2004

The Use of Embedded Open Source Software in Commercial Products

Hanna Heikinheimo
Nokia Corporation, hanna.heikinheimo@nokia.com

Tuija Kuusisto
Tampere University of Technology, tuija.kuusisto@tut.fi

THE USE OF EMBEDDED OPEN SOURCE SOFTWARE IN COMMERCIAL PRODUCTS

Heikinheimo, Hanna, Nokia Corporation, Multimedia Business Group, P.O.Box 407, FIN-00045 Nokia Group, Finland, hanna.heikinheimo@nokia.com

Kuusisto, Tuija, Tampere University of Technology, Korkeakoulunkatu 8, FIN-33820 Tampere, Finland, tuija.kuusisto@tut.fi

Abstract

The purpose of this paper is to increase understanding about the use of embedded open source software in commercial telecommunication products. This paper applies concepts of value-creating systems to sourcing and use of open source software. In value creating systems, open source software communities do not operate in similar way as companies. Differences derive from organizational culture, values and motivation to create value, e.g. to develop the software. Companies are monetarily motivated to create value for their customers but open source software communities are mainly voluntarily creating value to themselves and their peers. These differences have to be understood and taken into account when companies are using open source software in commercial products. In this paper, a model proposal describing collaboration with the open source community when using open source software in commercial products is outlined. Key findings about applying the model in a case company are presented.

Keywords: IS development, software sourcing

1 INTRODUCTION

The purpose of this paper is to increase understanding about the use of open source software in commercial products. This understanding is needed for developing software product processes covering the interface between open source software and commercial product development. There are various definitions for the term open source software. One rather comprehensible one is the definition given by Hansen et al. (2002). They define that for open source software it applies that: (1) the source code is distributed together with the executable form of the program, (2) the software is free to use and (3) the software has license which allows anyone to modify and redistribute it. In general, the definition of the term, open source software, is two-dimensional: the technical dimension is the open source code and the legal dimension is the mechanism of licensing terms that make the source code available for the users and allows modifying and redistributing it.

Open source software is developed in open source communities, which consists of a large number of mostly small-scale development projects. Companies can utilize open source software in various ways. In order to define the use of open source software in companies, a division to three types of use is presented. Firstly, open source software can be used internally as a part of company's information system infrastructure or in the case of a software company as development tools. Secondly, open source software can be used as a part of company's products. In this case, company distributes the open source software further inside its product to the customers. In this type of use, the risk and licensing management is emphasised due to visibility and possible impacts on the customer relationships. Thirdly, open source software can be used as a way to promote certain technologies to become de facto standards. This is based on the assumption that the open source development will interest numerous parties, which will use the software and take part in developing it. Therefore, it is supposed that the technology will eventually develop superior compared to the competing more closed technologies developed by only one or few companies. The third type of open source software use includes companies turning their current proprietary software into open source software and taking an active role in developing the software in open source communities.

This paper focuses on the use of open source software in company's products, i.e., on the second type of open source software use. Especially, the paper emphasizes the interface between open source software communities and commercial companies. The interfacing communities and companies are regarded as a value-creating system. The models of analysing value-creating systems date back to the value chain and value system models of Porter (1998, originally 1985). Some authors (see, e.g., Cartwright & Oliver 2000) have suggested that new models are needed because the value chain model is difficult to apply in the service- or information-goods-producing companies like software-intensive companies. Recently, value-creation systems have been examined in a complex operating environment that is largely influenced by the revolution of the communication technologies (Parolini 1999), (Cartwright & Oliver 2000), (Normann & Ramirez 1993), (Evans & Wurster 1999). Authors seem to emphasize two phenomena: (1) the role of information is becoming increasingly important in value-creating systems and (2) the focus of competition is shifting from the companies competing inside an industry to the competition between value-creating systems, networks, which go across the borders of traditional industries. In the long run, an individual company will win only if it belongs to value-creating system that is winning (Parolini 1999).

A value-creating system consists of activities that are creating, adding or consuming value. Sets of human-, tangible- and intangible resources are used for carrying out these activities. Activities are connected to each other by flows of information and goods. (Parolini 1999) This paper refers to the key concepts of value-creating systems and applies them to model the interface between open source communities and commercial companies. In addition, this interface is further investigated by a case study. The case study was performed in a telecommunication company using embedded open source software in their products.

2 OPEN SOURCE SOFTWARE COMMUNITY INTERFACE TO A COMMERCIAL COMPANY

Open source software communities interface to commercial companies differ from a typical interface between two companies. These differences derive from organizational culture, values and motivation and approaches to develop the software. Culture is most commonly defined as a set of shared values, shared understanding or even shared methods of problem solving (e.g. Hall 1971). Straub et al. (2002) argue that information systems (IS) research nearly always assumes that an individual belongs to a single culture. They proposed social identity theory to be used as a grounding for cultural research in IS. Social identity theory suggests that each individual is influenced by plethora of cultures. (Straub et al. 2002) When applied to open source software communities, this means that approaches trying to classify open source communities under a homogeneous culture are not enough for understanding the influencing factors behind the communities. Each individual working for an open source community is influenced by several national and work cultures. These cultures have an effect on the way the individuals interpret the aims and motivations of their communities as well as traditional companies' efforts to collaborate with the communities.

When collaborating with open source communities, a company typically faces unfamiliar motives to create value, i.e., develop software. A developer's motivation to participate in open source development can derive e.g. from the fame that showing ones skills can bring (Lerner & Tirole 2001), from the status in the community (Raymond 1998) or from strong interest toward solving challenging programming problems (Himanen 2001). In addition, open source software development approaches are almost as numerous as the projects. Fink (2003) describes that a development project can be managed e.g. by: a company, foundation, committee or an individual. Noteworthy is that the development is not chaotic. Although open source code gives anyone a possibility to modify the source code only few developers have the rights to directly contribute to the development projects (Hansen et al. 2002). Read access to code and to the mailing lists is usually available for anybody but the writing access to code repository requires that the programmer has proven his skills. (Hansen et al. 2002) Usually, a successful development project has also a leader or a leader team, which chooses the most appropriate solutions and informs the developers which problems are most in need to be solved (Lerner & Tirole 2001).

Considering internal hierarchy of communities, it is obvious that only respected developers are able to influence the project. Project leaders might not be willing to take orders from organizations or people, who have not proven to be competent by working with the communities. Therefore, it is not recommendable for companies to make demands to them. Communities are not financially depending on the companies and therefore they should not be considered to be suppliers that are trying to sell their software to the companies.

Software development in open source communities is not directly connected to companies' product strategies. The reason for this is that open source communities' and companies' end customers, and therefore also their aims, may be different. Open source communities are not primarily aiming to create value for the end-customers or for the owners of the company. Value creation in the open source communities is performed, e.g., for the users of the open source software, for the developers or for a common good. However, communities may also benefit from creating value for the end-customer of a company, e.g., if the penetration of open source software in the market is increased because of it.

So, open source communities do not operate in the same way in value-creating systems as companies. Companies, which are using open source software, have to adapt to that. Open source communities usually create value to companies by making the software, documenting the software and continuously developing the software. Traditionally, the term value has been considered to indicate customer's willingness to pay for a product or service, see e.g., (Porter 1998). In the case of open source software, value typically cannot be directly defined by looking at a company's willingness to pay for it. A broader definition for value is needed. This need is widely recognised, e.g., among the field of intellectual capital management (e.g., Roos et al. 1997) where value is not defined only in financial terms, but also in a wider context including non-monetary assets like human capital, social capital and organizational capital (Pike et al. 2002). Open source community is a pool of technical competence and knowledge, i.e., it is a source of human capital. Social capital resides in the relationships, which are created through exchange between open source communities and companies.

Added value has often been defined as a usefulness or goodness that a company is able to add to a product. Open source software communities are creating value although they are not selling their software to any customer. Value that they create is the usefulness and the good quality of the software that they develop. Open source software communities also create human capital that is one form of value provided by the communities. Companies that are integrating open source software into commercial products are utilizing this value created in the communities. Communities' value creating activities cover only part of the activities needed in the creation and delivery of a commercial product. When open source software is included in commercial products, the additional activities needed for making commercial products are conducted by the companies in the value-creating system.

3 VALUE-SYSTEM OF USING OPEN SOURCE SOFTWARE IN PRODUCTS

This chapter contains a model proposal describing collaboration with the open source community when using open source software in commercial products. Building the model starts by defining actors, activities and resource flows in the value-creating system. Each actor is examined to find out activities that it conducts to create value. Resource flows are mapped together with the activities. The actors are limited in this model proposal to be: the company using open source software in its products, open source software community and company's end-customer. In practise, there are several companies in one value-creating system. For example, there might exist companies that are focused on offering open source software support services for product integrator companies.

First actor observed is the open source software community. Open source software communities consist of two kinds of software developers. The most dedicated ones coordinate development projects and contribute to them significantly. They perform value-creating activity called "software development" in the Figure 1. The other group is the first-line open source software users that do not actively take part in the development, but create value by enhancing the software. They suggest fixes or report problems that they face while using the software. Even though their personal contribution would not be large, their number makes their value-creating activity important. In addition to software development and using activities, there is a communication activity between the developers and the first-line users. Communication happens on mailing lists that also companies can use as a source of technical information. In this way, also the intellectual capital form of value that community creates is accessible for companies. Role of open source communities in the value-creating system is described in the upper part of Figure 1.

Value creation activities inside the company are structured according to the main activities of the virtual value chain (Tapscott 1999) and the main parts of the product offering. The offering is assumed to consist of the actual product, its maintenance and support service. Parts of the offering are described horizontally in Figure 1. Virtual value chain is a value-creation system in which information has a value creating and not only a supportive role (Tapscott 1999). Because software is information in codified form it is produced in a virtual value chain. Tapscott (1999) lists four main value-creating activities in the virtual value chain:

- Gathering information
- Selecting information
- Synthesizing information
- Distributing information

When these main activities are applied to producing each part of the offering, a more detailed description of the activities is found. In product creation, information gathering means searching software components that are available on the market, including also open source software. In Figure 1 this is marked as an information flow from the open source communities to the company. In the information selection phase, company has to make decision about the product architecture and components that will be integrated into the product. The company performs evaluations about the technical, legal and business suitability of the software components, e.g., (Helokunnas & Nyby 2002). Based on evaluations the components to be used are selected and value, the software, is transferred from communities to company. In the synthesis phase, components are modified if needed, integrated and the product is tested. These activities are the company's part in value creation for customer. Modifications to open source components can be distributed back to the community as part of the information distribution activity, which indicates that companies are also creating value that is beneficial to communities. As the arrows in the Figure 1 illustrate, resources and value-added are flowing from the community through the company to the customer but also back to the communities.

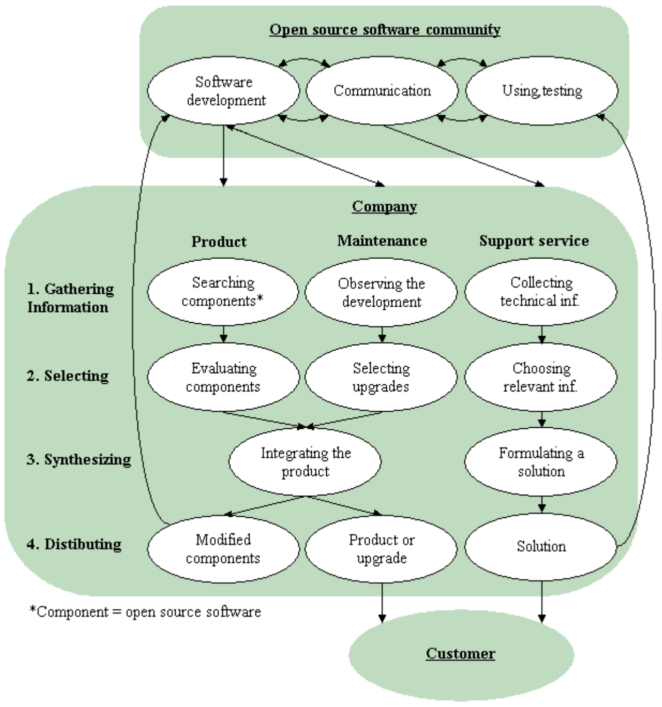

Figure 1. Value creating system of using open source software in products.

When offering product maintenance to customer, information about the further development of external software components used in the product is gathered. Components that might be more suitable and substitute the current ones are observed as well. Company follows how open source software components are developed in the communities. Sometimes the company can take directly part in the development. Therefore, the resource flow between the company and the communities is marked with two-way arrow in Figure 1. In the selection activity of the maintenance, company has to decide which upgrades and modifications to the product are feasible to make from business, technical and legal perspectives. Open source software communities develop their components continuously and new releases are made often. However, it may not be feasible to integrate all the new releases of open source software components to company's product. In the synthesis phase, selected upgrades are integrated to the next version of the product and again value is transferred from communities to the company. In the distribution phase, new version of the product is delivered to the customer and modified open source components to the community.

For being able to offer technical support for the customer the company has to gather technical information. In the case of open source software technical information can be gathered from the documentation offered by the community or from its mailing lists (see arrows in the Figure 1). In the selection phase, company has to select which information is relevant and in the synthesis phase technical information needs to be categorised and organised so that solutions for the customer can be drawn from it. In the distribution phase, the solution for customer's problem can be delivered. There is also a resource flow from the company back to the community marked in Figure 1. This means that if mistakes or possibilities for improvement in the open source software are found based on the customer feedback; they can be reported to open source communities. In this way customers of the company are creating value for open source software communities.

It has now been discussed how companies can utilize value, software and technical expertise, that open source software communities create. However, utilizing the values created in open source communities does not come without costs. Finding useful information and suitable open source software components causes search and evaluation costs. Since, most of the open source software is not advertised it takes competence to find suitable open source components. Acquiring and maintaining such competence by following the open source market does not come for free either. As mentioned earlier to influence open source community, one has to be respectful member of it. Therefore, if companies aim to influence the open source software development they have to be also wiling to bear costs that are born when taking actively part on the open source development in communities. Adjusting company's internal processes to open source use causes also some indirect costs that are born partially because of complicated licensing scheme of the open source software.

4 CASE STUDY

A case study was conducted in a global company operating in the telecommunications field. The company uses open source software in its products. The server platform products consist of hardware devices and most often embedded software. They provide a carrier-grade foundation for All-IP mobility systems. All-IP complements GPRS and EDGE/WCDMA and it supports add-on capabilities and capacity for increasing traffic. The all-IP systems combine multiple media streams for rich call, messaging, and browsing services. The target of the systems is to provide good quality of service, scalability, and minimized costs. The systems are based on industry-wide standardization of interfaces through forums like 3GPP, WAP Forum, Wireless Village, and IPv6 Forum. The development of the server platform products includes integration of embedded open source software components. For example, one part of the platforms is a carrier-grade server platform using the Linux® operating system. The server will be the foundation for core network products with functions such as session control and registers. In radio access, the product is used for managing the signalling plane of mobility control functions, including common radio resource management.

Eleven employees were interviewed individually in semi-structured interviews each lasting about two hours. Majority of the interviewees were working with the server platform development but some were also working with open source software in support functions. Interviewees were working as managers or specialists in legal and technical fields. Main reasons to use open source software in the case company were inexpensive licenses and reduced resource needs in software development. In addition, the productivity of development was considered to be increasing due to open interfaces and open source code. The company argued that by using open source software the company was able to become more independent of suppliers. In some cases, open source software was considered technically more advanced than other options available. On the other hand, also barriers for use were found in the interviews. One often mentioned risk was, e.g., technically immature open source software. This is related to the searching difficulties discussed earlier. Risk management seemed to have very important role in the open source software component selection phase of the product development.

All the proposed open source software to be used in the case company’s products goes through a careful evaluation process. Technical suitability e.g. ease of integration and costs of maintenance were estimated. From legal point of view the license and intellectual proprietary rights were checked. Also the developer community was evaluated. The decision whether to use open source software or not, was always performed case by case. Still, some general decision-making principles were found in the interviews. The decision-making was influenced e.g. by the customer, and the type of software that was needed. Customers or internal developer users might resist use of open source software because it has traditionally had quite low usability, i.e., user-friendliness or ease of use. Furthermore, changing the technology always causes indirect costs e.g. learning costs that should be taken into account in decision-making. Customer resistance might occur if it was suggested that a commonly used de facto component would be substituted with an open source software component. Of course the attributes of the open source software components available were also playing an important role in evaluations. Only component that fulfilled certain conditions could be taken into use. They had to be technically fitting for the purpose, the license had to be suitable and they had to be compatible with Commercial-Off-The-Self (COTS) and in-house developed software. In addition, an open source component should not be too small. The smaller and simpler a component, the more profitable it is to make the component in-house since search and evaluation costs of using open source software would be relatively heavy. This is in line with Szyperski (1997), who felt that too small components with a variety to choose from, e.g. hundreds of different implementations of stacks and queues, are not likely to increase software development productivity.

5 CONCLUSIONS

A model proposal describing collaboration with the open source communities when using open source software in commercial products was outlined. Key findings about applying the model in a case company were presented. According to the findings in the case company, the open source software evaluation activities in the value-creating system are to be emphasized. It can be concluded that the evaluations are very important for successful use of open source software in commercial products and that they also cause significant part of the costs of use. In addition, the synthesis activity was mentioned in the case company to cause integration costs. But it seemed that these costs could be already largely influenced by the quality of the evaluation activity. Therefore, it can be argued that if the evaluation activity fails to create value, i.e., accurate evaluations, the value consumption occurs in the synthesis phase of the model. The distribution activity in the model was hardly mentioned in the case study. This indicates that it is not very complicated activity from the software development point of view that the interviewees of the case company represented. However, when looking at the situation from the point of view of the whole value-system, it can be argued that distribution can create value significantly if it fuels open source software development in the community by distributing modified software back to it.

The presented findings from the case company are based on a small sample of empirical data. The interviewed people in the case company are well experienced in the use of embedded software in system products and they have a large number of deep contacts in open source communities. So, it can be assumed that the results are valid for other companies using embedded software in their products as well. The research on the interface between open source communities and commercial companies will continue as a process development for requirement engineering of commercial products containing open source software.

References

Cartwright, S. D. and Oliver, R. W. (2000). “Untangling the value Web.” *The Journal of Business Strategy*. Vol. 21(1), pp. 22-27, 2000

Evans, P. B. and Wurster, T. S. (1999). Blown to bits: How the new economics of information transforms strategy. Boston, Massachusetts, Harvard Business School Press. 261 p.

Fink, M. (2003). The Business and Economics of Linux and Open Source. New Jersey, Prentice Hall. 242 p

Hall, E. T. (1971). *Beyond Culture*, Doubleday & Company, Inc.

Hansen, M., Köhntopp, K. & Pfitzmann, A. (2002). "The open source approach – opportunities and limitations with respect to security and privacy." *Computer & Security*. Vol. 21(5), pp. 461-471, 2002

Himanen, P. (2001). Hakkerietiikka ja informaatioajan henki. Helsinki, WSOY. 205 p.

Helokunnas, T.,Nyby, M. (2002). Collaboration between a COTS Integrator and Vendors. In Kontio, J. & Conradi, R. (eds). *Software Quality* - *European Conference on Software Quality 2002*. Helsinki, Finland

Lerner, J. and Tirole, J. (2001). "The open source movement: Key research questions." *European Economic Review.* Vol. 45(4-6), pp. 819-826, 2001

Normann, R. and Ramirez, R. (1993). "From value chain to value constellation: Designing interactive strategy." *Harvard Business Review*. Vol. 71(4), pp. 65-77, 1993.

Parolini, C. (1999). The Value Net –A Tool for Competitive Strategy. West Sussex, John Wiley & Sons Ltd. 239 p.

Pike, S., Rylander, A., Roos, G. (2002). "Intellectual Capital Management and Disclosure", Choo, C.W. & Bontis, N. (ed.). *The Strategic Management of Intellectual Capital and Organizational knowledge*, Oxford, Oxford University Press. Pp. 657-671.

Roos, J., Roos, G., Dragonetti, N.C. and Edvinsson, L. (1997) *Intellectual Capital. Navigating the new business landscape*, MacMillan Business, Hampshire.

Porter M. E. (1998). Competitive Advantage – Creating and Sustaining Superior Performance. With new introduction, (originally published in 1985). New York, The Free Press. 557 p.

Raymond, E.S. (1998). The Cathedral and Bazaar. Version 1.33. First Monday. 24 p.Url: http://www.firstmonday.org/issues/issue3_3/raymond/ (Last modified 16.2.1998)

Szyperski, C. (1997). Component Software: Beyond Object-Oriented Programming. ACM Press & Addison-Wesley.

Straub, D., Loch, K., Evaristo, R., Karahanna, E., Strite, M., (2002). Toward a Theory-Based Measurement of Culture. *Journal of Global Information Management*. Jan-March 2002, 10(1): 13-23

Tapscott, D. (ed.). (1999). Creating Value in the Network Economy. Boston Massachusetts, Harvard Business School Press. 229 p.