A Model of Commercial Open Source Software Product Features

Florian Weikert and Dirk Riehle

Friedrich-Alexander-Universität Erlangen-Nürnberg
florian@weikert.it, dirk@riehle.org

Abstract. Commercial open source software has become an important part of the packaged software product industry. This paper provides a model of individual product features, rather than full-fledged business models, and their perceived value to customers. The model is the result of a three-iteration study, including interview analysis, literature review and the implementation of an empirical survey. Companies can use the feature model to determine their products and business model..

1 Introduction

Open source software (OSS) – software whose source code is publicly available and which allows modification and redistribution at no costs – represents a new approach in a world where software used to be kept proprietary. Nowadays, OSS is a major player in important areas such as web browsing (Firefox, Chromium), databases (MySQL), operating systems (Linux) and mobile (Android). Especially the success of Android, which runs on 75% of all smartphones shipped in the third quarter of 2012 [1] indicates that the trend of open source has not yet come to an end.

The $1 billion acquisition of open source company MySQL AB by Sun Microsystems (now part of Oracle Corporation) in 2008 is just one example which shows the commercial potential of this trend [2]. Today, major tech companies such as Google[1], Facebook[2] and Apple[3] use and contribute to open source projects, thus further highlighting the economic significance of OSS.

However, there are still companies that follow the traditional closed-source approach by keeping their software proprietary. Their refusal to reveal the source code of their products to the public might be due to the potential risks of "going open source", e.g. the loss of intellectual property or the increased attack surface for possible lawsuits [3]. Additionally, they might not see the commercial potential of OSS which would compensate them for taking these additional risks. These companies are likely to ask themselves: "How can you make money if you give the

[1] http://code.google.com/intl/de/opensource/projects.html
[2] http://developers.facebook.com/opensource/
[3] http://www.opensource.apple.com/

© Springer-Verlag Berlin Heidelberg 2013

software away for free?“[4]. Or, even more worse: “why should a firm further develop a product if competitors can freely appropriate these contributions?” [5].

As outlined in Section 7, prior research addressed this question on the level of business models. However, companies might still be confused which concrete product features can differentiate a free open source product from an commercial offering: “Which features are our customers willing to spend money on?”. We address this question in the present paper by generating a model of commercially viable product features of OSS. The model itself is the result of a three-step process including interview analysis, literature review and the implementation of an empirical survey. The contributions of this paper are:

- A hierarchical model of commercially viable product features of OSS products with detailed explanations on how these features can be used and what their characteristics are;
- A survey-based examination of individual product features based on their frequency and perceived importance.

The paper is structured in the following way: Section 2 sets the scope of this paper and provides definitions which are being used throughout the entire paper. Section 3 outlines the research process while Section 4 displays the final model of product features. Section 5 contains findings from the survey concerning the ranking of individual product features. An evaluation of both the model and the quality of the results can be found in Section 6. Related work is reviewed in Section 7. Finally, Section 8 concludes this paper.

2 Scope and Definitions

Open source software is defined as software covered by an *open source license* – a license which is approved by the Open Source Initiative (OSI) and which complies with the “Open Source Definition”[4]. Consequently, source code must be publicly available and both free redistribution and derivative works have to be allowed.

Open source companies are companies generating revenue based on at least one open source software product. Compared to closed-source companies, this puts them in a special situation since their commercial offerings have to compete with an open source product which is available for free. Consequently, these companies have to provide substantial additional value in order to motivate users to pay for their commercial offerings.

This paper considers three different types of open source companies, namely *software producers (vendors)*, *service providers* and *distributors*. Software producers create and sell OSS products, thus capitalizing on their own intellectual property. In contrast, service providers typically offer services such as training or consulting for third party OSS products which are not their intellectual property. Distributors integrate a set of third party OSS products into a configured, ready-to-use product.

[4] `http://opensource.org/osd`

It is important to note that this categorization is based on the dominant aspect of a company's business model. For example, software producers may also offer services for their own products without being considered a service provider. Hybrid business models are beyond the scope of this paper. Additionally, companies offering OSS products in order to sell complementary hardware or to generate revenue through advertisement are also not part of our research.

3 Research Process

The model presented in Section 4 is the result of a three-step iterative process with each iteration resulting in a different stage of the model. While the first two iterations were limited to qualitative research, we employed a quantitative approach in the final iteration in order to evaluate and enhance the model.

3.1 First Iteration: Initial Model

The initial model is based on a set of seven confidential interviews between Dirk Riehle, Anthony I. Wasserman and five employees of open source companies who have been working in the software industry for over 18 years each. They represented a total of three open source companies which have been existing for at least six years at the time of the interview. Furthermore, all three companies are based in the United States and can be categorized as software producers.

We used an open coding approach in order to extract relevant terms from the interview transcriptions. Related codes were grouped together in order to form categories such as *training* or *support*. We used codes within a group in order to find possible dimensions, e.g. *response time* for the category *support* [6].

3.2 Second Iteration: Revised Model

We revised the initial model in three consecutive steps. In each step, we analyzed literature and the product portfolios of open source companies in order to add new features to the model [7]. Literature review also helped us to describe individual features in more detail than we did in the initial model. Additionally, we introduced new categories in order to replace the flat structure with a multi-level hierarchy.

3.3 Third Iteration: Quantitative Evaluation and Enhancements

The last iteration aimed at evaluating and enhancing the revised model. We accomplished this by creating an online survey aimed at employees of open source companies. The questions of the survey can be grouped into four categories:

- First of all, participants were asked to categorize their company based on the definitions from Section 2. This allowed us to analyze answers based on specific business models.

- The second category contained product-feature-matrices where participants had to mark which of the product features was offered in which of their products. Based on these matrices, we were able to deduce how frequently certain features were offered, how features were bundled together and which features were not relevant in practice.
- In the third category we asked participants to rank product features based on their subjective importance. There was one ranking for each feature category such as "support" or "training". We used these rankings in order to compare the subjective importance of each feature with its frequency.
- Finally, we asked for missing features in order to enhance our model.

We used statistical methods to analyze the results of the survey. However, since only 15 valid responses were received, the implications derived from this survey can only be regarded as starting points for future research rather than definitive results.

4 Model of Product Features

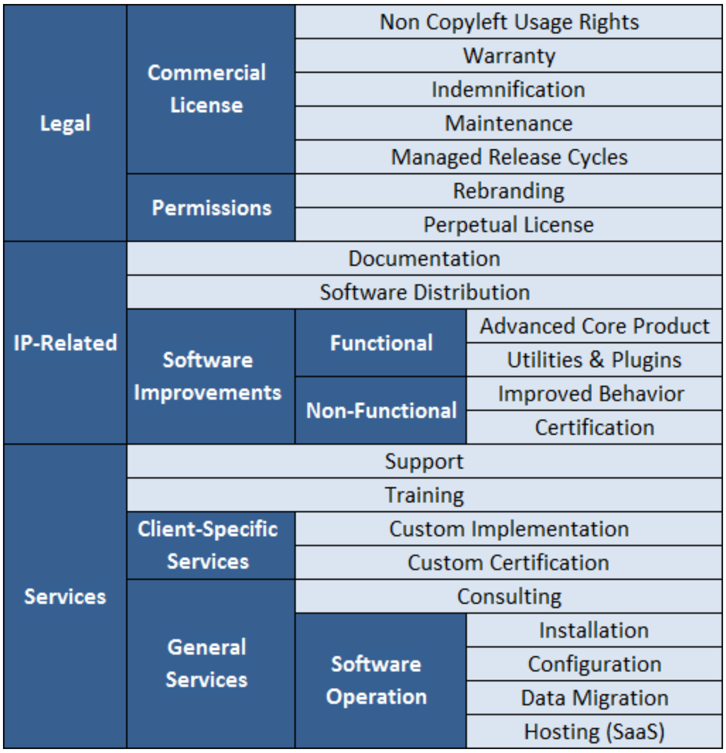
The top level of the revised model consists of three major feature categories: legal features, features related to intellectual property and service features. Each of these categories is discussed in detail in this section. Additionally, we will look at every single feature and explain how it is defined, why customers are willing to spend money on it and how its dimensions look like.

4.1 Legal Features

Legal features can be divided into two sub categories: *commercial license* and *permissions*. *Commercial licenses* are important when looking at the so-called "dual licensing" approach. If a company owns the intellectual property rights of a software product, then it can offer its products under multiple licenses. For example, a company can provide its product for free under the GNU GPL license in order to satisfy the condition of being an open source company. Additionally, it can sell the very same product covered by proprietary license, allowing customers to side-step the requirements enforced by an open source license. We identified five major features which might be a reason for such behavior:

Non copyleft Usage Rights. A special type of open source licenses – called reciprocal or copyleft licenses – contains a special requirement: If an OSS product A is covered by a reciprocal license and someone integrates it into his own software B and distributes the resulting product, B has to be licensed under the very same reciprocal license ("viral effect"). Consequently, this implies that B has to be made open source, as well. If someone is not willing to take this step, they might be interested in spending money on a commercial license including non copyleft usage rights.

Warranty. Open source software usually comes without any warranty. Business users, however, might be interested in having a warranty clause in order to mitigate potential damages. Consequently, warranty can be offered to customers as part of a commercial license. This feature has several dimensions: first of all, it is important to define what exactly is the subject of warranty. Furthermore, warranty period, actions in case of remedy, type of covered damages and limitations such as customer negligence have to be considered as well. These dimensions allow companies to tailor their warranty clause as needed or even to offer several warranty clauses with gradual pricing.



Legal	Commercial License	Non Copyleft Usage Rights	
		Warranty	
		Indemnification	
		Maintenance	
		Managed Release Cycles	
	Permissions	Rebranding	
		Perpetual License	
IP-Related	Documentation		
	Software Distribution		
	Software Improvements	Functional	Advanced Core Product
			Utilities & Plugins
		Non-Functional	Improved Behavior
			Certification
Services	Support		
	Training		
	Client-Specific Services	Custom Implementation	
		Custom Certification	
	General Services	Consulting	
		Software Operation	Installation
			Configuration
			Data Migration
			Hosting (SaaS)

Fig. 1. Hierarchy of product features. From left to right: main categories, sub-categories (both with dark background) and features (light background).

Indemnification. If a customer of an open source company distributes OSS as part of his own products, he can be held responsible for damages caused by the OSS product. An indemnification clause would allow him to transfer this responsibility, thus moving the risk to the original creator of the open source software. Similar to warranty, indemnification means that customers can mitigate potential damages. Consequently, they might be willing to spend money on this feature.

Maintenance. Commercial users typically want access to fast and defined problem solutions for the employed software. Open source users may have to wait for a long time until a bug gets resolved, so commercial users might be willing to pay for a maintenance contract that provides the bug fixes faster and in a way that matches their deployment..

Managed Release Cycles. In contrast to traditional closed-source software, new versions of OSS are released very frequently – sometimes even several new versions per month [3]. While such rapid releases are important in order to ship bug fixes as soon as possible, they also force customers to deal with updates frequently. This is not only time consuming – even more so, it might disrupt customers when significant changes occur often. Especially companies might be interested in receiving fewer, but more stable updates, as one of the interview partners indicated: “Once we’re running a production system, you really don’t want to have to upgrade and modify it too many times there”. Consequently, customers might be charged for such a guarantee.

The second sub category, *permissions*, contains the following two features:

Rebranding. Some software vendors like Openbravo require that their trademarks must not be removed from their open source products [8]. Consequently, even derivative work must display these trademarks. Due to customer perception, however, other companies might like to distribute such derivative work exclusively under their own trademark. This means that they are likely to spend money if they can rebrand the parts of the software belonging to an open source company.

Perpetual License. All interview partners mentioned that their companies use subscription-based payment models instead of charging upfront license fees. As a result, customers can use their OSS products only as long as they pay. However, if they redistribute the OSS product as part of their own product, their customers suddenly depend on the contract between these first-level customers and the open source firm: if those cancelled their subscription, their customers would no longer be allowed to use the derivative software, too. Consequently, they would have to renew the subscription until the last of their customers stops using the derivative software. An open source company can address this problem by selling these customers the permission to grant perpetual licenses to their customers.

It is important to note that several of the above features are completely irrelevant to end-users. In fact, features such as *non copyleft usage rights* and *rebranding* target resellers and OEMs exclusively since they do not provide any value to end-users. Consequently, this distinction has to be considered when defining the legal features of a particular commercial open source offering.

4.2 Features Related to Intellectual Property

This category contains features which are either software or documents to which usage rights are being sold. The top level of this category consists of a sub category and two features.

Documentation. Documents such as reference manuals and user guides are necessary in order to operate and maintain complex software products. Consequently, open source companies can sell (advanced) documentation to the users of their software products.

Software Distribution. This term describes a configured, ready-to-use software product which is the combination of several different OSS products. One prime example is the Linux operating system where several distributions exist, e.g. Ubuntu. It is important to notice that the resulting configuration is the intellectual property of the distributor while the individual components may be owned by third parties [9].

In addition to these two features, the top level also contains the sub category *software improvements*. Typically, open source companies can offer a commercial software product which is based on their OSS product while being somewhat superior. This superiority can be achieved in the following ways:

On the one hand, commercial products can have functional differences by providing functionality which is not implemented in the OSS product. Consequently, the commercial product can carry out additional tasks. Functional differences can be realized in the form of the following two features:

Advanced Core Product. This approach implies that the source code of the open source version is a subset of the commercial version's codebase. Consequently, the functional differences are implemented in the parts of the source code which are kept proprietary.

Utilities and Plugins. Functional differences are realized in the form of proprietary utility applications or plugins which can be used in conjunction with the open source product.

On the other hand, *non-functional differences* are another possibility to differentiate commercial products from their open source counterparts. This sub category contains the following features:

Improved Behavior. This means that the commercial product offers the same set of functions as the open source version does. However, these functions are executed in a superior way. Such qualitative differences can be realized by improving scalability, performance, security, safety, availability, reliability and user experience.

Certification. Commercial versions of an OSS product can be certified for the use with other software or hardware. Furthermore, certification against processes is also possible, i.e. products can be employed in a specific process or their development process meets a certified standard. Customers may be willing to spend money on certification since it guarantees that the product can be operated in the desired way. Moreover, certification may even be a legal requirement in some jurisdictions or where software is employed in highly critical environments.

4.3 Service Features

This category contains the features *support* and *training*, as well as the sub categories *client-specific services* and *general services*:

Support. Similar to maintenance, support is another commercial-only feature ensuring the smooth operation of the software. Although non-paying users can usually receive help through public forums and mailing lists, these options are neither reliable nor do they guarantee a certain response time. Consequently, business users are likely to spend money on a support contract. Several dimensions of support can be used to design this contract accordingly. First of all, support type and channel have to be defined. One possible option is called "managed support" which implies that customers can interact with employees over phone, email, online chat or official forums. On the contrary, "unguided support" means that customers get access to a set of resources such as knowledge bases and FAQs in order to solve their problems by themselves. Next, customers expect statements regarding the quality of support. Two metrics are of importance: response time and availability. Additionally, open source companies might offer a dedicated support representative to take care of their best customers.

Finally, quantitative characteristics have to be defined. For example, open source companies may limit the number of support incidents if they provide subscription-based support. Additionally, access to the support team can be restricted to a certain number of employees at the customers' companies.

As a result, these different dimensions allow open source companies to create multiple support offerings, each of them addressing the needs of a different customer segment. Consequently, they can sell managed support with 24/7 availability, low response time and unlimited incidents to business users while offering 12/5 support with longer response time and a limited number of incidents to private users.

Training. If the OSS product is sufficiently complex, users might be interested in getting trained on how to use it efficiently. Consequently, open source companies can sell training as a commercial feature. It can be provided online or in real-world class rooms. Online training can either be self-study – by providing documents and online resources - or instructor-led. Additionally, professional certification training can be offered as well. This enables employees to prove that they gained specific knowledge.

The sub category *client-specific services* contains service features which can be requested "on-demand" and whose execution details are particularly tailored to the specific needs of an individual customer.

Custom Implementation. Some customers might have very specific requirements which are not met by the standard software. Consequently, companies can offer to change the implementation of the software or to write additional software components upon the client's request. Since the result of such bespoke services will be superior compared to a general purpose software, customers might be willing to pay for it.

Custom Certification. Due to the great number of available software and hardware, companies are likely to certify their software only against a limited selection of products. If customers want the OSS product to be certified against a very specific product, the open source company might create an slightly changed product which has the requested certification.

The last sub category, *general services*, can be divided into two parts:

Consulting. Similar to traditional closed-source software, open source companies can offer consulting on the use of their products. Additionally, consulting on the specific risks and possibilities of open source software can be provided as well [3].

Software Operation. This term combines services whose sole purpose is to enable the operation of software. For example, companies can perform the installation and configuration process of their software at the client's office. Furthermore, they can migrate data from legacy systems of the customer to the new software. If an open source company follows a software-as-a-service approach, it can also offer hosting of the software in their own data centers. Consequently, hosting includes installation and migration.

5 Ranking of Features

One of the major goals of the survey was to rank the product features based on their frequency and their importance. Frequency was measured by looking at the number of products containing an individual feature. Additionally, we asked the participants to explicitly rank the features based on their subjective perception of importance. This enabled us to compare both frequency and importance for each feature category separately. As already mentioned in Section 3, the following findings are not representative due to the limited number of only 15 valid answers.

When looking at the category of legal features, we can see that *maintenance* is both the most frequent and the most important feature, followed by *updates*. *Warranty* is the third most frequent feature, although its importance is rather low (sixth place out of eight features). *Indemnification* and *non-copyleft usage rights* can be found in the lower third of both rankings.

Digital documentation and *additional functionality* are the most frequent features related to intellectual property. Furthermore, both features are also the most important ones. The last two spots in both rankings are occupied by *certification for the use in processes* and *certification of development process*. Both features do not appear in any of the recorded products, thus resulting in a frequency of zero.

Although *unguided support* is offered for every product, it is perceived as being the least important feature – *managed support* wins the ranking for the most important support-related feature. In terms of training, *on-site training* is the most frequent and most important feature.

The two remaining categories – client-specific services and general services – show that their most important features are also the most frequent ones. *Custom implementation* wins in the first category while *installation and configuration* followed by *integration* and *consulting* lead the latter.

6 Discussion and Limitations

Our study initially planned to combine exploratory work with confirmatory work. The survey was supposed to validate the models derived from interviews and literature review. The low response rate of the survey (15 valid responses) does not allow us to claim representativeness of the found model and its validation. Thus, the work presented in this paper provides a model of commercial open source features based on qualitative research only. A validation of its correctness is pending and has been left for future work.

How representative then is the presented work? In a still small segment of the commercial open source firms we chose leading companies as best-suited exemplars of their kind, e.g. SugarCRM or Red Hat. In total, we looked at eight commercial open source firms. As is the nature of theory-generating work, these selected exemplars provided deep insight and allowed us to build the model, but we cannot claim to have achieved representativeness, which also isn't the goal of such work.

One of the surprising results of this work is that commercial OSS offerings aren't that much different from traditional closed-source offerings. Most of what commercial open source firms sell has also been sold by traditional firms – except for non-copyleft usage rights. However, open source companies may have a different perspective on some of these features: "[...] closed-source companies are likely to see warranty as a nuisance since it implies additional expenses. On the contrary, open source companies regard such features as a possible way to generate revenue.“ [7].

Ultimately, the observation about the similarities between the features sold by open source and closed source firms also suggests that our model was able to capture most, possibly all, of the commercially relevant features of open source software products.

7 Related Work

Several papers have addressed the economic relevance of open source software. However, most of them focus on business models as a whole, therefore mentioning concrete product features only as a sideline. This section provides a short description of these features and compares them to our model..

Van Aardt describes thirteen open source business models [4]. By doing so, he also provides possible product features which can be used to generate revenue. First of all, he explains a feature called "packaging" which is similar to *software distribution* in our model. He also addresses *commercial licenses* by looking at the opportunity of dual licensing. Additionally, he describes a feature called "commercial, proprietary software" which is equivalent to *software improvements*. On the service side, he mentions *support*, *training* and *integration* services. Furthermore, he points out that complementary hardware components can be sold.

Some of these services and deliverables are also discussed by Hecker [10]. Additionally, he also mentions features which can be found in our model, including *printed documentation*, *re-branding*, *custom development* and *consulting*. He also discusses complementary online services, which are beyond the scope of our paper.

By classifying 80 open source companies based on their business model, Daffara outlines several product features which are characteristic for specific business models [11]. For example, he identifies a "twin licensing" business model where companies offer non copyleft usage rights as part of their commercial offering. Companies following a "split OSS/commercial products" approach create additional value by offering "proprietary plugins" as part of their commercial products. He also mentions service providers providing features such as training and consulting. Furthermore, he refers to a special type of open source companies labeled "platform providers". In this context, the term “platform” describes the integration of different open source products. Consequently, it is equal to the feature of “software distribution” in our model.

Fitzgerald identifies two major open source business strategies named “value-added service-enabling” and “loss-leader/market-creating” [12]. These strategies include possible product features such as services (*support*, *consulting*), intellectual property (*software improvements*, *software distribution*) and legal features (*commercial license* with *indemnification* and *warranty*).

As mentioned in Section 2, this paper focuses on three particular open source business models – software producers, service providers and distributors. This categorization is based on a paper by Krishnamurthy in which he also discusses possible product features [13]. For example, he mentions *software updates*, *software distribution* and services such as *support*, *training* and *consulting*.

Riehle identifies four major revenue sources for open source companies and provides an overview of their particular product features [14] [15]. The so-called “core product” refers to a dual-licensing approach, i.e. the open source product is sold with a *commercial license*. On the contrary, companies following a “whole product” approach sell an advanced version of their open source product which has *additional functionality*. Furthermore, companies can provide “operational comfort” by charging for supplementary services such as *support*. Finally, companies can sell “consulting services” such as *training* and *documentation*.

Especially dual-licensing and additional *commercial licenses* are a popular object of investigation. Details and legal implications are discussed by many authors, such as Comino and Manenti [16], Holck and Zicari [17], Lerner and Tirole [18] and Välimäki [19] [20].

8 Conclusions

This paper presents a model of commercially viable products features in open source software, resulting from both qualitative and quantitative research steps. This model provides a hierarchical overview of possible product features and discusses their definition, economic relevance and dimensions in detail. Finally, feedback from an empirical survey is used to enhance the model. Additionally, further details on the relation between individual product features are presented to form the foundation for future work on this topic.

Acknowledgments. We would like to thank the interview partners who kindly shared their knowledge with us, thus enabling us to write this paper. We would also like to acknowledge the contribution of Tony Wasserman, who co-interviewed the commercial interview partners with Dirk Riehle. Finally, we would like to thank the participants of our empirical study for their valuable feedback on the model.

References

[1] IDC, Press Release (November 1, 2012), http://www.idc.com/getdoc.jsp?containerId=prUS23771812#.UOqFc3eSHwx (accessed January 4, 2013)

[2] MySQL AB, Sun Microsystems Announces Completion of MySQL Acquisition (February 26, 2008), http://www.mysql.com/news-and-events/sun/ (accessed January 4, 2013)

[3] Helmreich, M., Riehle, D.: Geschäftsrisiken und Governance von Open-Source in Softwareprodukten. In: Praxis der Wirtschaftsinformatik (HMD 283), vol. 49, pp. 17–25. Jahrgang (February 2012)

[4] van Aardt, A.: Business Models on Open Source Software. In: 19th Annual Conference of the National Advisory Committee on Computing Qualifications (NACCQ 2006), Wellington, New Zealand (2006)

[5] Kumar, V., Gordon, B.R., Srinivasan, K.: Competitive Strategy for Open Source Software. Marketing Science, 1066–1078 (November 2011)

[6] Weikert, F.: How To Earn Money With Open Source Software. Friedrich-Alexander University of Erlangen–Nuremberg, Erlangen (2011)

[7] Weikert, F.: Product Features in Commercial Open Source Software. Friedrich-Alexander University of Erlangen-Nuremberg, Erlangen (2011)

[8] Openbravo S.L.U., Trademark Use Guidelines, Openbravo S.L.U. (December 10, 2008), http://www.openbravo.com/legal/trademark-guidelines/(accessed January 4, 2013)

[9] Riehle, D.: Controlling and Steering Open Source Projects, pp. 91–94. IEEE Computer Society (July 2011)

[10] Hecker, F.: Setting Up Shop: The Business of Open-Source Software. IEEE Software, 45–51 (January 1999)

[11] Daffara, C.: Business Models in FLOSS-based Companies. In: s Workshop presentation at the 3rd Conference on Open Source Systems (OSS 2007), Limerick, Ireland (2007)

[12] Fitzgerald, B.: The Transformation of Open Source Software. MIS Quarterly, 587–598 (2006)

[13] Krishnamurthy, S.: An Analysis of Open Source Business Models. Perspectives on Free and Open Source Software, 279–296 (2005)

[14] Riehle, D.: The Commercial Open Source Business Model. In: Nelson, M.L., Shaw, M.J., Strader, T.J. (eds.) AMCIS 2009. LNBIP, vol. 36, pp. 18–30. Springer, Heidelberg (2009)

[15] Riehle, D.: The Single-Vendor Commercial Open Source Business Model. Information Systems and EBusiness Management (2012)

[16] Comino, S., Manenti, F.M.: Dual Licensing in Open Source Software Markets. University of Trient, Trient (2007)

[17] Holck, J., Zicari, R.V.: A Framework Analysis of Business Models for Open Source Software Products with Dual Licensing. In: Copenhagen Business School Department of Informatics, Frederiksberg, Denmark (2007)

[18] Lerner, J., Tirole, J.: The Scope of Open Source Licensing. Journal of Law Economics and Organization, 20–56 (2005)

[19] Välimäki, M.: Dual Licensing in Open Source Software Industry. Systemes d'Information et Management (2003)

[20] Välimäki, M.: The Rise of Open Source Licensing A Challenge to the Use of Intellectual Property in the So ware Industry. Turre Publishing, Helsinki (2005)