7 Open Source Software

7.1 Overview

Software providers in the narrower sense create software in order to generate license sales and in some cases, revenue from services. A different motivation lies behind open source projects. Software developers come together in an international community to pool their knowledge and jointly solve a problem. In this scenario, many developers invest their time, normally without being paid. But it does not follow that open source software (OSS) is irrelevant in an economic sense. In this chapter, we will explore the fundamental questions that OSS poses to the software industry and to users.

We will begin by briefly introducing the nature and features of open source software (OSS) and exploring the origins of the open source movement. We will then examine the development process in open source projects and how it differs from the process in a traditional software company. We will also consider what motivates developers to become involved in open source projects in the first place. Furthermore, we will look at the introduction of OSS from a user perspective. We will then provide an overview of commercial software providers' strategies—in terms of opposing and utilizing OSS. We will conclude with some thoughts and early empirical results on the use of open source business apps.

7.2 Features of Open Source Software

Free software has been around for a long time. Private users have frequently taken advantage of freeware, for example certain database systems or games. The methods of distribution have evolved over time: during the early years of the personal computer, freeware was exchanged via floppy disk, then via CDs, and nowadays almost exclusively over the Internet. Programmers have always been in the habit of sharing their source codes and programs to help and learn from one another.

© Springer-Verlag Berlin Heidelberg 2013

In the 1970s, some companies began to sell only compiled software, and keep their source code under lock and key. A movement opposed to this practice evolved, and one of its pioneers was Richard Stallman. He began his academic career in 1971 at MIT's Artificial Intelligence Laboratory. The following quote from Stallman paints a picture of the culture that prevailed there at the time (Grassmuck 2004, p. 219):

"I had the good fortune in the 1970s to be a part of a community in which people shared software. We were developing software, and whenever somebody wrote an interesting program it would circulate around. You could run the program, add features, or just read the code and see how problems were solved. If you added features to the program then other people could use the improved version. So one person after another would work to improve the software and develop it further. You could always expect at least the passive cooperation of everybody else in this community. They might not be willing to drop their work and spend hours doing something for you, but whatever they had already done, if you could get some use out of it, you were welcome to do so."

As the quote shows, free software is by no means a phenomenon of the 1990s. Many people viewed sharing software code and knowledge as a matter of course.

In the end, dissatisfaction with the functionality of a printer driver sparked the development of the open source movement (Grassmuck 2004, p. 222), although it was not yet known by that name. The Xerox network printers at MIT had no function for displaying printer status directly on the PC. Stallman wanted to write a function to make this possible, and embed it in the printer driver's source code. But the Xerox employee responsible for the code refused to release it, because he had signed an undertaking not to share it with any third parties.

This prompted Stallman to do two things: develop the driver himself, and set up the GNU project (GNU is a recursive acronym for GNU's Not Unix). In order to prevent others from making commercial use of his work, Stallman resigned from MIT. To make a living and ensure the continuation of the GNU project, he founded the free software foundation. This organization collected donations, charged fees for the distribution of GNU software on data media complete with manuals, but not for the software itself, and hired developers. A complete GNU/Linux operating system was created in the early 1990s by combining the components of the GNU project with a Linux kernel.

Free system components were not the only fruits of the GNU project: it also produced the GNU general public license (GPL), a special software license that has had a substantial influence on the free software and open source movements. The GPL grants users free access to the source code, the right to copy and share the software, freedom to modify the code, and permission to distribute the modified version, albeit under the same terms.

From an economic perspective, this last condition rules out the possibility of later changes to the software's property rights. Rather than, the software developer waving his intellectual property rights (which is standard practice with public

Table 7.1 Selected features of some open source licenses (adapted from Perens 1999, p. 186)

Type of license	GPL	LGPL	MPL	BSD license
Can be integrated into proprietary software and redistributed without an OS software license	No	Yes	Yes	Yes
Modifications to OS licensed source code can remain proprietary on distribution	No	No	No	Yes

domain software), the so-called copyleft principle is applied, to guarantee the software remains permanently free. In accordance with this principle, modified versions of the software have to be subject to the same license.

In addition to GPL, there are many other open source licenses, such as the less restrictive library/lesser general public license (LGPL, originally developed for libraries), the Berkeley software distribution-style license (BSD-style license) and the mozilla public license (MPL). Table 7.1 summarizes selected features of these licenses.

The term "open source" was not coined until 1998, when the open source initiative (OSI) was founded. Until that point, free software had been the standard name. But the change was more than just linguistic. Eric S. Raymond, known principally for his essay, "The Cathedral and the Bazaar" (Raymond 1999), in which he compared a centrally run software project to the construction of a cathedral and the decentralized organization of a project in the Linux community to a bazaar—was not the only one interested in a realignment of the free software movement. Software companies also displayed an interest. One reason for founding the OSI was Netscape's announcement of its intention to publish its browser's source code.

In summary, the aim of OSI's founders was to set the free software movement on a new course. A key goal was to improve cooperation with software companies. Free software was renamed "open source" as a way to "market the free software concept to the people who wore suits" (Perens 1999, p. 173). Volker Grassmuck has commented that some developers probably feared that the word "free" could cause misunderstandings and could be interpreted as a communist "four-letter word" (Grassmuck 2004, p. 230).

Such considerations prompted more than just a change of name: the group wrote a definition of open source based on the work of Bruce Perens, the former project leader of Debian GNU/Linux. The definition included several criteria that must be met for software to be classed as open source. The criteria are listed in the box below.

The OSI open source definition

Open source does not just mean access to the source code. The distribution terms of open source software must comply with the following criteria:

1. *Free redistribution*

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. *Source code*

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of downloading the source code, without charge, via the Internet. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms, such as the output of a preprocessor or translator, are not allowed.

3. *Derived software*

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. *Integrity of the author's source code*

The license may restrict source code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. *No discrimination against persons or groups*

The license must not discriminate against any person or group of persons.

6. *No discrimination against fields of endeavor*

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. *Distribution of license*

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. *License must not be specific to a product*

   The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. *License must not restrict other software*

   The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open source software.

© 2011 Open Source Initiative. Opensource.org site content is licensed under a Creative Commons Attribution Noncommercial No Derivatives License (creativecommons.org/licenses/by-nc-nd/2.5/legalcode).

This definition is not itself a license but a standard against which licenses are measured. In effect, the OSI assumes the role of a certification authority. So far, more than 60 licenses have been certified, including the GNU GPL, GNU LGPL, MPL, and the New BSD License. A current list can be found at http://www.zopensource.org/licenses.

Some of these licenses make it easier for software providers to privatize or commercialize OSS. This is why the OSI's open source definition is controversial and has sparked so many debates. For example, the first criterion on the above list does not rule out the use of open source code in commercial software packages. This is how a BSD license enabled Microsoft to integrate open source code into Windows. If the code had been subject to a GPL, Microsoft would not have been permitted to use it without Windows becoming free software (Grassmuck 2004, p. 299).

As shown above, the GPL also prevents the privatization of modified codes, whereas under the terms of Apache or BSD licenses this is permissible (Grassmuck 2004, p. 301). A modified version of OSS can, then, be sold without having to release the source code.

It quickly becomes apparent that many software companies can profit from the OSI‘s new approach. GPL can prove to be something of a hurdle to software projects (even noncommercial ones); for instance, if open source code has to be integrated into commercial products. In response, the LGPL was developed to make code sharing more attractive by allowing libraries to be integrated more easily. Integrating a GPL licensed library into a software program would mean that the entire software would have to be subject to the GPL. Since this is often not what programmers want, and in order to incentivize developers to use free libraries, the LGPL relaxes this condition (Grassmuck 2004, p. 290).

The less restrictive software licenses that simplify the privatization and commercialization of open source code are a double-edged sword. On the one hand, it could be argued that commercialization is not necessarily financially damaging to open source developers and that many software projects reap its benefits. On the other hand, many of those involved in open source projects are sure to regard this as an injustice and an attempt by companies to get something for nothing, which could affect their willingness to participate in these projects. We will come back to these incentives later, but first we will turn to the development principles that shape open source projects.

7.3 Open Source Projects: Principles and Motivation of Software Developers

The process by which, open source software is developed, is very different to software development in a commercial company. In the following section, we will examine the main differences and how they affect the structure of a project and the motivation of those involved in it.

7.3.1 Organizational Structures and Processes in Open Source Projects

An open source project usually comes about when somebody would like to solve a problem. In the previous section, we described how Stallman's dissatisfaction with a Xerox printer driver was the starting point for the GNU project. Another oft-cited example is the development of the LINUX operating system. Linus Torvalds wanted to run a Unix operating system on his 386 PC. Finding nothing suitable, he began to develop his own and published his source code on the Internet. As we all know, the project met with keen interest and a number of software developers became involved in developing it. Even Torvalds was surprised by this turn of events and has repeatedly stressed that he had never dreamed that it would be so successful.

The development of OSS is an evolutionary and distributed process. Raymond evocatively represents the development process as a bazaar and contrasts this with the so-called cathedral model, synonymous in his eyes with conventional software development (Raymond 1999). If commercial software development is viewed in the context of early models from the field of software engineering, then the two approaches differ widely. However, concepts of evolutionary and of agile software development are similar to elements of the open source movement's approach (Sharma et al. 2002).

Once a project has been set up, its success depends on the prompt establishment of a community around the software. It is always helpful if some modules are already available for testing and execution.

The development team's decision-making structure and the composition of the team itself are centrally important. Large-scale open source projects usually have a coreteam composed of the developers who have been working on the project the longest, or who have contributed a large quantity of code. The core team of the Apache web server project comprised 22 programmers from six countries (Grassmuck 2004, p. 237).

In small-scale open source projects, the founder usually assumes the role of "maintainer". He assumes responsibility for project coordination and quality control. Developers with a good track record often act as maintainers in large projects, where a two-tiered system lets them coordinate module development and have a say in decisions on general principle in the overall project. It is quite often the case that the founding member has a special role in the core team. The Linux project, for instance, had a team of five or six developers who tested and selected incoming source code before passing it on to Torvalds, who would make the final decision (Dietrich 1999). In contrast, the Apache project's core team takes a democratic approach to decision making. The decision to integrate a module or not may be decided, for example, via mailing list.

The maintainers and software developers who contribute a great deal of code to the project are supported by a host of other people, who test the software, write the documentation, and provide localizations. It is often difficult to find qualified software developers to perform these supporting roles. Most developers regard documentation as boring, and in any case it is not the best way to boost one's reputation.

Open source projects tend not to be wound up like conventional ones. Instead, development work may be discontinued once the user is satisfied with the solution, or if either the maintainer or key developers have lost interest in the project. If a maintainer becomes inactive, the development community can appoint a new one. Occasionally, a project splits, or "forks". Forking happens when the core team can no longer agree on questions of principle.

Internet-based source code management systems such as concurrent versions system (CVS)) serve as an important function in open source projects. Programmers download the latest version of a module from the CVS, work on it, test it with their own development tools, and copy the results back to the CVS repository. If several developers have been working on a file at the same time, their changes are ideally merged into a new file in the repository. If this causes conflicts that cannot be resolved automatically, developers must come to an agreement amongst themselves. At regular intervals, the core team flags certain branches of the source tree in the repository as a new release.

7.3.2 Contributor Motivation

Open source projects are based on the joint work of a frequently global community of software developers. They participate on a voluntary basis and the majority are not paid for their work. Their involvement implies acceptance of opportunity costs, from sacrificing leisure time, to passing up alternative paid positions, to neglecting their day job. This is particularly relevant to the core team members in open source projects.

This raises the question of what motivates developers to participate in open source projects. Some authors try to explain the phenomenon through the personal gains a developer can make with his contribution (Lerner and Tirole 2002). Other studies assume that most programmers are driven by intrinsic motivations (Kollock 1999). Against this backdrop, Franck (2003) distinguishes between

- Rent-seekers and
- Donators.

Rent seekers behave like a conventional “homo economicus”. Rent-seeking developers are looking for benefits beyond regular pay checks. Empirical studies have repeatedly shown that contributors hope to boost their reputation on the job or capital market, expect to improve their know-how, or assume that their activities will help them in their everyday work (Lerner and Tirole 2002). Hence, these developers do not become involved unless they expect a pay-off –in other words, they are seeking “rent.”

Conventional rational motives alone cannot account for the phenomenon of people contributing to open source projects. Time and again, empirical studies have shown that developers hope to get enjoyment and entertainment out of their participation in open source projects, as well as advancing the open source movement. Furthermore, many open source developers are pursuing other goals, such as freedom of information. And very often, their activities are actually an attempt to break market leader Microsoft’s virtual monopoly. Franck terms people for whom this is a top priority “donators” (Franck 2003). They believe that they are investing their time in something worthwhile. For “donators”, open source projects subject to a GPL, or another similar license, are attractive, because they have no reason to fear that their “donation” will be commercialized.

Successful open source projects often manage to create a governance structure that appeals to both rent seekers and donators. The open nature of source code gives rent seekers the chance to improve their reputation, as their contributions are visible and verifiable. The more rent seekers involved, the more appealing an open source project is to donators because the chances of success are higher. This avoids the conflict that is often evident between rent seekers and donators.

However, gone are the days when the teams for most open source projects comprised unpaid hobbyists (Brügge et al. 2004, p. 101 f). Instead, there are a number of open source projects in which salaried programmers and other specialists work alongside each other. Almost 30 % of the developers at open source platform Sourceforge.net, for example, are paid for their work. It is rare for an open source project to be initiated by a commercial enterprise. The Apache project, for instance, was begun by employees who were responsible for their companies’ Web servers. But such a situation does not rule out rent seekers and donators working together. In fact, it may even encourage cooperation, especially in the case of projects subject to a GPL license or similar.

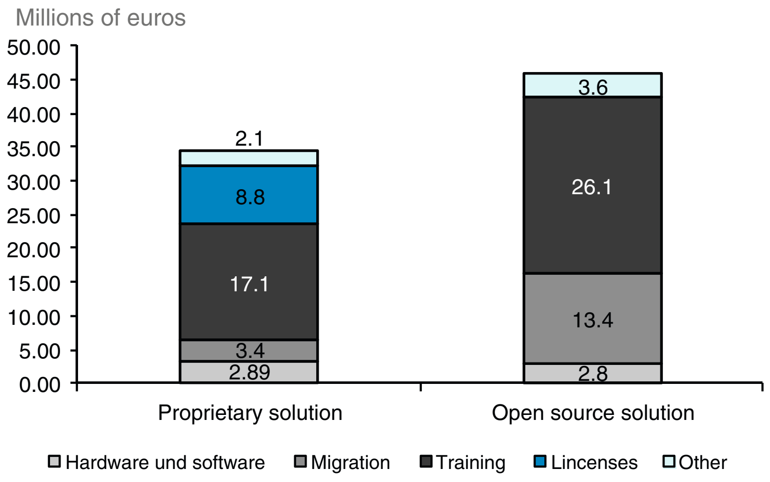

Fig. 7.1 Short-term cost comparison for the city of Munich

7.4 Open Source Software: The User Perspective

OSS has a particularly appealing benefit for users: it is free. This means that users save licensing costs of standard software, and expenses associated with company-specific solutions. But licensing costs are of course not the only relevant parameters that must be considered when choosing which software to deploy.

As an example, let us take a look at the decision by local government in Munich, Germany to replace Windows XP and Office solutions with Linux and open source software. This example was chosen, because the deployment rate of OSS in the public sector is particularly high. The study was carried out by consulting company Unilog in 2002 and considered the direct costs for the organization's 14,700 desktops. It concluded that migrating to an OSS solution is not the best option—at least not in the short term. It should be noted that the study focused only on costs. Although the city's licensing costs would be eliminated, any savings would be wiped out due to significantly higher migration and training expenses. Figure 7.1 shows a breakdown of costs (in millions of euros) for both options.

Upon considering the results of the study, the city of Munich still decided to migrate to the open source solution. It explained its decision by stating that it hoped to significantly reduce its dependence on Microsoft, and that the two solutions were identical in terms of their capabilities and standalone utility.

It is still unclear whether OSS can generally be regarded as cost-effective. Studies on this topic have reached varying conclusions (Brügge et al. 2004). For example, a Berlecon Research study (2002) found that, in addition to eliminating licensing costs, open source solutions have lower implementation and administration costs, and are more stable. But the latter two benefits were not demonstrated in the case of the Munich local government. Such great variance can be partly due to the fact that differentiating between costs is far from simple in practice. In addition, the debate and the studies on this topic are often ideologically biased.

7.5 Commercial Software Vendors' Involvement

In a previous section, we examined why developers and other software specialists participate in open source projects. In this section, we will take a look at what motivates enterprises to support these projects. There are three main reasons (Hecker 1999, Raymond 1999, Henkel 2004):

- Supporting sales of complementary products and services,
- Integrating OSS into own products and
- Reduction of market power of competitors' proprietary software.

The reason most commonly cited in the literature is the possibility of selling complementary products and services. This is a follow-the-free strategy, as described in Sect. 3.3.2.7. To put it another way: OSS creates additional demand on the user side. By meeting this demand, companies can create and leverage indirect network effects. Complementary add-ons can include hardware, software solutions, and services such as training. Linux distributer SuSe's business model is just one example.

Linux distributer SuSe

Linux is a free operating system that supports multitasking and multi-using. The system is now being deployed in many different areas (desktops, servers, mobile telephones, routers, etc.). Ready-made software packages, called distributions, are usually used. SuSe was the first company in Germany to successfully sell a Linux distribution on a broad scale. And it was one of the world's first companies to base its business model on Linux. SuSe penetrated the market with the first German-language installation program for Linux.

Background information

Roland Dyroff, Burchard Steinbild, Hubert Mantel, and Thomas Fehr founded Software und Systementwicklungsgesellschaft mbH (literally: "Software and System Development Company") in 1992. Their first product was merely an enhanced version of an existing Linux distribution; but in 1996, the company introduced the first distribution that it developed itself. The company's headquarters were initially in Fürth, Germany, before it was moved to Nürnberg, Germany in 1998. In 1997, SuSe opened an office in Oakland, USA. Six additional sites in Germany and three international ones (Italy, Czech Republic, UK) followed. In 2004, SuSe was acquired by American company Novell for 210 million dollars. Novell also assumed responsibility for all of Suse's employees around the world, around 380 at the time. That year, SuSe generated revenues of 37 million euros. Today, Novell is one of the world's leading vendors of complete Linux solutions. A key success factor is its numerous strategic partnerships with companies including SAP, Oracle Intel, and IBM.

Service portfolio

Novel's product range includes offerings for both business and private users. The SuSe Linux Enterprise 11 platform is an end-to-end solution for enterprises. It comprises a database, server, desktop, and hardware management. OpenSuSe, an open source solution, is the company's offering for private users. Originally called SuSe-Linux, it was renamed to openSuSe in December 2006 to differentiate it from nonfree products.

Novell's main source of revenues is the provision of services. This is why its portfolio includes a comprehensive range of support, consulting, and training offerings, the most important being its certified courses for enterprises. To underline its service centricity, the company offers numerous free services such as a discussion forum, databases, and documentation that help users find answers to their questions.

Sources www.novell.com, www.opensuse.org

Companies quite often incorporate OSS in their products and solutions. Therefore, it makes sense for them to support their development financially. Integrating OSS in embedded systems is an example. This approach is characterized by a dedicated connection between hardware and software components that can only be used for one specific purpose. Machine control is a typical example. These systems often leverage Linux and other OSS. A high-profile example of a device featuring an embedded system of this kind is TiVO, a digital video recorder. If the all-but-invisible software used in a product is licensed under GPL, the provider can only generate revenues by selling (specialized) hardware. At its core, this is another example of a complementary add-on that we discussed above.

A third motivating factor for software companies to actively engage in open source projects and provide manpower for them is limiting competitors' market power. IBM's involvement in the development of Linux is just one example. By doing this, it succeeded in checking Microsoft's dominance in PC-based operating systems—at least to a certain extent. IBM's considerable investment in Linux gained it one significant benefit: reduced dependency on Microsoft as an operating system vendor.

As an alternative to paying employees for their involvement in open source projects, companies can publish the code of software developed in-house and donate it to the OSS community. IBM is one of the enterprises pursuing this strategy. Other well-known examples are Mozilla and OpenOffice, and Sun Microsystems' decision to turn over Java and JDK to the open source community.

7.6 Open Source ERP Systems

Today, when we talk about OSS, it is generally the well-known and successful projects, such as Linux, that spring to mind. And no wonder, more than 30 million copies of this operating system have been deployed worldwide. This corresponds to a 20–25 % market share for server operating systems. Other success stories include Apache Webserver, the Eclipse development environment, and the MySQL database.

A closer look at the above contenders shows that OSS has predominantly taken root at the lower level of the "software stack", where it has become the standard. This leads to the question of whether and to what extent the OS paradigm is also suitable for ERP systems and whether it could compete with established software providers, such as SAP and Oracle, in this market segment.

One of the key issues is whether the ERP segment is alluring enough to attract many OS developers and whether they possess the necessary knowledge of business. A study of the largest repository for OS software, SourceForge.net, clearly demonstrates that developers are definitely interested in business applications. There are over 630 ERP-related projects on the platform. A notable example of how OS software can become established beyond the infrastructure level is the customer relationship management software, SugarCRM (Sterne and Herring 2006). Open source ERP systems have not been very prominent up to now.

A selection of open source and ERP production solutions is shown in the following table (Buxmann and Matz 2009). The table depicts how each of the responsible enterprises is organized, and describes the features of their individual software projects and characteristics of their business models. All of the offerings are international, multilingual projects. OS projects, such as aforementioned SugarCRM, or GnuCash, which supports financial accounting, are not included. These systems only support single functions of an ERP system (Table 7.2).

There are marked differences between the providers' business models: The development of systems, such as ADempiere and webERP, is driven primarily by a committed community of private developers. The full functionality of the software is available under a GPL license, and paid services are marketed exclusively by third-party providers. Other offerings are more commercial in nature. Their business model is similar to traditional software providers. The only difference of note is that, in addition to the commercial software packages, they also provide a version with an OS license, which often only offers basic functionality. Both the provider and the customer can reap benefits from this dual licensing model (Mundhenke 2007, pp. 130–131; Hecker 1999, p. 49). Customers have the opportunity to test and use the free version without restrictions. And, if necessary, they can switch to the proprietary versions, which allow them to leverage richer functionality and services. This way, the provider can simultaneously address different customer groups and expand the installed base. Furthermore, many providers hope that releasing a software version under an OS license will have a positive effect in terms of marketing.

Table 7.2 An overview of open source ERP systems (Buxmann and Matz 2009) [Die Zeilen“Letzte Version” bittewiehierweglassen]

	Adempiere	Compiere	ERP5 express	Openbravo	OpenERP
Provider					
Provider/trademark owner	ADempiere	Compiere, Inc.	Nexedi SA	Openbravo, S.L.	Tiny
Location	USA	USA	France	Spain	Belgium
Website	adempiere.com	compiere.com	erp5.org	openbravo.com	openerp.com
Software project					
Project began	2006	1999	2007	2006	2005
Type	Web application and rich client	Rich client (Web interface is paid software)	Web application	Web application	Web application and rich client
Registered developers at SourceForge	89 (including 9 admins)	75 (including 2 admins)	Not registered	81 (including 15 admins)	Not registered
Programming language	Java	Java, JavaScript	Python	Java, JavaScript	Java, Python
Supported platforms (server-side)	Platform-independent	Platform-independent	Linux, MacOSX, Unix, Windows	BSD, Linux, Solaris,Windows	Linux, MacOSX, Unix, Windows
Supported databases	Oracle, Postgres	Oracle, Postgres	DB2, MySQL, Oracle, Postgres	Oracle, Postgres	Postgres
Business model					
Target group	No information	For organizations of all sizes	For organizations of all sizes	For organizations of all sizes	No information

(continued)

Table 7.2 (continued)

	Adempiere	Compiere	ERP5 express	Openbravo	OpenERP
License	GPL	GPL for Community and standard edition and a commercial license for the professional edition	GPL	Openbravo Public License (based on Mozilla Public License)	GPL
Price discrimination, software	None	Community, standard, and professional edition	None	None	Module-dependent pricing
Price discrimination, services	None	Community, standard, and professional edition	Community version, starter, premium, and elite packs	Community edition, SMB network, Basic network, OEM network	Module-dependent pricing

	OpenTaps	xTuple ERP	SQL-Ledger	LX-Office	webERP
Provider					
Provider/trademark owner	Open source strategies, Inc.	xTuple	DWS Systems, Inc.	Lx-System–Holger Lindemann, and LINET services GbR	Administrator: Phil Daintree
Location	USA	USA	Canada	Germany	New Zealand
Website	opentaps.org	xtuple.com	sql-ledger.org	lx-office.org	weberp.org
Software project					

(continued)

Table 7.2 (continued)

	OpenTaps	xTuple ERP	SQL-Ledger	LX-Office	webERP
Project began	2005	2002	2000	2004	2003
Type	Web application	Rich client	Web application	Web application	Web application
Registered developers at SourceForge	38 (including 1 admin)	39 (including 9 admins)	Not registered	4 (including 3 admins)	9 (including 2 admins)
Programming language	Java	C ++, JavaScript	Perl	Perl, PHP	PHP
Supported platforms (server-side)	Linux, MacOSX, Unix, Windows	Linux, MacOSX, Windows	Platform-independent	Linux, Unix	Platform-independent
Supported databases	MySQL, Postgres	Postgres	Postgres	Postgres	MySQL
Business model					
Target group	No information	SMEs	No information	No information	Small companies
License	Honest public license (based on GPL) and commercial licenses	PostBooks edition under common public attribution license 1.0 and standard and open manufacturing edition under a commercial license	GPL	Artistic License, GPL and LGPL	GPL

(continued)

Table 7.2 (continued)

	OpenTaps	xTuple ERP	SQL-Ledger	LX-Office	webERP
Price discrimination, software	None	PostBooks, standard and open manufacturing edition	None	None	None
Price discrimination, services	The commercial license provides enhanced support options	None	By user group: user, tech, dev	None	None

An empirical study of development activities and forum postings in the SourceForge platform (www.sourceforge.org) offers insight into whether the development of open source ERP software can be driven by a dedicated community of private developers. SourceForge provides developers and users with various tools for communication and software development free of charge. Registered members can take part in projects and use or add to the community's resource pool. Each SourceForge project generally has several forums. Participants can start threads, generally to ask a question, to which registered SourceForge users can then post replies. According to SourceForge, the platform currently hosts over 23,000 OS projects with over 2 million registered users. All projects whose software is governed by an OSI-recognized license can be registered. In collaboration with the University of Notre Dame, Indiana (USA), SourceForge provides data from its project database for research purposes (cf. van Antwerp and Madey 2008). This enables an analysis of the projects above and beyond what the platform's standard statistics tell us.

We will begin by taking a look at the ranking of the most active projects on SourceForge. A SourceForge project's activity level is derived from the number of visitors to the project site, development activity, and communications, for example within the forums. The standard ranking is based on a cumulative analysis of all data since the start of the project.

In this ranking, there are no ERP projects in the top 20 and only six in the top 1,000. However, recent rankings, which only cover the last 7 days, for example, paint a different picture. For the last 2 years or so (as of July 2009) ERP projects have figured high up on the list. For example, the Openbravo project is regularly among the top ranked, often landing in first place. Of course, these rankings say nothing about the quality of the projects' contributions. But they do indicate that, in principle, the OS model appears to be suitable for ERP software, too.

For a more indepth comparison of ERP and other OS projects, we will now examine the development and communication activities in greater detail.

Our sample for this purpose comprises projects that meet the following criteria:

- at least two registered developers are involved,
- the project has existed for at least 1 year and
- the forums contain at least one posting.

These criteria filter out the very newest and the less active projects. They enabled us to identify 208 projects under the aegis of the ERP group. We then selected 208 projects unrelated to ERP at random.

We began the analysis by comparing the size of the communities. Leaving aside projects with only one registered participant, ERP projects have an average of 5.7 registered participants. As Fig. 7.2a shows, between 2 and 10 participants are registered in over 90 % of these projects. About 9 % of the projects have 11 or more contributors. The largest project (Openbravo) has 77.

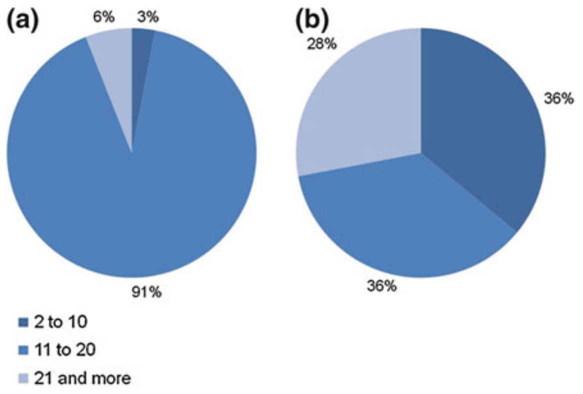

Fig. 7.2 **a** Number of participants in ERP projects; **b** Number of participants in the control group (Buxmann and Matz 2009)

In contrast, the average number of participants in the other, randomly selected projects are 26.7, with a maximum of 391. Figure 7.2b shows that within the projects a similar number of participants are registered in each of the three categories—2 to 10, 11 to 20 and 21 and more. In conclusion, there are significantly fewer users and developers registered in the ERP projects than in the control group.

Next, we will examine and compare how participants communicate in the different types of OS project. We will begin by looking at the forums in ERP projects. Our findings show that around 80 % of users start threads, while some 48 % post replies on the various topics. At least 32 % of threads receive zero replies.

We notice some surprising similarities when we compare these findings to those of the control group, where the proportion of participants who start thread discussions is also 80 %. At the same time, 35 % of participants post replies, which is around 12 % points less than in the ERP project forums. With respect to randomly selected projects, around 72 % of their forum postings stay unanswered. It is hard to tell whether a posting has been answered satisfactorily or whether it is a follow-up question by the thread starter or merely fleshes out the original question. In order to filter out follow-up postings of this kind, we only included postings by users other than the thread starter. Our findings show that less than one-third of users only post replies in other people's threads, but do not start any of their own. Figure 7.3 gives the breakdown.

In summary, we found that users and developers in ERP project forums tended to be more active. In the group of randomly selected projects; for instance, the proportion of forum postings with zero replies was more than twice as large.

We will now examine whether there are any differences with respect to response times—the time lapse between the posting of a question and the first answer (Lee et al. 2009, p. 431). The following chart gives a comparison between the response times for ERP projects and those of other projects. Over 60 % of threads received a reply within 1 day, and 83 % after 1 week. There were no significant differences between the ERP projects and other projects in this respect (Fig. 7.4).

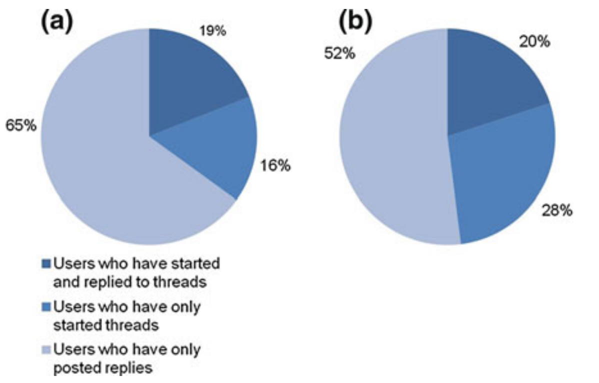

Fig. 7.3 **a** User activity in ERP project forums; **b** User activity in control-group forums (Buxmann and Matz 2009)

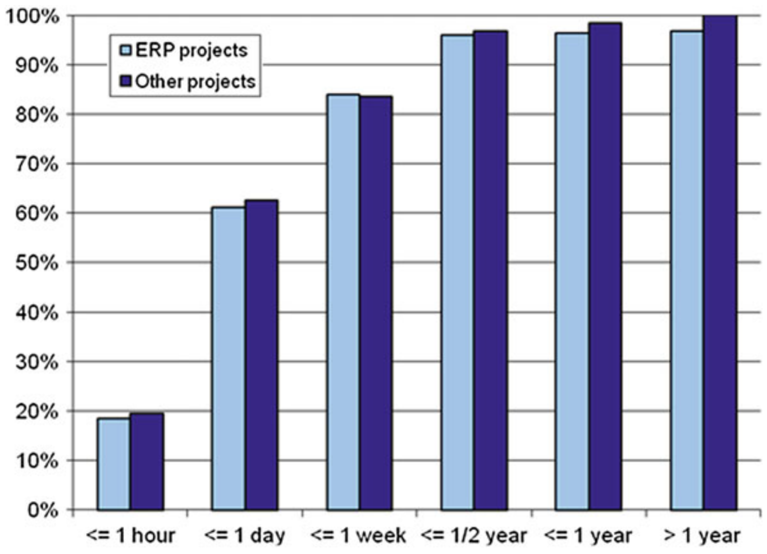

Fig. 7.4 Cumulated response times in ERP project forums and others (Buxmann and Matz 2009)

In conclusion, the OS scene plainly does now offer alternatives to proprietary ERP systems. Our study shows that although there tend to be fewer developers involved in ERP projects, these communicate with each other more intensively, responding to forum threads more willingly and more frequently. It is also noticeable that, ERP projects are increasingly among the most active projects in the SourceForgedatabase recently.

It must be emphasized that these findings say nothing about the quality of work; for example, how useful the posted responses actually were, or how much effort programmers put into the development process. Furthermore, the data we used did not allow us to differentiate between private and paid programmers. Many software and IT enterprises employ large numbers of programmers who spend all or most of their time working on particular OS projects. As a result, the findings of this study may only be interpreted as showing that the OS model can be successful in the ERP space. However, they do not allow us to estimate the market potential of ERP applications. On the contrary, it must be taken into consideration that software markets are subject to special rules and the best solution does not always become the standard. As we discussed in detail in Chap. 2 , there are considerable lock-in-effects on software markets. Due to the penguin effect, changing to a different product incurs high risks and switching costs.

What this means for providers of ERP software is that ultimately, having a good product is not enough. It is vitally important to market software solutions in attractive, customer-friendly packages. Economic simulation models reveal that on software markets, it is more effective to increase market share through pricing than by adding more functionality to a product (Buxmann 2002). The smaller a provider's share of the market in comparison to the market leader, the more this holds true. And it begs the question why some providers are offering their open source ERP software at prices comparable to those of SAP.

Judging by the way some providers present themselves, they still have some catching up to do in terms of professional communications and sales strategies. Against this background, it will not be easy for open source ERP software to capture market share from established ERP providers.

And yet, few experts could have predicted that OS software would become as widespread and popular as it is today in various segments of the operating-system market. Time will tell whether the open source community will shake up the software industry again in the ERP space.