public's responses in less than one month. Mozilla.org was formed to organize, coordinate, arbitrate and act as final authority for Mozilla changes. The Web site is overseen by Netscape employees and contains the entire process framework for Mozilla, including a mission statement and how others can participate. The Web site also describes the lessons learned in one year of operating in the open-source mode.
Feb. 1998	**The term *open source* wins over *freeware/ shareware.***	*Open source* was selected to replace the terms *freeware* and *shareware* in a brainstorming session attended by key open-source champions. The advantages of using the term *open source* are that the business world usually tries to keep free technologies from being installed and commercially oriented developers want to distance themselves from Stallman's ideological philosophy regarding the free-software movement.
Nov. – Dec. 1998	**Confidential Microsoft documents about open source end up on the Web.**	Internal memos written by Vinod Valloppillil of Microsoft identifying the open-source movement as a threat to Microsoft's proprietary software-development model were leaked to Raymond. Raymond named them the "Halloween documents" and annotated and republished them.
Nov. 1998	**"The Open Source Revolution, Release 1.0" was published.**	Esther Dyson devoted an issue of her highly influential newsletter to the open-source movement. The issue refers to the Halloween documents and describes the licensing arrangements and business models for making money in the open-source world.
Nov. 1998	**The Open Source Initiative (OSI) was formed.**	Influenced by the Mozilla release, Eric Raymond and Bruce Perens launched the Open Source Initiative as a research and educational association whose mission is to own and defend the Open Source trademark. In June 1999, OSI abandoned its trademark initiative because the U.S. Patent and Trademark Office ruled that the term was too descriptive. OSI is now pursuing a certification mark.

tures, social status is determined not by what you control but by what you give away. . . . It is quite clear that the society of open-source hackers is in fact a gift culture. Within it, there is no serious shortage of the 'survival necessities' — disk space, network bandwidth, computing power. Software is freely shared. This abundance creates a situation in which the only available measure of competitive success is reputation among one's peers."

As important as reputation is as a motivation for open-source participation, ideology also undoubtedly

plays a role. The belief that giving software away is the right thing to do is not uncommon in university computer-science research labs, where much open-source software originates. The corollary is the belief that proprietary ownership of software is evil. In the public Web site of the open-source GNU project, one page asks, "Is Microsoft the Great Satan?"[7] The page gives the strong impression that the answer is yes. A similar impression is gained from reading the "Halloween documents," which are edited versions of what are alleged to be confidential Microsoft documents discussing the open-source "threat."[8] In essence, some of the fuel for the open-source movement is the almost anarchic glee of hackers hoping to destroy what they regard as Microsoft's evil empire — as evidenced by their enthusiasm for the Justice Department's move to break up the company.

The Economic Benefits of Open-Source Participation

Although altruism, reputation and ideology are certainly powerful motivators of open-source participation, professionals cannot afford to be indifferent to economic issues. Self-employed professionals must earn a living somehow, and employed professionals must convince their superiors that working on open-source projects during company time is valuable. If volunteers' labor provided economic benefits to someone else but not to the volunteers, contributions to open-source projects would likely not occur very often. As one developer explained: "My development contributions could possibly earn you money, but I would not contribute my time and programming effort solely for you to profit. I would be contributing to your source code because it would benefit me directly in some other way."[9]

In other words, many professional developers worry about their *own* ability to benefit economically from participating in open-source projects. One benefit would be having a better software product to use.[10] Another would be sharing in the collective wealth generated by successful commercialization of an open-source software product.[11]

The appeal of the better mousetrap underlies much open-source participation. Individuals and companies can benefit by enhancing the products they have already decided to use. For example, Linus Torvalds was a university student when he initiated the development of the Linux operating system. He liked the Unix operating system but couldn't afford a workstation to run it, so he decided to use a version of Unix that would run on his PC. (See "A Brief History of the Open-Source Movement.")

Self-employed professionals must earn a living somehow, and employed professionals must convince their superiors that working on open-source projects during company time is valuable.

Similarly, individual professionals and organizations can often benefit from participating in open-source projects when their volunteering enhances the products they use. One open-source contributor was motivated to participate when he ran into problems with a 3Com card he was using. "I've been a Linux user for years," he wrote, "but I'm only just now beginning to fully appreciate open source. Using open source is like getting dozens of detectives to work together to solve a good mystery. . . . With open source, everyone shares the clues they find. And . . . a dozen other detectives are there to confirm or poke holes in the theory."[12]

In that instance, the fix the developer submitted was rejected by the software's primary author because it might have caused problems elsewhere, but the author was able to solve the problem differently. Then the author published the revision so that others could benefit from it.

The 3Com example illustrates an important dynamic of the open-source world: The results of volunteer labor are provided as a public good, free of charge. The rights and obligations associated with open-source software are spelled out in licenses, as is the case with proprietary software. Open-source rights include the right to make copies and the right to access the source code, which is necessary to modify it. Open-source obligations may include the obligation to check enhancements with lead developers (called "checking in") or to make all enhancements available to others. Although there are variations in open-source licenses and some are more favorable than others to the interests of commercial developers, the features of open-source licenses differ markedly from the licenses of proprietary software products.[13]

Open-source licenses prohibit the sale (collection of license fees) of open-source software and serve to ensure that no one party can unfairly monopolize the commercial rewards from others' voluntary labor. The approach removes powerful reasons professionals might have for not contributing to an open-source project. ("I would not contribute my time . . . solely for you to profit.")

At the same time, open-source licenses may appear to remove a powerful motivator to contribute — the ability to commercialize the product. And in fact, proponents of the open-source movement readily acknowledge that making money from open-source software is "nonintuitive."[14] Nevertheless, a number of different open-source business models, proposed or tried, can give volunteers economic rewards from their participation. The most prominent example is that of Red Hat Inc., which sells support for open-source software. Other business models include selling educational materials and using open-source software as a loss leader or as a hardware add-on. (See "Open-Source Business Models.")

The opportunity to benefit commercially from open-source software — albeit in nontraditional ways — provides a powerful incentive for many individuals and companies to collaborate in the development of a product that none of them can own. Open-source business models can provide a level playing field — giving each open-source participant a fair chance to benefit from the common effort, while licenses pre-

Open-Source Business Models

Model	Description	Examples
Support Sellers	Revenue comes from media distribution, branding, training, consulting, custom development and post-sales support.	**Red Hat Inc.** **Caldera Systems Inc.**
Loss Leader	An open-source product is used as a loss leader for traditional commercial software (similar to practices in retailing).	**Sendmail Inc.**
Hardware Add-Ons	Industry-related companies, such as computer hardware companies, use the open-source model to enable software such as printer drivers and operating-systems interface code.	**Corel Corporation** **VA Linux Systems**
Accessories	Companies obtain their revenue through sales and support for products related to open-source software, such as educational books, CD-ROMs and computer hardware peripherals.	**O'Reilly & Associates Inc.**
Service Enabler	This model develops and/or distributes open-source software to enable the creation of revenue from online services.	**Netscape Communications Corporation** • **Netscape's Netcenter Services** • **Netscape Communicator**
Brand Licensing	Brand-licensing organizations obtain revenue by creating brand names for open-source software and then charging other companies to use the brands and trademarks to create derivative products.	**Netscape Communications Corporation** • **Netscape Communicator (only Netscape can use that name; others must use Mozilla name).**
Sell It, Free It	Companies initially create and sell proprietary products but convert to open-source products. It is the extended loss-leader model.	**Netscape Communications Corporation has some elements of the model.**
Software Franchising	In this model, a company exploits its dominance — for example, its brand name — and creates other companies from which it provides services such as training. It receives franchise fees in return.	**None known, but sourceXchange and Cosource.com have business models that possess such characteristics.**

vent any participant from unfairly monopolizing all the benefits. Open-source licenses and business models provide incentives for individual companies to band together to challenge dominant software products. Of course, the dynamics of network marketplaces are such that the open-source movement could deflate rapidly if the products lose popularity.

The open-source example tells us that money as a motivator is not likely to go out of style — and that should be reassuring for traditional organizations. But a closer look at the social and economic benefits of open-source participation suggests some areas that traditional organizations will need to reflect on. (See "The Open-Source Challenge to Conventional Software Companies.") Financial compensation in the open-source world usually takes the form of a share in collectively produced wealth, rather than wages or contract fees. That type of compensation is more similar to the stock options that high-tech firms distribute widely to their knowledge workers than to the compensation practices of most traditional firms. Furthermore, credit and reputation may be equally important motivators to open-source participants. Such benefits are more rarely bestowed by today's traditional organizations. To the extent that credit and reputation are becoming a coin of the realm in knowledge work, managers should consider them essential to motivating employees.

Governing Open-Source Projects

The potential for chaos in open-source projects appears great. First of all, open-source volunteers are neither

The Open-Source Challenge to Conventional Software Companies

Many believe that software is a winner-take-all industry that produces natural monopolies. It is often the case that a single software product, such as the Microsoft Office Suite, comes to dominate competitors so completely that it drives them out of the market. Eventually, a better product may come along to challenge the monopolist. But the new product will succeed only if it becomes dominant in its turn. That involves building market share quickly and inexorably against the incumbent.*

A small entrepreneurial company with a proprietary software product faces formidable challenges in competing with a well-entrenched and well-funded monopolist. Even if the product has innovative features that are valued by the market, the dominant company will often be able to imitate the innovation, and the startup may lack the resources to continue improving its product in a competitive race. Furthermore, because the product is proprietary, other companies have little incentive to help the startup succeed. If anything, they may oppose it with their own proprietary products.

The open-software movement lets companies that could never challenge an incumbent on their own do so as collaborators. If companies codevelop or comarket a common product, that product may have a fighting chance of becoming the next monopoly. But for companies to be willing to cooperate in that way, they need to be sure they will have a fair chance of benefiting from the cooperative effort and that no other company will be able to monopolize or appropriate all the benefits of their efforts.

Open-source participants prevent monopolies through open-source software licenses. As source proponent Bruce Perens, cofounder with Eric Raymond of the Open Source Initiative, explains: "The volunteers who made products like Linux possible are only there, and the companies are only able to cooperate, because of the rights that come with Open Source. The average computer programmer would feel stupid if he put lots of work into a program, only to have the owner of the program sell his improvement without giving anything back. Those same programmers feel comfortable contributing to Open Source because they are assured of these rights:

- "The right to make copies of the program, and distribute those copies.
- "The right to have access to the software's source code, a necessary preliminary before you can change it.
- "The right to make improvements to the program."[†]

Perens goes on to explain that those rights are important to open-source volunteers because they level the playing field, allowing many participants, not just a single proprietary software owner, to benefit from the commercialization of open-source software. But because open-source licenses generally prohibit the sale (collection of license fees) of open-source software, participants need alternative commercialization strategies. Several successful business models have been tried, including the sale of support for open-source software (Red Hat Inc.) and the sale of educational resources (O'Reilly & Associates Inc.).

* *S.J. Liebowitz and S.E. Margolis, "Winners, Losers & Microsoft" (Oakland, California: The Independent Institute, 1999).*

† *B. Perens, "Open Sources: Voices from the Open Source Revolution, Freeing the Source, The Open Source Definition" (Sebastopol, California: O'Reilly & Associates, 1999).*

employees nor contractors. They can defect at any time, and the threat of job loss or wage cuts is largely ineffective as a management tool. Second, large numbers of people may participate in some open-source projects. There may be as many as 750,000 open-source developers; organizing the contributions of even a tiny fraction of that number would challenge any project leader's skills. Third, some projects welcome all comers. For example, Netscape's Mozilla site, which is updated regularly, at one point read: "So you want to help? Great! Please read the rest of the introductory documents on this web site. . . . After that, it would be a really good idea for you to join the appropriate mailing lists and to get a feel for how the community works together. Before you actually start coding up a project, we'd strongly recommend that you let us know about it. If you like, we can publicize the fact that you're working on it, and perhaps someone else will want to work with you on it. If you don't want it publicized, that's fine too, but it would help us . . . to know how many different people are tackling the same kinds of problems. . . . Active contributors can be given write access to our CVS repository. Consult the guidelines . . . to learn about our build process and to find out how to get access."[15]

Since anyone can join some open-source projects, project leaders would seem to lack that important mechanism of quality control — the ability to select people and assign them to particular tasks. In general, open-source projects have conditions that tend to promote freeloading, unstable membership or low-quality contributions.

In fact, however, many open-source projects work remarkably well, despite such management challenges. As a rule, open-source projects have well-structured governance models with clearly identified leadership. Often, the initial software developer maintains a lead role in the ongoing development and distribution of open-source products. Examples include Larry Wall's leadership in Perl and Linus Torvalds' in Linux. But it is usually the case that formal authority for an open-source project is vested in a team. For example, FreeBSD is managed by a 15-person team, called the "Core." In a few cases, project leaders are elected by the membership. (See "Open-Source Governance Models.")

As with proprietary-software-development efforts, work on open-source projects is partitioned by lead architects or designers into smaller units or modules, which can be tackled by individuals or teams. The architects also manage issues that require coordination across teams. The same pattern of deconstruction may occur within each module. Although module owners solicit (or receive) input from members, they have final authority for decisions in cases of disputes.

Open-source projects exhibit four interrelated coordination mechanisms, all in the context of the shared hacker culture: managed membership, rules and institutions, monitoring and sanctions, and reputation. By means of the interaction of such governance mechanisms, open-source projects can stay on track despite their obvious potential for chaos.

Managing Membership in Open-Source Projects

Anyone can get involved in a particular open-source project by identifying a bug or contributing a fix using Internet-based reporting systems. But achieving a responsible position in an open-source project requires a more formal process of vetting and quality control. For example, Apache volunteers are allowed to work on a project for a number of months, during which time the quality of their work can be assessed. After a probationary period, formal membership — the ability to remain in a particular role — is determined by a consensus vote of core Apache Foundation members.[16]

The Debian development effort (GNU/Linux, GNU/Hurd) is less democratic when it comes to accepting members. Project leaders (or delegates) have total authority over who works on their projects: They can admit or expel members. Project leaders are elected by developers; project leaders in turn appoint technical committees. If a developer does not agree with the actions or behavior of the project leader or technical committee, he or she can challenge those actions by following a documented resolution process.[17]

These examples show that those working on open-source projects find ways to restrict a potentially large and possibly unqualified workforce to one that is small enough to be manageable and of high enough quality to get the job done. But they also show that managing the membership of open-source projects works in conjunction with rules and institutions (such as how members and leaders are chosen) and with monitoring and sanctions (such as dispute-resolution processes and the ability to expel members).

Rules and Institutions

In well-functioning, cooperative efforts, community members participate in making and changing the rules, and the rules they adopt fit their unique needs.[18] An example of the adaptability of open-source rules is the variety of open-source licenses. For example, the Berkeley Software Distribution (BSD) license poses no limits on the commercial use of open-source software, whereas the General Public License (GPL) prevents the release of proprietary derivative products through its "copyleft" (as opposed to copyright) provisions. In other words, some open-source licenses are more compatible with the interests of developer communities that want to commercialize open-source software. A drawback of so much variety is that some open licenses are incompatible with others. It is impossible, for example, to create — legally — an open-source product that combines elements of products licensed under GPL and Mozilla Public License (MozPL).

Another interesting aspect of rules and institutions in open-source communities is their procedures for discussing and voting on important issues. Most voting takes place over the Internet. For example, the Perl project uses a phased discussion-group process: request for discussion (RFD), first call for votes (CFV), last CFV, and results. (Each phase of the process is conducted during a specific, limited time period.) An example of the process at work is documented online. When Perl's central electronic discussion group got bogged down with issues unrelated to

Open-Source Governance Models

Project	Governing Body/Leadership	Dispute Resolution
Apache	The governing body is the Apache Foundation (formerly called the Apache Group). It is a core group of about 20 members formed from the founders of the project, who vote on decisions by committee. The members elect a board of directors (currently nine members) who manage organizational affairs per the bylaws. The board appoints officers to oversee day-to-day operations, and officers assign project leaders to specific projects.	Changes to products are proposed on a mailing list and voted on by active members. Three yes votes and zero no votes are needed to agree to a code change for any given release cycle.
The Debian Project (GNU/Linux, GNU/Hurd)	According to the Debian Constitution, "Each decision in the Project is made by one or more of the following: "1. The Developers, by way of General Resolution or an election; "2. The Project Leader; "3. The Technical Committee and/or its Chairman; "4. The individual Developer working on a particular task; "5. Delegates appointed by the Project Leader for specific tasks; "6. The Project Secretary." For the Linux product, Torvalds is the universally accepted leader who delegates responsibility for the major components or modules to trusted subordinates. Raymond refers to this as the "benevolent dictator" model of leadership.	Resolutions are placed on a discussion list for review and vote. The voting process is well defined in the Debian Constitution: http://www.debian.org/devel/constitution.
Perl	The Perl Institute was established to oversee and support Perl projects. It consists of six core members, including Perl creator Larry Wall. The Perl model for leadership is a rotating dictatorship, but the model is in the process of being redefined to reflect its grass-roots approach to product development. Recently, the board voted to dissolve the institute and combine its resources with Perl Mongers Inc. (more of a high-level user group than a decision-making body). There is significantly less structure than in the Apache, Debian/GNU/Linux or Mozilla projects. Project leaders have a high degree of autonomy in decision making.	Project leaders, or "pumpkins," are responsible for the projects' decision making and conflict resolution.
Mozilla	Mozilla.org was created in 1998 to organize and manage the Mozilla development community. It is led by a small group of Netscape employees to oversee operations. Module owners are viewed as benevolent dictators who arbitrate what happens in a module. The majority of module owners are Netscape employees.	Mozilla.org has the final say in any dispute.

product development, core members proposed a moderated discussion group for general discussions. The RFD related to that change included the rationale for creating the group, its charter, specifics of administration and the names of the action's proponents. After two CFVs (including detailed instructions for the electronic voting process), an unbiased, third-party volunteer tabulated and posted the results. In total, approximately 1,200 members of Perl newsgroups participated in the vote, in which a two-thirds majority was required for the motion to pass.

Linux uses a similar voting process. When community members believed that the existing newsgroup, comp.security.unix, was aimed at Unix administrators rather than at home and small-business users of Linux, they voted to create a new newsgroup to complement the other newsgroup.[19] The first call for discussion refers the reader to the online Introduction to Usenet Voting, which reveals another important aspect of open-source governance — it is not a democracy: "Understanding this principle is necessary to understanding how Usenet operates. You, the user, have absolutely no rights regarding Usenet as a whole, and you cannot complain about anything being 'undemocratic,' 'violating your civil rights,' 'un-Constitutional,' etc. . . . The final arbiter of which groups are created and which ones are not is not the voting system! The one person who can overrule the voting system is the moderator of news.announce.newgroups. Currently this job is held by David 'Tale' Lawrence, occasionally known as the 'Thousand-Pound Gorilla' thanks to the old joke (Q: Where does a 1,000-pound gorilla sit? A: Anywhere he wants to.). Jokes aside, while Tale has overturned voting results on occasion, he has done it only very, very rarely and even then only after lengthy deliberations."[20]

Monitoring and Sanctions

Rules and institutions are vulnerable to decay if members have no means of observing behavior and ensuring compliance. Successful communities must have access to graduated sanctions against those who violate group norms and to low-cost mechanisms for resolving conflict.

Sanctions and conflict-resolution mechanisms appear to be reasonably well developed in open-source communities. There are strong social pressures against noncompliance with norms in open-source communities. Flagrant or continued noncompliance results in flaming (sending someone angry or hostile e-mail), spamming (flooding someone with unsolicited e-mail) and shunning (deliberately refusing to respond). Faced with such sanctions, members often leave the community on their own initiative; leaders don't always have to expel offending members. Even leaders are not immune to social sanction when community members are incensed by their behavior. At one point, a "flame war" erupted between open-source leaders Eric Raymond and Bruce Perens. When details of the interaction were posted online, community members responded with sharp criticism.[21]

Although not always resolved to everyone's satisfaction, conflict is generally not considered detrimental for open-source projects, because deliberation is a valued principle of community behavior.

Although not always resolved to everyone's satisfaction, conflict is generally not considered detrimental for open-source projects, because deliberation is a valued principle of community behavior. Sometimes, of course, conflicts end badly for some or all participants. One developer explained how a badly handled conflict destroyed motivation to work on an open-source project: "I now have such bad feelings associated with the whole affair that I don't like to think about [it], much less work on it. I've stopped working publicly on it."[22]

In the conflict, the same tools that support collaboration were used to destroy it: "Mr. J proceeded to mail me back and tell me that he would do whatever he pleased (again, putting it mildly). He also added a text description offering to let me perform an obscene act on him to his sig file [signature file, appended to the end of e-mail messages], which he used publicly on mailing lists and whatnot. Completely appalled at this point, I e-mailed his providers for web space and connectivity, threatening them and him with lawsuits if he didn't remove my name from his postings. This got another nasty response from Mr. J, but eventually did get him to remove my name from his sig file."

The Importance of Reputation

Monitoring and sanctions work because people care about what others think of them. While the opportu-

Performance and behavior in the open-source world are always visible: All members can see whether a volunteer has done a good job or whether a manager has followed accepted norms of behavior or the results of a vote.

nity to build a reputation may be an important motivation for joining an open-source project, the desire to maintain a good reputation is a key mechanism in making sure that open-source projects continue on track. The fear of exclusion can be a motivational factor. Performance and behavior in the open-source world are always visible: All members can see whether a volunteer has done a good job or whether a manager has followed accepted norms of behavior or the results of a vote.

Of course, it is important to realize that the same things that make communities strong also can make them narrow-minded, self-deluding and contentious.[23] As in any other community, the open-source movement occasionally exhibits political behavior, lack of fairness, work stress and burnout.

The Shared Culture of Open Source

An essential precondition for the operation of all four governance mechanisms — membership management, rules and institutions, monitoring and sanctions, and reputation — is a set of shared values and assumptions about how things work.[24] "Hacker culture" grows from, and is maintained by, direct and indirect interactions among open-source community members, which in turn are enabled by the Internet.[25] Shared cultural knowledge and enabling technology make it possible for open-source participants to collaborate successfully, often without face-to-face interaction or prior personal acquaintance.

To leaders of traditional organizations, the findings may at first glance appear to be a bit troubling. Although the importance of organizational culture is often stressed in traditional organizations, much less emphasis has been placed on building self-control (to preserve one's reputation) and social control (for peers to monitor and sanction others' behavior) into formal organizational control systems. And the idea of voting procedures for key business decisions may unsettle more than a few managers. However, the specific forms that the governance mechanisms take in open-source projects are probably less important to traditional organizations than the fact that they exist. Managers should give careful thought to the categories, the relationships among them and how they can be applied in different contexts.

The Open-Source Movement and the Organization of the Future

The open-source movement exhibits many of the qualities said to characterize new organizational forms. But those characteristics are tempered by mechanisms and processes ensuring that order reigns despite the potential for chaos:

- The intrinsic motivation and self-management of autonomous knowledge workers play important roles in open-source projects. Economic rewards and management processes are also important. Still, the types of economic rewards and management behaviors observed in open-source projects differ quite substantially from those observed in traditional organizations. For example, open-source rewards emphasize collective performance (the successful commercialization of an open-source product) and individual benefit (benefit in use) as much as they emphasize an individual's performance. And the development and maintenance of personal reputation figure prominently in both motivation and control when it comes to open-source projects.
- Membership in open-source projects is fluid, but only to some extent. Open-source projects maintain a stable core of participants while capitalizing on the temporary efforts of numerous volunteers. Membership in particular open-source communities is an important part of members' professional identities.
- Control in the open-source world is maintained through autonomous decision makers following a few simple rules. The key rules, defining appropriate conduct and fair play in benefiting from the collective effort, are embodied in software licenses, rules of membership and voting procedures. Different open-source projects adopt somewhat different rules, showing adaptability to unique circumstances.
- Self-governance, both formally through discussion and voting and informally through social control, is evident in open-source projects. At the same time, project initiators and "alpha" hackers can retain a

powerful influence on a project's direction — as long as they maintain their reputations for high-quality work and appropriate social behavior.

• Information technology is a key enabler of the open-source movement. Technology is required both for product development and support and for project coordination. Communication technology makes behavior highly visible to the community, which in turn permits enforcement of the rules. Many open-source governing bodies hold occasional face-to-face meetings to make decisions, but day-to-day decision making occurs almost solely through the Internet.

In short, there is a relatively high degree of correspondence between the open-source movement and popular depictions of the organization of the future and the virtual networked organization. Therefore, the open-source movement provides some suggestions about how management must change in the years ahead. At the same time, care should be taken in attempting to apply the lessons of the open-source movement to other contexts, because many of its characteristics are unusual or unique:

• A precondition for the open-source movement is a large "community of practice" with a strong, shared culture of technical professionalism. Whether open-source motivation and governance principles would work in other settings may depend on the existence of an active community of talented practitioners.

• Successful open-source projects have involved work that is intrinsically challenging. Whether the open-source model would work for routine activities, such as basic customer service or projects such as developing a new word processor, remains to be seen.

• The open-source movement is evolving rapidly. Whether current rules and processes will retain their character in the face of such rapid change is unknown.

• Software developers enjoy using technology for communicating and decision making. Other types of workers might prefer traditional communication and decision-making practices and fail to adapt to electronically mediated self-governance.

• Finally, self-governance is essential to the success of the open-source movement. Without self-governance, perceptions of fairness would diminish and the motivation of the volunteers would erode. Without volunteers, the movement would collapse. Furthermore, self-governance in the open-source movement is heavily reinforced by business models that offer the possibility of economic benefit and by legal arrangements designed to ensure fairness. It is not clear how well this style of self-governance would work in the absence of such strong reinforcing conditions.

Although managers in industries other than software development may prefer more traditional styles of management, they should remember that the world is changing and workers are changing along with it. In a labor force of volunteers and virtual teams, the motivational and self-governing patterns of the open-source movement may well become essential to business success.

Acknowledgments

The authors would like to thank Les Gasser of the University of Illinois, Alexander Hars of the University of Southern California, John Ferejohn of Stanford University and Josh Ober of Princeton University for their valuable suggestions about the research.

Carole E. Agres, In Memoriam

After raising her children, author Carole Agres started a career in information systems, combining work and family life with active community service and the pursuit of higher education. She had reached the senior ranks of IT management in an aerospace company — one of the few women to do so — and was well on her way to completing her Ph.D. in information science when she died suddenly in fall 1999 at the age of 50. She was an active participant in this project. She is missed.

Additional Resources

Readers might find helpful "The Economics of Online Cooperation," a 1999 Routledge book edited by P. Kollock and M. Smith. Also, the Internet site First Monday has an interesting article by K. Kuwabara called, "Linux: A Bazaar on the Edge of Chaos" (http://firstmonday.org/issues/issue5_3/kuwabara/index.html).

References

■ 1. P.F. Drucker, "Management's New Paradigms," Forbes, Oct. 5, 1998, 152-177.
■ 2. E.S. Raymond, "The Cathedral and the Bazaar" (Sebastopol, California: O'Reilly & Associates, 1999).
■ 3. E.S. Raymond, "Homesteading the Noosphere," http://www.tuxedo.org/~esr/writings/homesteading/homesteading.html.
■ 4. E.S. Raymond, "How To Become a Hacker," http://www.tuxedo.org/~esr/faqs/hacker-howto.html.
■ 5. W.C. Taylor, "Inspired by Work," Fast Company, Nov. 29, 1999, 200.
■ 6. J. Lerner and J. Tirole, "The Simple Economics of Open Source," working paper, Harvard Business School, Boston, Feb. 25, 2000.
■ 7. "Is Microsoft the Great Satan?" http://www.gnu.org/philosophy/microsoft.html.
■ 8. E.S. Raymond, "The Halloween Documents," http://www.opensource.org/halloween/.
■ 9. J. Corbet, "Letters to the Editor," Linux Weekly News, March 25, 1999, http://www.lwn.net/1999/0325/backpage.phtml.
■ 10. N. Bezroukov, "A Second Look at the Cathedral and the Bazaar," First Monday, Dec. 9, 1999, http://firstmonday.org/issues/issue4_12/bezroukov/index.html.
■ 11. C. Shapiro and H.R. Varian, "Information Rules: A Strategic Guide to the Network Economy" (Harvard Business School Press: Boston, Massachusetts, 1999); and "The Business Case for Open Source," http://www.opensource.org/for-suits.html.
■ 12. N. Petreley, "It Takes a Village of Detectives

to Solve the Case of the Faulty 3Com," Infoworld, May 4, 1998, 4.
■ 13. F. Hecker, "Setting Up Shop: The Business of Open-Source Software," http://www.hecker.org/writings/setting-up-shop.html; and T. O'Reilly, "The Open-Source Revolution," Release 1.0, November 1998, http://www.edventure.com/release1/1198.html.
■ 14. L. Radosevich and B. Zerega, "Free Money Model," Infoworld, June 8, 1998, 102-110.
■ 15. "Getting Involved," http://www.mozilla.org/get-involved.html.
■ 16. "Apache HTTP Server Project," http://www.apache.org/ABOUT_APACHE.html.
■ 17. "Debian Constitution," http://www.debian.org/devel/constitution.
■ 18. Our thinking in this section has benefited from a governance-research collaboration involving McKinsey & Company's Organization Practice, Josh Ober of Princeton University and John Ferejohn of Stanford University.
■ 19. J. Billones, "First Call for Votes," Linux Weekly News, March 15, 1999, http://lwn.net/1999/0318/a/cols.html; and J.Billones, "Results of Vote," Linux Weekly News, April 11, 1999, http://lwn.net/1999/0415/a/cols.html.
■ 20. J. Patokallio, "An Introduction to Usenet Voting," http://www.hut.fi/~jpatokal/uvv/intro.html.
■ 21. L. Kahney, "Open-Source Gurus Trade Jabs," *Wired*, April 10, 1999, http://www.wired.com/news/news/technology/linux/story/19049.html.
■ 22. N. Bezroukov, "Open Source Software Development as a Special Type of Academic Research," First Monday, Oct. 12, 1999, http://firstmonday.org/issues/issue4_10/bezroukov/index.html#b5.
■ 23. J.S. Brown and P. Duguid, "Organizing Knowledge," California Management Review 40, no. 3 (spring 1998): 90-111.
■ 24. C. Jones, W.S. Hesterly and S.P. Borgatti, "A General Theory of Network Governance: Exchange Conditions and Social Mechanisms," Academy of Management Review 22 (Oct. 4, 1997): 911-945.
■ 25. S. Turkle, "The Second Self: Computers and the Human Spirit" (New York: Simon and Schuster, 1984).

Reprint 4211