7 Innovation Communities

It is now clear that users often innovate, and that they often freely reveal their innovations. But what about informal cooperation among users? What about *organized* cooperation in development of innovations and other matters? The answer is that both flourish among user-innovators. Informal user-to-user cooperation, such as assisting others to innovate, is common. Organized cooperation in which users interact within communities, is also common. Innovation communities are often stocked with useful tools and infrastructure that increase the speed and effectiveness with which users can develop and test and diffuse their innovations.

In this chapter, I first show that user innovation is a widely distributed process and so can be usefully drawn together by innovation communities. I next explore the valuable functions such communities can provide. I illustrate with a discussion of free and open source software projects, a very successful form of innovation community in the field of software development. Finally, I point out that innovation communities are by no means restricted to the development of information products such as software, and illustrate with the case of a user innovation community specializing in the development of techniques and equipment used in the sport of kitesurfing.

User Innovation Is Widely Distributed

When users' needs are heterogeneous and when the information drawn on by innovators is sticky, it is likely that product-development activities will be widely distributed among users, rather than produced by just a few prolific user-innovators. It should also be the case that different users will tend to develop different innovations. As was shown in chapter 5, individual

From Democratizing Innovation by Eric von Hippel - MIT Press, 2005
Free Download Under Creative Commons License at http://web.mit.edu/evhippel/www

users and user firms tend to develop innovations that serve their particular needs, and that fall within their individual "low-cost innovation niches." For example, a mountain biker who specializes in jumping from high platforms and who is also an orthopedic surgeon will tend to develop innovations that draw on both of these types of information: he might create a seat suspension that reduces shock to bikers' spines upon landing from a jump. Another mountain biker specializing in the same activity but with a different background—say aeronautical engineering—is likely to draw on this different information to come up with a different innovation. From the perspective of Fleming (2001), who has studied innovations as consisting of novel combinations of pre-existing elements, such innovators are using their membership in two distinct communities to combine previously disparate elements. Baldwin and Clark (2003) and Henkel (2004a) explore this type of situation in theoretical terms.

The underlying logic echoes that offered by Eric Raymond regarding "Linus's Law" in software debugging. In software, discovering and repairing subtle code errors or bugs can be very costly (Brooks 1979). However, Raymond argued, the same task can be greatly reduced in cost and also made faster and more effective when it is opened up to a large community of software users that each may have the information needed to identify and fix *some* bugs. Under these conditions, Raymond says, "given a large enough beta tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone. Or, less formally, 'given enough eyeballs, all bugs are shallow.'" He explains: "More users find more bugs because adding more users adds more ways of stressing the program. . . . Each [user] approaches the task of bug characterization with a slightly different perceptual set and analytical toolkit, a different angle on the problem. So adding more beta-testers . . . increases the probability that someone's toolkit will be matched to the problem in such a way that the bug is shallow to *that person*." (1999, pp. 41–44)

The analogy to distributed user innovation is, of course, that each user has a different set of innovation-related needs and other assets in place which makes a particular type of innovation low-cost ("shallow") to *that user*. The assets of *some* user will then generally be found to be a just-right fit to many innovation development problems. (Note that this argument does not mean that *all* innovations will be cheaply done by users, or even

Table 7.1
User innovation is widely distributed, with few users developing more than one major innovation. NA: data not available.

	Number of users developing this number of major innovations					
	1	2	3	6	NA	Sample (*n*)
Scientific Instruments[a]	28	0	1	0	1	32
Scientific Instruments[b]	20	1	0	1	0	28
Process equipment[c]	19	1	0	0	8	29
Sports equipment[d]	7	0	0	0	0	7

a. Source: von Hippel 1988, appendix: GC, TEM, NMR Innovations.
b. Source: Riggs and von Hippel, Esca and AES.
c. Source: von Hippel 1988, appendix: Semiconductor and pultrusion process equipment innovations.
d. Source: Shah 2000, appendix A: skateboarding, snowboarding, and windsurfing innovations.

done by users at all. In essence, users will find it cheaper to innovate when manufacturers' economies of scale with respect to product development are more than offset by the greater scope of innovation assets held by the collectivity of individual users.)

Available data support these expectations. In chapter 2 we saw evidence that users tended to develop very different innovations. To test whether commercially important innovations are developed by just a few users or by many, I turn to studies documenting the functional sources of important innovations later commercialized. As is evident in table 7.1, most of the important innovations attributed to users in these studies were done by *different* users. In other words, user innovation does tend to be widely distributed in a world characterized by users with heterogeneous needs and heterogeneous stocks of sticky information.

Innovation Communities

User-innovators may be generally willing to freely reveal their information. However, as we have seen, they may be widely distributed and each may have only one or a few innovations to offer. The practical value of the "freely revealed innovation commons" these users collectively offer will be increased if their information is somehow made conveniently accessible. This is one of the important functions of "innovation communities."

I define "innovation communities" as meaning nodes consisting of individuals or firms interconnected by information transfer links which may involve face-to-face, electronic, or other communication. These can, but need not, exist within the boundaries of a membership group. They often do, but need not, incorporate the qualities of communities for participants, where "communities" is defined as meaning"networks of interpersonal ties that provide sociability, support, information, a sense of belonging, and social identity" (Wellman et al. 2002, p. 4).[1]

Innovation communities can have users and/or manufacturers as members and contributors. They can flourish when at least some innovate and voluntarily reveal their innovations, and when others find the information revealed to be of interest. In previous chapters, we saw that these conditions do commonly exist with respect to user-developed innovations: users innovate in many fields, users often freely reveal, and the information revealed is often used by manufacturers to create commercial products—a clear indication many users, too, find this information of interest.

Innovation communities are often specialized, serving as collection points and repositories for information related to narrow categories of innovations. They may consist only of information repositories or directories in the form of physical or virtual publications. For example, userinnovation.mit.edu is a specialized website where researchers can post articles on their findings and ideas related to innovation by users. Contributors and non-contributors can freely access and browse the site as a convenient way to find such information.

Innovation communities also can offer additional important functions to participants. Chat rooms and email lists with public postings can be provided so that contributors can exchange ideas and provide mutual assistance. Tools to help users develop, evaluate, and integrate their work can also be provided to community members—and such tools are often developed by community members themselves.

All the community functionality just mentioned and more is visible in communities that develop free and open source software programs. The emergence of this particular type of innovation community has also done a great deal to bring the general phenomenon to academic and public notice, and so I will describe them in some detail. I first discuss the history and nature of free and open source software itself (the product). Next I outline key characteristics of the free and open source software development projects typically used to create and maintain such software (the community-based development process).

Open Source Software

In the early days of computer programming, commercial "packaged" software was a rarity—if you wanted a particular program for a particular purpose, you typically wrote the code yourself or hired someone to write it for you. Much of the software of the 1960s and the 1970s was developed in academic and corporate laboratories by scientists and engineers. These individuals found it a normal part of their research culture to freely give and exchange software they had written, to modify and build on one another's software, and to freely share their modifications. This communal behavior became a central feature of "hacker culture." (In communities of open source programmers, "hacker" is a positive term that is applied to talented and dedicated programmers.[2])

In 1969, the Defense Advanced Research Projects Agency, a part of the US Department of Defense, established the ARPANET, the first transcontinental high-speed computer network. This network eventually grew to link hundreds of universities, defense contractors, and research laboratories. Later succeeded by the Internet, it also allowed hackers to exchange software code and other information widely, easily, and cheaply—and also enabled them to spread hacker norms of behavior.

The communal hacker culture was very strongly present among a group of programmers—software hackers—housed at MIT's Artificial Intelligence Laboratory in the 1960s and the 1970s (Levy 1984). In the 1980s this group received a major jolt when MIT licensed some of the code created by its hacker employees to a commercial firm. This firm, in accordance with normal commercial practice, then promptly restricted access to the "source code"[3] of that software, and so prevented non-company personnel—including the MIT hackers who had been instrumental in developing it—from continuing to use it as a platform for further learning and development.

Richard Stallman, a brilliant programmer in MIT's Artificial Intelligence Laboratory, was especially distressed by the loss of access to communally developed source code. He also was offended by a general trend in the software world toward development of proprietary software packages and the release of software in forms that could not be studied or modified by others. Stallman viewed these practices as morally wrong impingements on the rights of software users to freely learn and create. In 1985, in response, he founded the Free Software Foundation and set about to develop and diffuse a legal mechanism that could preserve free access for all to the software developed by software hackers. Stallman's pioneering idea was to use the existing mechanism of copyright law to this end. Software authors interested in preserving the status of their software as "free" software could use their own copyright to grant licenses on terms that would guarantee a number of rights to all future users. They could do this by simply affixing a standard license to their software that conveyed these rights. The basic license developed by Stallman to implement this seminal idea was the General Public License or GPL (sometimes referred to as copyleft, in a play on the word "copyright"). Basic rights transferred to those possessing a copy of free software include the right to use it at no cost, the right to study its source code, the right to modify it, and the right to distribute modified or unmodified versions to others at no cost. Licenses conveying similar rights were developed by others, and a number of such licenses are currently used in the open source field. Free and open source software licenses do not grant users the full rights associated with free revealing as that term was defined earlier. Those who obtain the software under a license such as the GPL are restricted from certain practices. For example, they cannot incorporate GPL software into proprietary software that they then sell.[4] Indeed, contributors of code to open source software projects are very concerned with enforcing such restrictions in order to ensure that their code remains accessible to all (O'Mahony 2003).

The idea of free software did not immediately become mainstream, and industry was especially suspicious of it. In 1998, Bruce Perens and Eric Raymond agreed that a significant part of the problem resided in Stallman's term "free" software, which might understandably have an ominous ring to the ears of businesspeople. Accordingly, they, along with other prominent hackers, founded the open source software movement (Perens 1999). Open source software uses the licensing practices pioneered by the free software movement. It differs from that movement primarily on philosophical grounds, preferring to emphasize the practical benefits of its licensing practices over issues regarding the moral importance of granting users the freedoms offered by both free and open source software. The term “open source” is now generally used by both practitioners and scholars to refer to free or open source software, and that is the term I use in this book.

Open source software has emerged as a major cultural and economic phenomenon. The number of open source software projects has been growing rapidly. In mid 2004, a single major infrastructure provider and repository for open source software projects, Sourceforge.net,[5] hosted 83,000 projects and had more than 870,000 registered users. A significant amount of software developed by commercial firms is also being released under open source licenses.

Open Source Software Development Projects

Software can be termed “open source” independent of how or by whom it has been developed: the term denotes only the type of license under which it is made available. However, the fact that open source software is freely accessible to all has created some typical open source software development practices that differ greatly from commercial software development models—and that look very much like the “hacker culture” behaviors described above.

Because commercial software vendors typically wish to sell the code they develop, they sharply restrict access to the source code of their software products to firm employees and contractors. The consequence of this restriction is that only insiders have the information required to modify and improve that proprietary code further (Meyer and Lopez 1995; Young, Smith, and Grimm 1996; Conner and Prahalad 1996). In sharp contrast, all are offered free access to the source code of open source software if that code is distributed by its authors. In early hacker days, this freedom to learn and use and modify software was exercised by informal sharing and codevelopment of code—often by the physical sharing and exchange of computer tapes and disks on which the code was recorded. In current Internet days, rapid technological advances in computer hardware and software and networking technologies have made it much easier to create and sustain a communal development style on ever-larger scales. Also, implementing new projects is becoming progressively easier as effective project design becomes better understood, and as prepackaged infrastructural support for such projects becomes available on the Web.

Today, an open source software development project is typically initiated by an individual or a small group seeking a solution to an individual's or a firm's need. Raymond (1999, p. 32) suggests that "every good work of software starts by scratching a developer's personal itch" and that "too often software developers spend their days grinding away for pay at programs they neither need nor love. But not in the (open source) world. . . ." A project's initiators also generally become the project's "owners" or "maintainers" who take on responsibility for project management.[6] Early on, this individual or group generally develops a first, rough version of the code that outlines the functionality envisioned. The source code for this initial version is then made freely available to all via downloading from an Internet website established by the project. The project founders also set up infrastructure for the project that those interested in using or further developing the code can use to seek help, provide information or provide new open source code for others to discuss and test. In the case of projects that are successful in attracting interest, others do download and use and "play with" the code—and some of these do go on to create new and modified code. Most then post what they have done on the project website for use and critique by any who are interested. New and modified code that is deemed to be of sufficient quality and of general interest by the project maintainers is then added to the authorized version of the code. In many projects the privilege of adding to the authorized code is restricted to only a few trusted developers. These few then serve as gatekeepers for code written by contributors who do not have such access (von Krogh and Spaeth 2002).

Critical tools and infrastructure available to open source software project participants includes email lists for specialized purposes that are open to all. Thus, there is a list where code users can report software failures ("bugs") that they encounter during field use of the software. There is also a list where those developing the code can share ideas about what would be good next steps for the project, good features to add, etc. All of these lists are open to all and are also publicly archived, so anyone can go back and learn what opinions were and are on a particular topic. Also, programmers contributing to open source software projects tend to have essential tools, such as specific software languages, in common. These are generally not specific to a single project, but are available on the web. Basic toolkits held in common by all contributors tends to greatly ease interactions. Also, open source software projects have version-control software that allows contributors to insert new code contributions into the existing project code base and test them to see if the new code causes malfunctions in existing code. If so, the tool allows easy reversion to the status quo ante. This makes "try it and see" testing much more practical, because much less is at risk if a new contribution inadvertently breaks the code. Toolkits used in open source projects have been evolved through practice and are steadily being improved by user-innovators. Individual projects can now start up using standard infrastructure sets offered by sites such as Sourceforge.net.

Two brief case histories will help to further convey the flavor of open source software development.

Apache Web Server Software

Apache web server software is used on web server computers that host web pages and provide appropriate content as requested by Internet browsers. Such[7] computers are a key element of the Internet-based World Wide Web infrastructure.

The web server software that evolved into Apache was developed by University of Illinois undergraduate Rob McCool for, and while working at, the National Center for Supercomputing Applications (NCSA). The source code as developed and periodically modified by McCool was posted on the web so that users at other sites could download it, use it, modify it, and develop it further. When McCool departed NCSA in mid 1994, a small group of webmasters who had adopted his web server software for their own sites decided to take on the task of continued development. A core group of eight users gathered all documentation and bug fixes and issued a consolidated patch. This "patchy" web server software evolved over time into Apache. Extensive user feedback and modification yielded Apache 1.0, released on December 1, 1995.

In 4 years, after many modifications and improvements contributed by many users, Apache became the most popular web server software on the Internet, garnering many industry awards for excellence. Despite strong competition from commercial software developers such as Microsoft and Netscape, it is currently used by over 60 percent of the world's millions of websites. Modification and updating of Apache by users and others continues, with the release of new versions being coordinated by a central group of 22 volunteers.

Fetchmail—An Internet Email Utility Program

Fetchmail is an Internet email utility program that "fetches" email from central servers to a local computer. The open source project to develop, maintain, and improve this program was led by Eric Raymond (1999).

Raymond first began to puzzle about the email delivery problem in 1993 because he was personally dissatisfied with then-existing solutions. "What I wanted," Raymond recalled (1999, p. 31), "was for my mail to be delivered on snark, my home system, so that I would be notified when it arrived and could handle it using all my local tools." Raymond decided to try and develop a better solution. He began by searching databases in the open source world for an existing, well-coded utility that he could use as a development base. He knew it would be efficient to build on others' related work if possible, and in the world of open source software (then generally called free software) this practice is understood and valued. Raymond explored several candidate open source programs, and settled on one in small-scale use called "popclient." He developed a number of improvements to the program and proposed them to the then maintainer of popclient. It turned out that this individual had lost interest in working further on the program, and so his response to Raymond's suggestions was to offer his role to Raymond so that he could evolve the popclient further as he chose.

Raymond accepted the role of popclient's maintainer, and over the next months he improved the program significantly in conjunction with advice and suggestions from other users. He carefully cultivated his more active beta list of popclient users by regularly communicating with them via messages posted on an public electronic bulletin board set up for that purpose. Many responded by volunteering information on bugs they had found and perhaps fixed, and by offering improvements they had developed for their own use. The quality of these suggestions was often high because "contributions are received not from a random sample, but from people who are interested enough to use the software, learn about how it works, attempt to find solutions to the problems they encounter, and actually produce an apparently reasonable fix. Anyone who passes all these filters is highly likely to have something useful to contribute." (ibid., p. 42)

Eventually, Raymond arrived at an innovative design that he knew worked well because he and his beta list of co-developers had used it, tested it and improved it every day. Popclient (now renamed fetchmail) became standard software used by millions users. Raymond continues to lead the group of volunteers that maintain and improve the software as new user needs and conditions dictate.

Development of Physical Products by Innovation Communities

User innovation communities are by no means restricted to the development of information products like software. They also are active in the development of physical products, and in very similar ways. Just as in the case of communities devoted to information product, communities devoted to physical products can range from simple information exchange sites to sites well furnished with tools and infrastructure. Within sports, Franke and Shah's study illustrates relatively simple community infrastructure. Thus, the boardercross community they studied consisted of semi-professional athletes from all over the world who meet in up to 10 competitions a year in Europe, North America, and Japan. Franke and Shah report that community members knew one another well, and spent a considerable amount of time together. They also assisted one another in developing and modifying equipment for their sport. However, the community had no specialized sets of tools to support joint innovation development.

More complex communities devoted to the development of physical products often look similar to open source software development communities in terms of tools and infrastructure. As an example, consider the recent formation of a community dedicated to the development and diffusion of information regarding novel kitesurfing equipment. Kitesurfing is a water sport in which the user stands on a special board, somewhat like a surfboard, and is pulled along by holding onto a large, steerable kite. Equipment and technique have evolved to the point that kites can be guided both with and against the wind by a skilled kitesurfer, and can lift rider and board many meters into the air for tens of seconds at a time.

Designing kites for kitesurfing is a sophisticated undertaking, involving low-speed aerodynamical considerations that are not yet well understood. Early kites for kitesurfing were developed and built by user-enthusiasts who were inventing both kitesurfing techniques and kitesurfing equipment interdependently. In about 2001, Saul Griffith, an MIT PhD student with a long-time interest in kitesurfing and kite development, decided that kitesurfing would benefit from better online community interaction. Accordingly, he created a site for the worldwide community of user-innovators in kitesurfing (www.zeroprestige.com). Griffith began by posting patterns for kites he had designed on the site and added helpful hints and tools for kite construction and use. Others were invited to download this information for free and to contribute their own if they wished. Soon other innovators started to post their own kite designs, improved construction advice for novices, and sophisticated design tools such as aerodynamics modeling software and rapid prototyping software. Some kitesurfers contributing innovations to the site had top-level technical skills; at least one was a skilled aerodynamicist employed by an aerospace firm.

Note that physical products are information products during the design stage. In earlier days, information about an evolving design was encoded on large sheets of paper, called blueprints, that could be copied and shared. The information on blueprints could be understood and assessed by fellow designers, and could also be used by machinists to create the actual physical products represented. Today, designs for new products are commonly encoded in computer-aided design (CAD) files. These files can be created and seen as two-dimensional and three-dimensional renderings by designers. The designs they contain can also be subjected to automated analysis by various engineering tools to determine, for example, whether they can stand up to stresses to which they will be subjected. CAD files can then be downloaded to computer-controlled fabrication machinery that will actually build the component parts of the design.

The example of the kitesurfing group's methods of sharing design information illustrates the close relationship between information and physical products. Initially, users in the group exchanged design ideas by means of simple sketches transferred over the Internet. Then group members learned that computerized cutters used by sail lofts to cut sails from large pieces of cloth are suited to cutting cloth for surfing kites. They also learned that sail lofts were interested in their business. Accordingly, innovation group members began to exchange designs in the form of CAD files compatible with sail lofts' cutting equipment. When a user was satisfied with a design, he would transmit the CAD file to a local sail loft for cutting. The pieces were then sewn together by the user or sent to a sewing facility for assembly. The total time required to convert an information product into a physical one was less than a week, and the total cost of a finished kite made in this way was a few hundred dollars—much less than the price of a commercial kite.

User-to-User Assistance

Clearly, user innovation communities can offer sophisticated support to individual innovators in the form of tools. Users in these innovation communities also tend to behave in a collaborative manner. That is, users not only distribute and evaluate completed innovations; they also volunteer other important services, such as assisting one another in developing and applying innovations.

Franke and Shah (2003) studied the frequency with which users in four sporting communities assisted one another with innovations, and found that such assistance was very common (table 7.2). They also found that those who assisted were significantly more likely to be innovators themselves (table 7.3). The level of satisfaction reported by those assisted was very high. Seventy-nine percent agreed strongly with the statement "If I had a similar problem I would ask the same people again." Jeppesen (2005) similarly found extensive user-to-user help being volunteered in the field of computer gaming.

Table 7.2
Number of people from whom innovators received assistance.

Number of people	Number of cases	Percentage
0	0	0
1	3	6
2	14	26
3–5	25	47
6–10	8	15
> 10	3	6
Total	53	100

Source: Franke and Shah 2003, table 4.

Table 7.3
Innovators tended to be the ones assisting others with their innovations ($p < 0.0001$).

	Innovators	Non-innovators	Total
Gave assistance	28	13	41
Did not give assistance	32	115	147
Total	60	128	

Source: Franke and Shah 2003, table 7.

Such helping activity is clearly important to the value contributed by innovation communities to community participants. Why people might voluntarily offer assistance is a subject of analysis. The answers are not fully in, but the mysteries lessen as the research progresses. An answer that appears to be emerging is that there are private benefits to assistance providers, just as there are for those who freely reveal innovations (Lakhani and von Hippel 2003). In other words, provision of free assistance may be explicable in terms of the private-collective model of innovation-related incentives discussed earlier.