Chapter 1
Open Source Software as Open Innovation: *Experiences from the Medical Domain*

Björn Lundell and Frank van der Linden

Abstract In the past decade, we have witnessed an increased interest amongst commercial and public sector organisations for Open Source Software (OSS). As any individual and organisation has the right to freely read, use, improve and redistribute the source code for software that is developed and released under an OSS licence, it creates new opportunities for Open Innovation. In this chapter, we report on how companies collaborate on production of software artefacts in an OSS project, thereby showing how a form of Open Innovation can be utilised by a large company that goes beyond collaborative development of ideas. In doing so, we report on company decisions and development practices concerning how a software project evolved from proprietary to an open collaborative software development project that is released under an OSS licence (LGPLv2).

1.1 Introduction

Open Source Software (OSS) is software that is licenced and made available under certain "open" conditions, which inherently stimulates a collaborative development of ideas and software artefacts. Anyone who has developed, obtained or adopted such software has the right to freely read, use, improve and redistribute the source code for such software. Over the years, collaboration based on (or stemming from) OSS has influenced many individuals and organisations who have adopted new work practices, and in some cases, even fundamentally changed their way of working.

This chapter gives an illustration of how development of OSS, as an example of Open Innovation, has been adopted by a large company in the secondary software

B. Lundell (✉)
University of Skövde, Informatics Research Centre, Skövde, Sweden

F. van der Linden
Philips Healthcare, Best, The Netherlands

sector. We discuss challenges in software development and present a framework, the commodification diagram, in order to conceptualise experiences from a case study of OSS development in the medical domain. By drawing from the case, we discuss emerging trends and relate these to Open Innovation. The case study was conducted in a large European company and illustrates how emerging commodification trends in the software domain have impacted on company decisions and development practices.

With the adoption of OSS and development practices in companies, the practice for development and deployment of software systems is changing. When companies collaborate on OSS projects, the Open Innovation goes beyond collaborative development of ideas and also includes collaborative production of software artefacts in Open Source projects outside the traditional organisational boundaries. This trend of open collaboration can be seen as an implication of the ongoing trend of contemporary commodification of software which inevitably has consequences for companies leading to new forms of collaborative development.

In the secondary software sector, we have recently seen an increased interest in new forms of development practices, such as inner and open source development of software systems. This form of new development models can be seen as one way by which companies in this sector can deal with the contemporary commodification of software. However, for large companies, it may not be so easy to change established traditions and current work practices. Consequently, adoption of new principles and practices for software development certainly imposes new challenges. In this chapter, we comment on these by drawing from a specific case, which we then relate to the broader picture of Open Innovation. Our case study, stemming from the medical domain, gives insights into how organisational and development practices have evolved over time with a resulting increased "openness". Today, our case constitutes an interesting exemplar of how a large company can utilise a form of Open Innovation to collaborate on the production of software systems.

1.2 Open Source Software as an Exemplar of Open Innovation

Since the late 1960s, researchers and practitioners have struggled with how to cope with an ever increasing complexity in the development of software systems. One way by which companies have sought to address challenges in various projects has been to utilise new development models, including OSS and its associated collaborative model for development (Bonaccorsi and Rossi 2006; Fitzgerald 2006; Lundell et al. 2010).

Open Source is the widely used term for a type of software licence that Richard Stallman referred to as "Free Software" when he founded the Free Software

Foundation (FSF[1]). The term "Open Source" was coined in 1998 to give the phenomenon a more "business-friendly" association. The definition is controlled by the Open Source Initiative (OSI[2]) and the term "Open Source Software" is today more widely used in company contexts than "Free Software". The definition of such software used by the FSF states that it is "a matter of the users' freedom to run, copy, distribute, study, change and improve the software". More specifically, the definition refers to "four kinds of freedom, for the users of the software", namely the freedom to:

- Run the program, for any purpose
- Study how the program works, and adapt it to need
- Redistribute copies
- Improve the program, and release the improvements

Although there are differences in terms of value between OSI and FSF, both organisations refer to the essentially same type of software. Today, both organisations have much in common with collaboration on many practical issues. For example, OSI accepts almost all of the licences defined by FSF, and vice versa. For OSI, the "openness" of source code for any piece of software is primarily a practical issue that allows an open form of collaborative development, whereas for FSF the "freedom" is primarily an ethical issue.

Sometimes, the term "FOSS" is used to stress the similarities rather than the differences between the two. Further, the term "Libre Software" is also used for this type of software, especially in the Latin speaking countries. The term "Libre" avoids the ambiguity of the word "free" in the English language (i.e. "free" as in no-cost vs. "free" as in freedom) and sometimes the term "FLOSS" (Free, Libre and Open Source Software) is used when referring to the collective phenomenon whilst trying to avoid an ideological debate. However, when referring to such software for the purpose of this chapter, we adhere to the term Open Source Software (OSS), which is commonly used in industry.

Irrespective of which term is used for denoting software systems that have been developed using this open form of collaboration, it is essential to recognise that a number of industrial strength software systems have been developed as a result of this form of collaborative development over the years. For example, the operating system kernel Linux (Moon and Sproull 2000), the web server Apache (Mockus et al. 2002) and the web browser Mozilla/Firefox (Mockus et al. 2002) are all being developed, maintained and made available by their respective communities as OSS. Such communities, typically, involve a variety of different stakeholder groups that collectively contribute to OSS projects. It should be noted that there are a range of different motivations, including pure self-interest, that encourage stakeholders to contribute to an OSS project (e.g. Bonaccorsi and Rossi 2006). For many years now, it is clear that a "significant amount of software developed by commercial

[1] http://www.fsf.org/

[2] http://www.opensource.org/

firms is also being released under open source licences" (von Hippel 2005, p. 99). In fact, many companies have experienced that open collaboration in OSS projects, which involve a number of different active users and developers representing a variety of different organisations, can together bring the software to high value and quality.

Today, all the above-mentioned examples of OSS projects are being used in a range of different usage contexts, including mission critical applications in many different organisations. In doing so, it is clear that OSS development shares some fundamental ideas with Open Innovation, such as "greater external sources of information to create value" (Chesbrough 2006). It is therefore, perhaps, not surprising that embedded Linux has been ascribed as a prominent success story of Open Innovation (Henkel 2006).

Collaboration in OSS projects represents a novel way for open collaboration on both ideas and production of software artefacts. As stated by von Hippel (2005), "open source software communities do not allow contributing innovators to use their intellectual property rights to control the use of their code. Instead, contributors use their authors' copyright to assign their code to a common pool to which all—contributors and non-contributors alike—are granted equal access. Despite this regime, innovation seems to be flourishing" (p. 113).

Practical Tip

Collaboration is beneficial for you, especially if you do not break your own added value. Collaboration with the competition may be useful, as you both improve without harming each other. A healthy competition is good for your market, as being the monopolist implies that you have to do any innovation on your own.

In fact, many of the successful OSS projects attract interest and contributions from a range of different individual contributors and commercial organisations. According to von Hippel (2005), "Open source software projects are object lessons that teach us that users can create, produce, diffuse, provide user field support for, update, and use complex products by and for themselves in the context of user innovation communities" (p. 14). Further, it has been noted that there are also similarities between communities related to physical products and OSS communities, in that "complex communities devoted to the development of physical products often look similar to open source software development communities in terms of tools and infrastructure" (von Hippel 2005, p. 103).

It should be noted that adoption of Open Source and Open Innovation principles in company contexts is not always without problems, as experienced by Wallin and von Krogh (2010). They identified tensions in a company between top- and middle-level management concerning its adoption of an Open Source strategy, and report that "While top management embraced an open source policy, middle-level managers who supervised the internal developers were negative toward it. Perhaps the use of external developers undermined their power or prestige, or created concerns about the quality of the products in other ways. Political forces may make it difficult to open up the innovation process to outsiders" (p. 419).

However, a recent study conducted in 13 companies in the secondary software sector, involving senior decision makers with experience of assessing OSS adoption, found that "open innovation practices are already in operation in all of the companies studied" (Morgan and Finnegan 2010, p. 91). Further, the study revealed "the need to increase innovativeness by opening up internal software innovation processes" (p. 91).

1.3 Software Commodification and Its Implications for Software Development

In the past decade, there has been an increasing trend towards changing established software development processes in many organisations. Many companies in the secondary software sector have experienced an increasing amount of complex software systems which are no longer providing a competitive advantage to the company, and it is clear that organisations need to adapt to a new situation which involves an increasing amount of commoditised software systems. With a broader recognition of this commodification trend amongst different stakeholder groups within companies, and with the availability of an increasing amount of complex commodity software, it is clear that organisations need to strategically consider their own development practices in light of their own business goals.

Many companies in the secondary software sector need to deal with how to obtain best leverage from the changing conditions which affects their own established practices. In fact, "only a small part (5–10 %) of the software is differentiating" (van der Linden et al. 2009), and it is this part of the software (which constitutes only parts of a product in this market) that "provides added value over the competitors" (van der Linden et al. 2009). This implies that it is only this small part of the software that helps distinguish a company's own developed product "from competitors' products" (van der Linden et al. 2009). Hence, there is potential for collaboration amongst competitors over large (non-differentiating) parts of the software, and many companies have realised that new development models, including different Open Source development models, may be beneficial for addressing challenges in developing and maintaining complex software systems in many situations in this sector.

When utilising such, network-enabled collaboration in developing the software is jointly developed by stakeholders representing different concerns, both within a single company and beyond its organisational borders. Typically, the software may be produced by a group of designated developers that share a common vision for the Open Source development project. However, for individual companies in

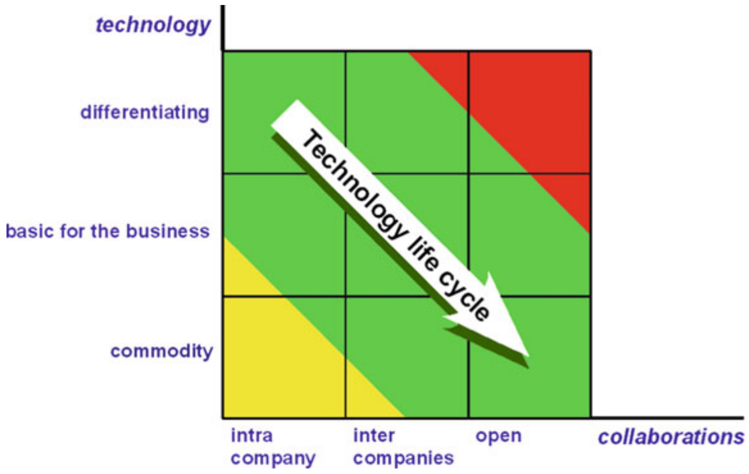

Fig. 1.1 The commodification diagram

this domain, it is important to strategically consider how a company's business and development strategies are congruent and reinforce each other. Obviously, business goals must be in line with development practices, and it is critical for any company to strategically consider what (and when) to initiate and engage in Open Source development related to any specific software systems. In general, the advantages and drawbacks of utilising OSS and its methodology in software intensive systems are not by and large completely understood. However, there are many companies in this sector which engage in Open Source development and "utilise commodity software in order to free resources for innovation" (Lundell et al. 2011).

A significant amount of all software being developed in the secondary software domain does not provide any added value for an individual company over its competitors, as it is more or less common to the product domain. It may even be the case that specific software is a commodity across different domains. Hence, for achieving efficiency and effectiveness in software development, it makes sense for any company to only focus on producing the differentiating parts, i.e. those with the highest added value, in the in-house development. For the remainder, collaboration between different companies and individuals in Open Source development projects is a viable option.

Figure 1.1 shows the commodification diagram for software development (see also van der Linden et al. 2009). It illustrates the landscape of technology versus business decisions on how to develop (or acquire) software. There are two corner areas to be avoided in producing technology. The upper right-hand (red coloured) corner should be avoided, since it would mean exposing (and passing) added value to competitors. For example, a company has reasons to diversify before opening up a unique software system for which they have to spend considerable resources on development in a highly specialised niche market, and as it therefore can be envisaged that the pool of potential contributors from external organisations will be very limited. The lower left-hand (yellow coloured) corner should also be avoided in order to minimise development costs, since commodity technology can be obtained cheaper by adopting (buying) existing technology instead of making it. Hence, healthy software development is characterised by the middle (green coloured) area, from top left to bottom right. Differentiating software, with high added value, is developed within the organisation (top left corner). Commodity software, with low added value, is bought at the market or even available at low (or no) costs (OSS).

Over time, all software is moving from top to bottom in Fig. 1.1. Most (innovative) software development starts out as being differentiating software for some party. At a certain moment, the specific software will not provide a competitive advantage for the company that initiated its development. In such a situation, it can be considered as basic for the business. Further, at a later stage, the software even moves towards commodity. Healthy software development is characterised by combining this vertical movement with the move, for any software, from left to right, from in-house to (open) collaborations. In order to avoid the top right-hand and the lower left-hand corners, companies need to consider strategically when to change their existing development model for any specific software, in order to change the development model at the right pace.

In essence, any company needs to analyse carefully its software with respect to Fig. 1.1 in order to know when to change their development model and approach for collaboration. Further, it is essential to realise that different companies have different business objectives, which affects the interpretation of the status of their software. It is clear that a specific software, which may be a commodity software for company A, may at the same time, for example, be a software which is "basic for the business" for company B. For example, where an integration platform may be seen as commodity for a company offering specific hardware which uses this platform, it may not be for a vendor offering integration platforms as part of its proprietary products. Implications of the commodification of software (i.e. the move from top to bottom in Fig. 1.1) mean that an increasing amount of the software stack in a company is commoditised over time. Hence, it is essential to realise that each company needs to understand these shifts and consider its own software in light of its own business objectives in order to make the right decisions concerning choice of collaboration. In summary, each individual company needs to stay on the "green" in Fig. 1.1 in order for them to stay competitive.

Practical Tip

All organisations must continuously consider technology shifts and assess their own development and adoption of software systems in light of the commodification diagram. In order to obtain leverage from opportunities with open collaboration, an organisation must fully understand technology shifts and how prerequisites for open collaboration evolve over time in different business scenarios and contexts.

In addition to "pure" Open Source software development, some companies also utilise the development model from Open Source. This has been referred to as inner source development, and it involves a set of teams which collaborate in a cooperative ecosystem (Stellman and Greene 2009). Its scope is more restricted compared to "pure" Open Source software development and relates only to the first two vertical columns of Fig. 1.1. Similar to open source development, inner source development applies an open, concurrent, model of collaboration. However, for the rest of this chapter we will focus on Open Source software development.

Currently, a number of companies are utilising open and inner source development to address the commodification of industrial software. In the next section, we draw from a case study in order to illustrate how a large European company has addressed the software shift towards open collaborations, using Open Source software development. In particular, we comment on the evolution through the landscape of Fig. 1.1 and show how the case has moved from a closed to an Open Source software development model which is freely provided on an open platform.

1.4 Open Source Software Development in the Medical Domain

This section gives an example of an endeavour originating from Philips Healthcare[3] in increasing the amount of open innovation of parts of its software, by opening up software in an open source community (Engelfriet 2007). This endeavour was partly based on business reasons—the software was becoming commodity—and partly it was a test case to discover the consequences of starting an open source community. The company should spend most effort on the most business-relevant technology. For the rest, it should cooperate with others. In cases when software becomes a commodity, this means that one should consider opening up software. It will be successful if it attracts enough external collaboration that relieves the company from part of its development costs. It becomes even better when contributors from a large community that are affiliated to other external organisations provide fresh new ideas to innovate the software for the benefit of all involved. Several measures were taken to improve the motivation of participants in the community. The company found that this endeavour of utilising an Open Source community as a strategy for Open Innovation can be profitable, and the conclusion from this experience was that it is profitable. This example is further discussed in van der Linden et al. (2009).

Exchange of medical information has been subject to standardisation and standards since the end of the 1980s, and we have seen an increasing number of standards in this area over the last decades. One such standard is the DICOM standard (Digital Imaging and Communications in Medicine), which is used as a basis for the exchange of medical images. Over the years, different companies have

[3] Formerly called Philips Medical Systems

developed various implementations of the standard. Therefore, conformance to the standard has become important, leading to a need for having a validation tool that can check conformance to the standard. This, in turn, leads to less field problems in interoperability and reduced field support costs.

Since 2000, Philips Healthcare and AGFA Healthcare have been distributing a free binary DICOM validation tool. To make this tool independent of the companies and to improve an open collaboration on the topic, it was decided in 2005 to release the tool as an Open Source software project. The Open Source software DVTk[4] is made available under the GNU Lesser General Public Licence (LGPL v2[5]) and the Open Source project is hosted on the SourceForge.net platform.

The success for the two initiating companies of this transition was based on the community that uses and contributes to the tool. The scope of the open source tool is extended towards the creation of state-of-the-art standard tools to prevent and solve integration problems of systems in the medical imaging domain.

By opening up the software, the initiating parties aimed to create worldwide acceptance of DVTk as an independent (de facto standard) and trustworthy tool, involving a large base of users. By initiating an open collaborative development of the tool, the initiators expected to get the best value out of the development cost. Further, it was also expected that a large user and developer group would result in higher quality and fewer overhead costs. To support these goals, the initiators have remained active in providing community mechanisms that address different motivations for participation in the open collaboration amongst individuals and companies.

A number of initiatives to stimulate collaboration with (and within) the project have been undertaken by the initiators of the project, including:

- Creating DVTk website with forum and registration
- Organising and executing timely User Events
- Implementing the concept of "trainee project"
- Participating in IHE Gazelle Open Source project
- Responding to tenders for Test SW development

The latter two initiatives illustrate the extension of the scope towards other interoperability standards in healthcare. To measure the achievement of the goals, the company has monitored the Open Source project since its start, and in doing so continuously measured a number of aspects which indicate project activity:

- The number of downloads of the tool per year; indicating whether the tool, which is updated regularly, is still useful for a wide community
- The number of comments on the tool received per year; indicating the active interest of the users in improving the tool

[4] http://www.dvtk.org/; http://en.wikipedia.org/wiki/DVTk

[5] http://www.opensource.org/licenses/LGPL-2.1

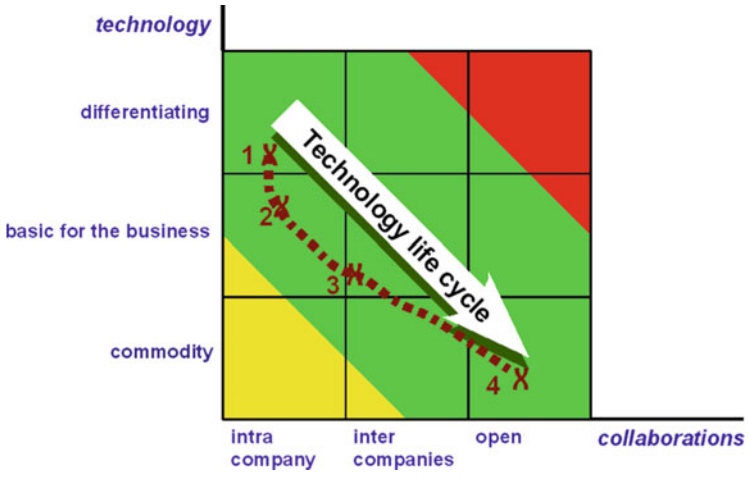

Fig. 1.2 The DVTk project and DICOM in the commodification diagram

- The number of companies participating in the development of the tools; indicating whether there is enough sharing of development

Figure 1.2 shows an overview of four important stages in the history of the DVTk project, clearly illustrating how different company decisions have resulted in an evolution of the project through the commodification diagram (additional details of the project can be found in van der Linden et al. (2009)). Figure 1.2 shows how the project has evolved, via four stages, into a "commodity" project for which there is open collaboration.

Briefly, the four stages in the evolution of the project can be characterised as follows:

1. In 1995, DICOM was a quality interface, only available as a system option for those that really needed interoperability on medical images. Several vendors provided their own solutions, and this was part of the competition. DICOM support was an added value for medical equipment companies.
2. In 1999, DICOM was no longer differentiating. The clients all needed interoperability and they just expected DICOM support and Philips Healthcare decided to provide the DVTk application binary freely downloadable via their own website.
3. In 2001, to share development costs and increase adoption of DVTk, a joint development started with another company (AGFA). The development of DVTk was still developed and provided under a proprietary licence. The functionality is necessary for each company that supports DICOM interoperability.
4. In 2005, it was decided to create an open platform to ensure uniformity. The software is still domain specific, but for the companies involved it is regarded as commodity.

The DVTk project is very ambitious in its objective and strives for global acceptance and more co-developing partners. The project has several different types of participants and users, which can be characterised as: the common user, the interested collaborator and the dedicated developer (O'Reilly 1999). Over time, some users eventually migrate and become more involved, so a common user could become a future dedicated developer. The DVTk project decided to implement some mechanisms which address the motivational factors of developers to promote contributions. These are to:

- Implement User Registration on the website to address the "reputation among peers" motivation. The idea is that when people can have a virtual face when communicating with the community they are more likely to communicate more actively.
- Implement the concept of "trainee project" in the DVTk project to address the "learning" motivation. Co-developing on DVTk is positioned as a way to learn the standard. By having a set of trainee projects, people can select a work item which helps them in understanding the standard.
- Organise and execute timely User Events to motivate the "sense of belonging to the community". If you can meet your co-developers face to face in timely events people tend to feel more committed to the co-developers and thus to the project.

After User Registration was implemented, the number of posts on the discussion forum increased, indicating growing activity and involvement. About ten trainee projects were defined in December 2006. Several of these project assignments led to new developers in the project. In February 2007, the user event attracted 40 participants from 30 companies. After the workshop, five parties were considering participation, of which three eventually became involved. The event resulted in a new collaboration with the IHE organisation which is a leading organisation in the healthcare domain.

The user registration has a positive impact on motivating users to post questions and provide problem reports. It is not a burden for people to register, since it provides the ability for people to gain reputation. For DVTk it seems as it is better to have a smaller group providing a lot of feedback than having a large group of users only posting a small set of comments. The User Event is an effective mechanism to meet potential new parties. Having the training projects is a controlled and effective mechanism to get the potential parties really involved.

The number of downloads has increased enormously since the project was provided as Open Source software. In the first two years, the number of downloads increased from 1,000 to 14,000 per year and the number of comments increased from 5 to 80 per year. The number of users increased by 1,200 % within the same time period. The number of companies and initiatives that work together with the project has grown from 2 to 9. This means that sharing the maintenance and

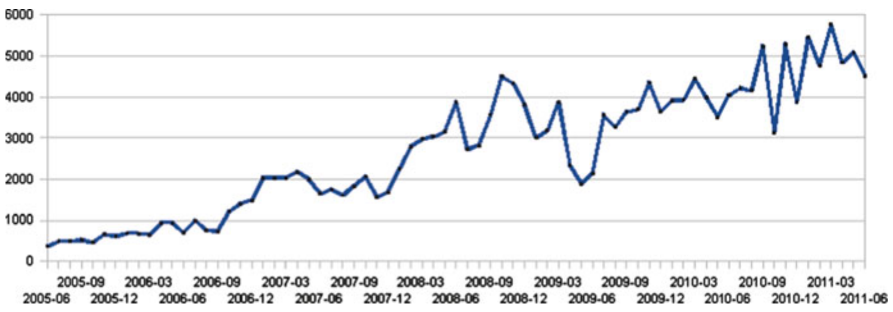

Fig. 1.3 Number of downloads of the software per month

development of the DVTk tool is of value for a large community. It suggests that the project is becoming the de facto standard. More recently, since July 2009, the number of downloads for the software per month has exceeded 3,000 downloads per month (which may be considered significant, given the specific nature of the software). Figure 1.3 shows an overview of the evolution for the number of downloads for the software (until the end of June 2011). From this, it is evident that the overarching trend has been an increased number of downloads of the software each month, ranging from around 500 downloads each month (for the last 6 months during 2005), whereas there have been more than 4,000 downloads each month (for the first 6 months during 2011).

In order to extend the community and the funding base of the project, it successfully applied to a commercial software tool development tender for the IHE-Radiation Oncology Test software and the IHE Gazelle Open Source tooling project. This extends the scope of the community, but stays within the healthcare interoperability domain.

It should be noted that the results reported in this example are not restricted to the medical domain. Instead, the experiences should be useful to the entire secondary software domain. Further, an additional important aspect of this example is that there is a standard that can be used by many companies in the domain. At first the provision of support for the standard gives a significant added value for the provider of an implementation which supports the standard. However, over time, as several competitors will implement the standard, the associated added value stemming from support for the standard becomes reduced as it becomes expected by the clients. Therefore, despite competition on the market, it is wise to open up the supporting software as a strategy for sharing development costs between companies. However, an important prerequisite for success is that the knowledge of the standard is not restricted to one (or a few) companies. In addition, when it becomes successful, it might be attractive for both the company and other external contributors to extend the scope of the open collaboration in the OSS project.

1.5 Discussion and Conclusions

In this chapter, we have presented an illustration of how the development of Open Source Software, as an example of Open Innovation, has been adopted by a large company in the secondary software sector. We have discussed challenges in software development and presented a framework, the commodification diagram, as a means for conceptualisation. We have used this framework as a basis for our presentation of experiences from a case study of Open Source Software development in the medical domain. In doing so, we have elaborated on company decisions and development practices in a software development project that is central for the company and openly provided under the LGPL software licence.

By drawing from the case study conducted in a large international company, we have discussed emerging trends and thereby presented how software development—when conducted as Open Source Software development—constitutes a novel exemplar of how Open Innovation can be conducted. Open Source Software development goes beyond collaborating on ideas, as it also includes collaboration on software artefacts, therefore it can potentially constitute an inspiration for how other areas can adopt an open development beyond the established form of open development as we have seen in the software domain in the form of Open Source Software.

It is envisaged by many that new forms of more open collaborations between different types of communities, as well as large and small companies, will emerge. It remains to be seen which of these collaboration models will be sustainable in the long term.

References

Bonaccorsi, A., & Rossi, C. (2006). Comparing motivations of individual programmers and firms to take part in the open source movement: from community to business knowledge, technology & policy. *Winter, 18*(4), 40–64.

Chesbrough, H. W. (2006). *Open innovation: A new paradigm for understanding Industrial Innovation, Chapter 1 in open innovation: Researching a new paradigm*. Oxford: Oxford University Press.

Engelfriet, A. (2007) Open Source and Open Innovation, Koninklijke Philips Electronics NV, handout: *LinuxWorld Open Summit 2007*, Stockholm, at http://www.idc.com/nordic/downloads/events/linuxworld07/9%20-Arnoud%20Engelfriet.pdf, accessed 7 July 2011.

Fitzgerald, B. (2006a). The Transformation of open source software. *MIS Quarterly, 30*(3), 587–598.

Henkel, J. (2006). Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy, 35*(7), 953–969.

Lundell, B., Lings, B., & Lindqvist, E. (2010). Open source in Swedish companies: where are we? *Information Systems Journal, 20*(6), 519–535.

Lundell, B., Lings, B., & Syberfeldt, A. (2011). Practitioner perceptions of open source software in the embedded systems area. *Journal of Systems and Software, 84*(9), 1540–1549.

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(3), 309–346.

Moon, Y. J. & Sproull, L. (2000) Essence of distributed work: The case of the Linux kernel, *First Monday*, 5(11) http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1479

Morgan, L., & Finnegan, P. (2010). Open innovation in secondary software firms: an exploration of managers' perceptions of open source software. *SIGMIS Database, 41*(1), 76–95.

O'Reilly, T. (1999) Ten Myths about Open Source Software, http://www.oreillynet.com/pub/a/oreilly/opensource/news/myths_1199.html, accessed 7 July 2011.

Stellman, A. & Greene, J., 2009. Inner source 'An interview with Auke Jilderda', Chapter 8 in. O'Reilly, T. (2009) *Beautiful Teams*, 103–111.

von Hippel, E. (2005). *Democratizing Innovation*. Cambridge, MA: MIT Press (April).

van der Linden, F., Lundell, B., & Marttiin, P. (2009). Commodification of industrial software: a case for open source. *IEEE Software, 26*(4), 77–83.

Wallin, M. W., & von Krogh, G. (2010). Organizing for open innovation: focus on the integration of knowledge. *Organisational Dynamics, 39*(2), 145–154.

Further Reading

Dedrick, J. & West, J. (2003) Why firms adopt open source platforms: A Grounded Theory of Innovation and Standards Adoption, Proceedings of MISQ Special Issue Workshop on Standard Making: A Critical Frontier for Information Systems. Minneapolis*: MIS Quarterly*, pp. 236–257.

Fitzgerald, B. (2006b). The transformation of open source software. *MIS Quarterly, 30*(3), 587–598.

Jaaksi, A. (2007) Experiences on Product Development with Open Source Software, in: Feller, J. Fitzgerald, B. Scacchi, W., Sillitti, A. (Eds.), *IFIP International Federation for Information Processing*, Vol. 234, Open Source Development, Adoption and Innovation, Boston: Springer, 85-96.

Jilderda, A. A. (2004) *Inner Source Software Engineering at MIP Fostering a Meritocracy of Peers*, Research report, Philips Research.

Wesselius, J. (2008). The bazaar inside the Cathedral: Business models for internal markets. *IEEE Software, 25*(3), 60–66.