Management Decision

From open source in the digital to the physical world: a smooth transfer?

Nizar Abdelkafi, Thorsten Blecker, Christina Raasch,

Received March 2009
Revised August 2009
Accepted August 2009

From open source in the digital to the physical world: a smooth transfer?

Nizar Abdelkafi and Thorsten Blecker
Institute of Business Logistics and General Management, Hamburg University of Technology, Hamburg, Germany, and
Christina Raasch
Institute of Technology and Innovation Management, Hamburg University of Technology, Hamburg, Germany

Abstract

Purpose – The purpose of this paper is to investigate the transferability of the open source principles of product development from the realm of software to the realm of physical products.

Design/methodology/approach – Based on the inherent differences between software and physical products, a theoretical discussion of the challenges that face the implementation of open source principles in the physical world are provided. A multiple case study methodology is adopted to provide insights into the applicability of the open source concept in product development outside software.

Findings – Many of the challenges identified theoretically are actually encountered in practice. To cope with these challenges effectively, hardware design activities can be translated into software development tasks, using programmable hardware. When dealing with open source projects in the physical realm, it is useful to distinguish between projects driven by commercial firms and those driven by individuals, as each project type can impose different conditions on successful implementation.

Originality/value – Although much scholarly attention has been devoted to open source software, the issue of transferability of the identified principles to other industries has undergone little in-depth research. This paper provides a solid foundation for further investigation of this topic based on theory and empirical case examples. It derives recommendations for industrial experts wishing to benefit from the open source model in new product development.

Keywords Computer software, Product development

Paper type Research paper

1. Introduction

Over the last decades, firm boundaries have become increasingly permeable, opening up companies to external partners. This opening process can be witnessed at several levels, especially business models and new product development (NPD). New business models are increasingly involving customers into the value adding process. Build-to-Order and mass customization (Pine, 1993), for instance, integrate customers into production by enabling them to specify their requirements in terms of product variety, quantity, and delivery time. These models, if implemented successfully, enable firms to outpace their competitors (e.g. Abdelkafi, 2008). In product development, Chesbrough (2006) coined the comprehensive concept of "open innovation" emphasizing that valuable ideas do not only originate from a firm's

Management Decision
Vol. 47 No. 10, 2009
pp. 1610-1632

0025-1747
DOI 10.1108/00251740911004718

internal R&D, but also from other external sources, and that they can be brought to market from inside or outside the firm. Companies collaborating with external partners, suppliers and customers in NPD can achieve competitive advantages in terms of development cost, time, and product quality.

More recently, however, open innovation trends have begun to go beyond contract-based NPD partnerships to encompass more "extreme versions" of opening (Gassmann, 2006, p. 227). Firms open their innovation process by freely revealing product designs to solicit ideas and solutions for modifications and improvements from hobbyist and professional developers. Due to advances in information and communication technology, it is no longer necessary that only commercial firms initiate and control product development projects. In an era, in which social production (Benkler, 2006) has increasingly gained in importance, loosely connected individuals collaborate freely over the internet to create high-quality designs or contents. The online encyclopedia Wikipedia illustrates the power of collaboration via the internet (Malone, 2004; Tapscott and Williams, 2007). In the realm of software, a plethora of programs and applications have been developed following the "open source model" (Osterloh and Rota, 2007). According to this novel development model, geographically dispersed programmers improve the source code and freely reveal their results, thereby contributing to the common code base.

The open source model has proven to be successful in the software realm (Baldwin and Clark, 2006). A fair body of research investigates many facets of the open source software (OSS) phenomenon, particularly the motivations of developers to contribute, project governance, and the role of open licences as a crucial institutional foundation. Although much of this work assumes that at least some aspects of the OSS model can be generalized, little is known about the transferability of this model to the development of physical products. Because of inherent differences between software and tangible products, there is little evidence that a process serving for the creation of operating systems can be emulated in car development. More importantly, little is known of the factors influencing the success of such projects.

This paper aims to investigate the transferability of the open source model of development from the software industry to the world of physical artefacts. To achieve this goal, we structure the paper into eight main sections. Following this introduction, we present a brief description of OSS. Section three provides a theoretical analysis of the transferability of the open source model to the physical realm. Starting from the differences between physical products and software code, we trace the challenges that may face the creation and production of physical products according to open source principles. In section four, we describe the methodology of our empirical work. Section five provides an overview of five case studies that apply the open source model to the development and/or production of physical products. In sections six and seven, we discuss the case studies and make recommendations for industry practice. Section eight concludes and provides directions for future research.

2. Open source software

The free software movement was initiated in the 1980s by Richard Stallman. The word "free" refers to the freedom of download, utilization, modification, and redistribution of source code. According to Stallman (2002, p. 34), free code does not only save users the price of proprietary software; it also eliminates wasteful programming efforts that can

be reallocated to advance the state of the art. To avoid confusion of "free" with "gratis", the term "open source" has been introduced. What is crucial for collaborative OSS development is the revelation of software in the form of source code. If only the machine code – a chain of binary digits that result from the compilation of source code – is made available, code inspection, modification and improvement become impossible.

There has been a large body of research investigating the OSS phenomenon. A question that especially interests researchers concerns the motives inducing geographically dispersed developers to write code and thereby advance the development project without any expectation of recompense. Intrinsic motivation and altruism – the wish to help others – are factors that drive developers to contribute to the production of software. The resulting software programs sometimes exhibit stronger performance than any competing proprietary code (Benkler, 2006).

Currently, over 170,000 projects are registered at the SourceForge web site. Many OSS projects such as Linux, Mozilla Firefox, Apache web server, and MySQL provide evidence of the potential of this new phenomenon for success, while a majority of OSS projects never attract a sufficient number of co-developers (Healy, 2003).

Successful OSS projects feature many typical practices. Raasch *et al.* (2008) identify five groups of features that matter when initiating such a project:

(1) the motivation of developers;

(2) the conditions of contributing, e.g. rules of access to the community;

(3) information sharing;

(4) project governance and organization, e.g. self-organization or leadership; and

(5) technical prerequisites of the object under development and of the tools enabling co-design, coordination and communication.

The OSS model of development has been discussed within different contexts and intersects with other models of collaborative development, introduced in the innovation and technology management literature. For instance, the private-collective innovation model by von Hippel and von Krogh (2003) describes a concept, in which private resources are spent to contribute to the production of a public good. Within OSS development projects, innovations are created because software programmers are investing their own resources in order to write source code that is made available to everyone for free. Similarly, the "commons-based peer production" concept by Benkler (2006) overlaps with OSS model. "The quintessential instance of commons-based peer production has been free software" (Benkler, 2006, p. 63). In Benkler's model, the inputs and outputs of the development process are shared freely, whereas individual action is self-selected and decentralized. Consequently, the OSS concept can be seen as an instance of more comprehensive models that may not only serve for the development of software, but also for the creation of any type of products.

3. Beyond open source software: open source development in the physical realm

3.1 Open source development outside software: some preliminary thoughts

The term "open source" is related to the source code, which is downloaded, used, modified, and distributed free of charge. The production of purely physical goods,

however, does not involve any source code, but technical drawings and manufacturing specifications. Consequently, it is not accurate to use the open source term to designate tangible goods that are designed in an open and collaborative way. It should be noted that a generally accepted terminology for this new field of application has not been established yet, although the term "open design" (Vallance *et al.*, 2001) is becoming more widely used.

Viewing the physical and digital realms as unrelated worlds is inappropriate because software has increasingly permeated physical products. In many industries, products are a combination of both hardware and software. Hardware functionality can be altered by installing modified or new software on electronic devices. Shirky[1] therefore notes that "[a]n increasing number of physical activities are becoming so data-centric that the physical aspects are simply executional steps at the end of a chain of digital manipulation." In general, products can be placed on a continuum, reaching from pure hardware to pure software. Consequently, a project following open source principles of development may simultaneously tackle hardware and software designs. The community then creates, modifies, and distributes software code as well as product design and documentation such as technical drawings.

The software content of physical products may vary depending on the type of industry. No matter how much digital code is embedded into the product, many product functions cannot be performed unless the artefact has a physical existence. Therefore, we should provide compelling arguments why the open source model of product development, as pioneered in the software industry, can be applied to a world that is purely physical.

First, "physical products consist of knowledge and information at the design stage – one can freely reveal CAD models of an airplane design as readily as one can freely reveal software code" (von Hippel and von Krogh, 2003, p. 219). Second, suitable communication technology, promoting smooth information exchange and collaborative design among geographically dispersed developers, is available for many fields. Third, communities around physical products, especially user groups collaborating to improve existing products and develop new artefacts, have been around for a long time (Lerner and Tirole, 2002). von Hippel (2005) goes further to state that most important innovations originate from product users. Note, however, that open source and user-driven innovation projects exhibit a small difference, but with substantial consequences. Open source projects do not restrict participation to product users and involve the participation of any individual or group who can benefit the project. Therefore, the base of participants can exceed the base of contributors in user-driven projects, thus imposing several management and coordination challenges.

Some authors explicitly refer to the applicability of open source principles outside the software realm (Garcia and Steinmueller, 2003; von Hippel/von Krogh, 2003; Shirky, 2005; Maurer and Scotchmer, 2006; Carr, 2007; Chesbrough and Appleyard, 2007; Nuvolari and Rullani, 2007). For instance, Ulhøi (2004, p. 1108) states that "open source innovation is not only confined to the domain of software development, which is the impression one may easily get from the still rather limited literature on open source projects."

Because the academic literature covering this subject is still scarce (one notable exception is the work of Hope (2004) in the field of biotechnology), we conducted a set of preliminary interviews with more than 25 experts from academia and practice to

better understand the concept. Our investigation led to the conclusion that the dimensions that matter when studying the feasibility of open source projects in the physical realm are the properties of the object and the characteristics of the developer community, independently of the type of industry.

Recapitulating, we have advanced some arguments supporting the assumption that the open source model of development can be applied to fields other than software. Nevertheless, many questions remain unanswered: For instance, are all the principles of OSS development transferable one-to-one to the physical realm? Are there any barriers that impede this transfer? What are the peculiarities of the digital world that may particularly favor open source development? Conversely, are there characteristics of physical goods that are alien to software and may be conducive to open source development? In this paper, we mainly focus on the challenges and barriers that face the application of open source principles to the development of products in the physical world.

3.2 The differences between physical and software products

To investigate the transferability of open source principles to the physical realm, we begin by examining the differences between software and tangible products, focusing on factors supportive of open source development in the digital world. These differences can be used to examine how far these factors are expected to prevail in the development of physical products.

To determine these differences, we should note at first that software represents an information product. At the machine level, software is merely a chain of zeros and ones, and in the mathematical theory of communication, pioneered by Shannon (1948), information is also represented by binary digits. Consequently, software inherits all the properties of information, which have been widely discussed in the academic literature (e.g. Shapiro and Varian, 1999).

In total, we identify nine essential features discriminating between software and tangible products: lifetime, modularity, material supply, production, distribution, inventory, replication, cost structure, and patenting. For ease of exposition, we integrate material supply, production, distribution, and inventory into one feature called supply chain.

Lifetime. In contrast to the world of digits, the world of atoms obeys the laws of physics (e.g. Springer, 2008). Physical products wear out due to friction, for instance, but software does not. Therefore, tangible products have limited lifetimes, whereas software can be used indefinitely. Software is replaced not because it is no longer functional, but because more powerful versions have been released (Carr, 2004).

Modularity. The pace of the development of artefacts, be they in the digital or physical world, is accelerated, if designs are modular (e.g. Ericsson and Erixon, 1999). Modularity means that the artefact's building blocks are loosely coupled, in the sense that interactions between modules are minimized (Baldwin and Clark, 2000; Baldwin, 2008). Software is, in fact, written in programming languages that promote loose coupling and independence of software modules. Object-oriented programming languages, in particular, increase software modularity and data abstraction; they also support encapsulation due to rigidly defined and strongly enforced interfaces to objects (Muckelbauer and Russo, 1995). Another programming paradigm that further supports software modularity is aspect-oriented programming. Modularity research in physical

products concentrates on development methods that drive object modularity (Abdelkafi, 2008). Most methods, however, provide guidelines to modularize an existing product. We are aware of a single method (Dahmus *et al.*, 2001) that supports the design of modular architectures and can be applied at the very early stages of product development.

Supply chain. A physical product cannot be created without an adequate supply chain. The network of upstream suppliers should be coordinated to ensure a smooth movement of materials from one step in the supply chain to the next. The production system transforms the input factors such as materials and components to a tangible output, which is transported to the final customer via various distribution channels. Along the supply chain, stock points hold inventories of components, sub-assemblies, and finished products. In contrast, software is "manufactured" as it is written; programmers can run the code, test it, and improve it, using a computer and a compiler. Because software is a set of binary digits, it only occupies a server or disk drive space, but does not induce any inventory in the traditional sense. Furthermore, distribution is supported by the internet. The program can be made available instantaneously to all internet users.

Replication. In contrast to physical products, digital software can be copied perfectly. The same digits can be reproduced indefinitely without any loss of quality. In the area of manufacturing, however, the output quality of the production process can fluctuate considerably.

Cost structure. The cost structures that are induced by manufacturing and software production are very different. The cost of the first software copy is very high, whereas the marginal cost of any subsequent copy is extremely low (Shapiro and Varian, 1999). Physical products are different in this regard. While there is also high up-front investment and high design cost, marginal costs still represent a substantial portion of total costs, in spite of scale economies and learning curve effects.

Patenting. The speed at which software development is conducted is very high. The prediction that patents slow down the pace of innovation in the software industry is one of the reasons why Stallman (2002) considers software patents disadvantageous. Patented code and algorithms prevent the use of the underlying ideas without the express permission of the patent holder during the period of 20 years after their creation. Patenting has been transferred from the physical to the digital realm without considering the peculiarities of software products and the process by which code is developed (Stallman, 2002) (see Table I).

3.3 Transferability of open source to physical production: a theoretical viewpoint

Based on the specific features of tangible products and software, we can analyze the question of transferability from a theoretical viewpoint. As will be shown in the following, most of the identified differences impede this transfer and represent a barrier against the application of open source principles to the physical realm. Moreover, we identify an additional factor, principally related to the hacker culture that render software more suited to accommodating open source principles than physical products.

The impact of lifetime differences on transferability. Product lifetime is related to the issue of warranty. As physical products cannot be functional forever, manufacturers guarantee to their customers that products can operate adequately for at least a certain

Feature	Product type: Software	Product type: Physical good
Lifetime …	… is unlimited	… is limited
Modularity …	… is driven by object-oriented and aspect-oriented programming	… is very difficult to ensure at the very early phases of product design
Supply chain		
Material supply …	… is not needed	… is needed
Production …	… requires only a computer and compiler	… requires manufacturing and assembly equipment
Distribution …	… occurs via an information exchange platform, e.g. the internet (instantaneously)	… occurs via traditional distribution channels (after production; with delay)
Inventory …	… consists of a chain of digits placed on a data storage medium or uploaded on a web server	… consists of components, semi-finished, and finished products placed at many stock points along the supply chain
Replication …	… is perfectly done by making copies of binary digits	… is not perfectly done because the outcome of the production process fluctuates
Cost structure …	… involves low (almost no) marginal costs	… involves marginal costs representing a substantial portion of total costs
Patenting …	… is ambiguous and represents a barrier against software development as it strongly inhibits incremental innovation	… is less problematical than in software, as long as one product leads to one patent

Table I.
The differences between software code and physical products

time period. During this period, if they are defective, a replacement or a reparation service is offered free of charge. Customers expect this service, even when the product is developed according to open source principles. This service, however, can induce additional costs that the open source project may not be able to shoulder. These costs are particularly high, if the maintenance service should be carried out on site. Software code, by contrast, is downloaded and run for free whereas maintenance or upgrade services (e.g. version updates and patches) can be distributed over the internet at virtually no cost.

One of the factors contributing to the unprecedented success of Linux is that a set of businesses providing maintenance and administration services have developed around the operating system (e.g. Weber, 2004). Therefore, can such a model, which consists in giving products for free to all customers and then making money by selling services around it, work in the physical world? This depends on the fact if the revenues over the product lifecycle compensate initial product costs. Furthermore, at the start of the business capital is necessary to bridge the first period of no revenues, when products are manufactured and shipped free of charge.

The impact of modularity differences on transferability. A further important feature, at which physical products and software can differ, is modularity. Since a modular product is decomposable into modules, the whole development task can be split into independent and granular activities with low levels of complexity. In the open source context, modularity implies that projects can be divided into small subprojects, on which dispersed designers can work autonomously. Software has the advantage of being produced within a framework that supports modularity. By analogy,

programming language for software is comparable to matter in tangible goods. To see the advantage of software in this regard, one should imagine a kind of matter that naturally improves the modularity of physical products.

The impact of supply chain differences on transferability. The requirement of a supply chain, which receives orders, procures components and materials, manufactures the product, and finally transports end items to customers can complicate the practical implementation of an open source project in the physical realm. Manufacturing and assembly require high initial investments that may be unavailable at the beginning of the project. Furthermore, though many business functions such as purchasing, logistics, accounting, etc. can be (partly) outsourced, a lot of efforts and know-how are still required to coordinate the supply chain network. Running a supply chain effectively and efficiently calls for experienced managers who, among other tasks, conduct daily business operations (production and inventory control, management of supply, quality checks, etc.). The lack of resources at the disposal of an open source project can induce many difficulties of supply chain management. Software production, however, has not to cope with these challenges, as writing code in itself does not involve any operational management activities.

The impact of replication differences on transferability. The success of OSS projects is due, among other factors, to quick bug-fixing: "Given enough eyeballs, all bugs are shallow" (Raymond, 2001, p. 30). As code can be perfectly replicated and instantaneously distributed, developers can make frequent and continuous improvements, the pace critically depending on the release schedule, i.e. the periodicity of new official releases. In OSS development, code can (and should) be released early and often (Raymond, 1999). There are two types of releases: stable and experimental versions. Stable releases are intended for users since they offer stable functionality, whereas experimental releases serve as a test bed for new features (Lee and Cole, 2003). For physical products, these soft release practices, which are central to OSS development, cannot be implemented so effectively. Though technical drawings can be replicated, modified and released frequently, the product itself usually cannot. But many design problems and errors in production are not detected until the product is manufactured. Because of this, rapid problem-fixing is difficult.

The impact of cost structure differences on transferability. Extremely low marginal costs, virtually equal to zero, render it possible for OSS to be distributed free of charge. The audience of OSS is not subject to any constraint; every person interested in the software can make a copy and then run it. In the physical realm, however, a price is charged, and this evidently restrains product diffusion. Therefore, OSS is likely to appeal to much more users than physical goods created according to open source principles. Thus if we measure project success by the size of the community using the product, OSS projects will do much better. With respect to the transferability question, the main topic of this paper, this means that projects in the physical realm will not be able to achieve the same success that known OSS projects achieved.

The cost issue may also trigger a far-reaching problem. If an industrial firm freely reveals its designs and initiates an open source project, competitors may use the project outcome to produce and sell goods, while making profits. Competitors cannot be prevented from marketing these products and asking for a price. Thus the company initiating the project is exposed to the risk of profit losses. This risk can represent an

essential factor discouraging companies from embarking on open source projects. In the software realm, these issues have been already addressed by adequate and effective licensing schemes, which are still missing in the physical world.

The impact of patenting and licensing scheme differences on transferability. The negative impacts of patents on incremental innovation and pace of progress are more serious in the digital than the physical world. Large software programs with millions of lines of code typically combine so many ideas and algorithms that the likelihood of infringing an already existing patent is very high (Stallman, 2002). Consequently, the problems caused by the patenting system can be expected much harder to solve in the OSS case. As opposed to expectations, clear licensing schemes have actually emerged in the realm of software. To regulate the rights and duties of the community, OSS projects have a whole range of standardized open licences to choose from. To date, 72 licence texts have been accredited by the Open Source Initiative (www.opensource.org). Under all licences, OSS code is a commons open for all to use and modify. Differences lie in the extent to which public property may be combined with proprietary solutions. In the copy-left model, derived works are required to be released under the same licensing scheme as the original work. As a consequence, a software company is not allowed to modify an open source program and then market it in proprietary fashion. Under the second model, called permissive licence, the original program and its derivatives may be subject to different licensing schemes (St Laurent, 2004). In contrast to this well-developed menu of software licences, licensing schemes for physical products are still to be developed. Markus Merz, the initiator of the OScar (Open Source Car) project, states that licensing is a very complex topic, and that it is often difficult to determine, whether the licence should exclude product commercialization or not[2].

A further factor affecting transferability. In a survey related to the Linux kernel, respondents state that one of the main reasons driving their participation in OSS development is their belief that information should be free (Lee and Cole, 2003, p. 645). This belief is, in our understanding, incorporated in a common culture that many software programmers share. Raymond (2001) explains the main principles hackers are supposed to respect in their community. The first principle, for instance, is "write open-source software" (Raymond, 2001, p. 204). In general, all the principles in the hacker culture can be said to be conducive to OSS. In the physical realm, however, there is no noticeable common culture among product developers. We believe that the absence of such a culture promoting openness and free revealing of ideas makes the transfer of the open source principles to the physical realm less evident.

This theoretical analysis shows that the inherent characteristics of software are conducive to the open source model of product development. There are, however, many projects in the physical realm that are worth investigating. For many of them, a prototype has been developed; other projects resulted in marketable products that went into production and distribution.

4. Methodology

Although the number of projects currently using the principles of open source development in the physical realm is increasing, little scholarly attention has been devoted to this field so far. The scarcity of works in this area motivates exploratory research, aiming to generate new knowledge and valuable insights. The case study

methodology is particularly suitable for exploratory studies investigating new areas where theory is still developing (Eisenhardt, 1989; Yin, 2003). Case studies represent the best approach to explore the theories-in-use when existing literature does not lead to new propositions (Dul and Hak, 2008).

Case studies support theory building and are suitable to examine "how" and "why" research questions (Yin, 2003). In this research, the unit of analysis is a project applying the open source principles to develop a physical good. Out of a considerable number of existing projects, we select five case studies. We argue that five cases are sufficient to achieve the objectives of exploration. This research should lead to new insights that may serve as an input to a large-scale empirical study in the future.

Three criteria have been used to select the sample of projects to be considered in this study. The first criterion is the position of the project on the lifecycle axis, which extends from development over prototyping and production to sales and distribution. This criterion ensures that projects at early and advanced stages are available in the sample. The second criterion is related to the commercial support that the projects benefit from. Projects initiated by industrial companies have a better access to financial and organizational resources than projects created by individuals. This criterion is especially important because the availability of resources is expected to have a strong effect on the success of a project. The third criterion is related to the degree of involvement of software development in the project. Some projects only develop open source software code to be used with the physical product whereas hardware design remains proprietary closed. Other projects develop both software and hardware according to the open source principles. In our sample, we consider both types of projects.

The theoretical analysis presented in the previous sections serves to guide the development of the case studies. Using the insights from theory, we investigate whether the difficulties expected to face the projects developing physical products in an open fashion are actually available in practice. More importantly, are there additional factors impeding these projects or helping them to overcome the challenges of developing functional tangible goods?

Our data collection relied on two sources: first secondary data, such as project web sites and project-related articles, and second interviews with project leaders or core-team developers. Interviews were conducted based on a questionnaire with open questions. The responsible persons were contacted via e-mail. After a short description of our research objectives, the contact person was asked for a telephone interview. If requested, the interview questions were sent in advance. Then, an appointment for a phone call was scheduled. The interviews were semi-structured, thus allowing for additions to the initial set of questions. With this procedure, the interview develops dynamically depending on the answers of the interviewees. The interviews have been conducted by November 2009. The interviewed persons had high positions in their projects. Since the projects were relatively small with a few people in the core team, it was not necessary to interview further persons to check the accuracy of collected information. The triangulation of data collection methods: interviews, Internet websites and publications have principally contributed to increase the reliability and validity of the results. In order to avoid any information loss, all interviews have been recorded and transcribed.

5. Case studies

In this section, we briefly describe the case studies conducted within the scope of this research. Table II provides an overview of each project, showing the project name, Internet link to the project website, type of product, stage of the project lifecycle, position of the responsible person contacted to make an interview with, and all the information sources used to carry out the case study.

5.1 Openpandora

The Openpandora project has been founded by seven people, making up the core team, with backgrounds from different disciplines such as programming, design, and marketing. Openpandora is a handheld pocket gamer. With the integrated Linux operating system and emulators for Playstation, the console is primarily intended for games, but can also be used as a mini-notebook, since it is equipped with a keyboard. The Pandora project has been a group effort with a community of about 30,000 people at the beginning of the project. The hardware development team designed openpandora based on specifications requested by potential users. The device is manufactured by a mass production company and then sent to customers by the project team. The first production run accounts for about 3,000 units. Concerning software, people can use the existing programs to run the machine or can modify them according to their requirements.

5.2 Elphel

Elphel, Inc. was started in 2001 to provide high performance cameras based on free software and hardware designs. Elphel cameras can be used as an out of the box solution. Furthermore, Elphel grants its users the freedom to modify any part of the product. The cameras of Elphel are protected by the GNU General Public Licence, covering both software and hardware. The licensing scheme protects the user's right to create and distribute derivative products. In this way, many applications can emerge that the company's founder himself never thought about.

Elphel cameras use Xilinx programmable logic devices. These devices represent a frontier between hardware and software; they offer the opportunity to apply the principles of open source software solutions to hardware design (Filippov, 2003).

5.3 RepRap

The RepRap project was initiated by the University of Bath in 2004. It strives to design and produce a machine that replicates itself; more specifically, a 3D-printer that can produce most of the parts needed to assemble an identical machine. The price target of RapRap is US$400 and thus considerably below the costs of a commercial 3D-printer (about US $20,000). The first fully functional, self-replicated copy of RepRap was completed in June 2008. Currently, the objective is to reduce the number of third-party components, to improve the printing technology, and to expand the community.

At the beginning of the project, the community of the volunteers was small. As the community grew, a core team was formed to drive and coordinate development. Tasks are mostly self-selected by developers, and decisions are usually taken by voting within the core team. Both development and production of the 3D-printer are decentralized, being accomplished by geographically dispersed project members who communicate over the Internet (Raasch *et al.*, 2009).

Project name	Openpandora	Elphel	RepRap	OpenEEG	Neuros OSD
Website	www.openpandora.org	http://www3.elphel.com/	http://reprap.org	http://openeeg.sourceforge.net/	http://www.neurostechnology.com/osd
Product	Handheld pocket gamer	High performance cameras	3D-Printer	Electroencephalography devices	Media centre
Project lifecycle stage	Sales and distribution	Sales and distribution	Development and prototyping	Prototyping and production	Sales and distribution
Interviewed person	Main project responsible	Project founder and leader	Project founder and leader	Developer	Chief executive officer
Sources of information	Phone interviews project web site	Phone interviews project web site published articles	Phone interviews project web site	Phone interviews project web site	Phone interviews project web site

Table II.
Overview of the examined projects

5.4 OpenEEG

EEG (electroencephalography) devices capture neuron feedbacks, enabling the visualization of the activities of the human brain when doing a mental task. The outcomes of the project are design plans and software for do-it-yourself EEG devices available for free. Information on the hardware, technical schematics, bills-of-materials, building instructions, etc. is posted on the website, and any interested party can use the plans to build the device. Developers from all over the world are collaborating to advance the product based on a GPL. Members of the community have also developed software, which is hosted on their own websites. OpenEEG costs less than one-third of a similar device on the market. A major issue in this project is security. The project initiators declare that they are not responsible for any damage resultant from using OpenEEG.

5.5 Neuros OSD

Neuros Technology is a Chicago-based company with approximately 35 employees. It sells audio and video devices, among them the Neuros OSD, an open source, Linux-based, embedded media center. The device records digital content from various sources and then stores and outputs this content in a standardized format. Neuros opened its design specifications and development process to a community of users to obtain feedback, to track bugs, and to promote co-development. Some parts of the design are not open because they are either outsourced to suppliers or critical to corporate business strategy.

At present, a new generation of Neuros OSD is being developed. While most of the work is done in-house, user solutions are integrated as available. Additionally, contract-based development assignments to users and the shipping of testing versions (either for free or with costs, depending on development stage) to user developers support their involvement (Raasch *et al.*, 2009).

6. Discussion

Many of the difficulties of applying the open source principles to physical goods that we identified theoretically have been confirmed by our field investigation. In this section, however, we focus on the problems that our theoretical analysis did not reveal and what the projects did in practice to be successful.

Slow pace of development

Product development in the projects we studied progresses at a relatively slow pace. Our respondents find it difficult to stick to time plans; in fact, some of the projects did not even develop a schedule. The acceleration of development speed is rendered difficult in projects relying on volunteer contributions. Duplication of effort, though advantageous for product quality, further slows down projects. One interviewee confirms this problem:

> Sometimes we have two people doing similar things or undoing each other's work or even just misunderstanding.

Consequently, it is unexpected to see in the foreseeable future that open source projects design and deliver products requiring high responsiveness. For instance, the fashion industry should be built on extremely fast supply chain networks. Small delays can

result in big economic losses for firms in fast-moving industries. Relying only on volunteers is therefore risky in such an environment, in particular if the community does not reach the critical mass that ensures fast development progress.

Requirements on Kernel

In order to initiate a project working according to the open source model of development, a starting point that guides the design efforts is necessary. By analogy to the kernel, which is the central component in operating systems, we expected that the basic design from which development in the physical realm sets out will have a similar structure: a product platform (main product module or "kernel"), around which developers can create new modules. The platform would correspond to the structural backbone that serves as the starting point for developing a product. In the Openpandora, Elphel, and Neuros OSD cases, however, we found that the starting points – "the kernels" – were not platforms, but complete and functional products. They were developed internally by the project founders. In the other projects, RepRap and OpenEEG, product design grows progressively with community contributions. Combined with the fact that the first three projects pursue commercial interests whereas the other two do not, one could hypothesize that, at least for commercial open source projects, a functional, albeit imperfect basic design promises higher success rates. Projects that are not profit-driven, however, may start with a not marketable prototype, or even with much less: for instance, a list of requirements to guide the development efforts. This hypothesis is crucial and requires more in-depth analysis, as it addresses the least requirements the "kernel" should satisfy when launching an open source project in the physical realm. In the software area, Goldman and Gabriel (2005, p. 256) raise a similar issue: Should initial software code work or not? They recommend releasing functional code, as it better enables incremental improvements. "The classic complaint about Mozilla in its early days was that the source code Netscape released was incomplete and could not make a working web browser" (Goldman and Gabriel, 2005, p. 256).

Use of programmable hardware

Open source projects tackling the development of physical products in many instances do not only deal with hardware, but also with software development. In the OpenPandora case, for example, the community contributes suggestions to hardware design, and actively participates in writing code. With programmable hardware, it is possible to translate some hardware design tasks into software development activities. It is a way to take advantage of the inherent characteristics of software that are conducive to open source development. One of our interviewees illustrates the use of such programmable hardware, as he mentions:

> We constantly upgrade the firmware and so you can download and install newer firmware that includes the code that changes the hardware behaviour.

Similarly, another respondent points out the importance of software in changing the functionality of hardware:

> We discovered that merely by improving the software we can get much better hardware.

Supply and manufacturing more challenging than distribution

Based on theoretical analysis provided above, we predicted that open source projects developing physical products may find it more difficult to cope with distribution costs than OSS projects. According to the case studies, however, distribution is not the main problem; supply and manufacturing prove more challenging. First, operational problems such as inventory, lead times, and quality frequently emerged during our interviews. A second crucial issue is related to the openness of outsourced components incorporated in the design. In order for the community to be able to work on their integration, it is advantageous that the suppliers open their designs. Suppliers, however, are typically averse to revealing detailed information on their components. One project leader explains:

> If they [suppliers] need to make modifications, we like to ask them to just let us know what those are; in particular, of course, the PC-board modifications and the firmware modifications. So we like to have all of this information and we like to have it documented, but there is no way to enforce that.

When manufacturing is outsourced to third-party manufacturers, an open source project tends to have little control over production. OpenPandora, for instance, provided a product upgrade at no extra cost to compensate customers for the Chinese manufacturer's delay in production. In terms of manufacturing the RepRap project is in a unique position. Since the project aims to build a 3-D printer that reproduces its own components, everybody in possession of a machine can act as parts supplier. In this case, a model similar to OSS development is possible, since the community not only develops, but can also produce and distribute many parts of the product.

Product complexity

In the studied cases, product complexity presents a substantial barrier to open source projects developing physical products. Our respondents mention that complexity endangers the success of open source projects as it causes coordination difficulties and increases costs. However, even though an operating system is a very complex system, it has been developed successfully within the scope of an open source project. It seems therefore interesting to investigate whether complexity has a stronger impact in the physical realm than in software. In other words, as we move from purely digital to purely physical in the continuum of products, do the negative effects of complexity on project success assume more weight?

7. Recommendations for practice

We are not studying open source development because it is an end in itself, but because it offers a novel form of organizing for innovation, which may promise advantages over more traditional models (Demil and Lecocq, 2006). New firms with business models resting on open source development can emerge to disturb the leading positions of established firms. Microsoft saw its dominance decline in the field of operating systems due to the rise of Linux, for instance. Firms like Neuros Technology find open source development a suitable means to improving their competitive position. In this section we want to discuss how firms can turn the open source model to use in their new product development.

We distinguish between two fundamental models for commercializing products developed in an open source fashion (Figure 1), depending on when a commercial company eventually joins the project. In the first model, a start-up or an established commercial company exploits the outcome of a pre-existing open source project, but does not participate in the design phase. This commercialization model is successful, only if the project idea and development work of the community are appealing and promise sufficient benefits to attract capital. If this is not the case, the project is unlikely to go into industrial implementation.

The second commercialization model is industry-driven and can be divided into two sub-models. Either an established company creates a platform to initiate the open source design of a new, not yet existing product (sub-model 2a); or the company freely reveals an existing design and invites developers to join an open source co-development process (sub-model 2b). In both cases, the project benefits from the available know-how, financial support, and organizational infrastructure of the initiator.

It is difficult to say which model would be more successful and more frequently implemented, but so far we could observe all these models in practice. For instance, OpenEEG is an example of the first model, since independent developers work on creating and improving software and hardware whereas manufacturing is the responsibility of a third party producer in England. Model 2a is implemented by Openmoko, a project that is initiated by a company producing hardware in Taiwan called FIC (First International Computer) to develop a mobile phone according to the open source principles. The first and second generations of Openmoko mobile phones launched in the market by July 2007 and July 2008 were very successful and sold out within a short period of time. Finally, an example of model 2b is Neuros, which opened its existing designs to enable distributed volunteers to work on the product.

At this point, many questions may be asked: When should companies open their designs and initiate open source projects? Are there businesses that are better suited to open source projects than others? Put reversely, which businesses cannot accommodate open source principles of product development? What can we learn from the case studies and what recommendations for practice can we make?

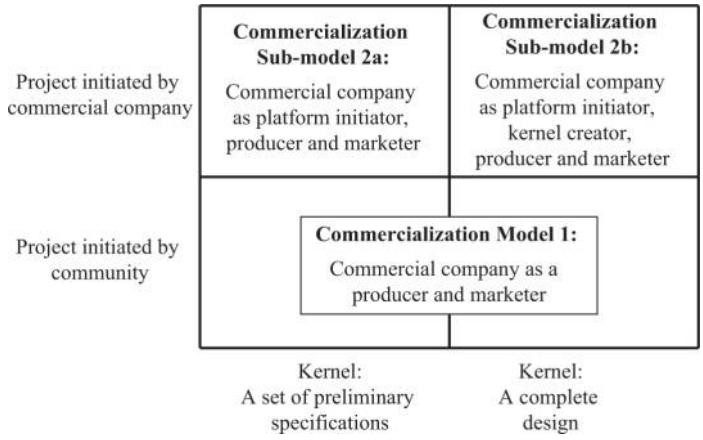

Figure 1.
Commercialization models of products developed according to the open source principles

Managers who are tempted to apply the principles of open source development in their companies should realize that even in the software realm many past projects failed. Very often, management literature focuses on successful examples and overlooks the large number of projects that never took off. "We tend not to hear very much about the failures. Only successful projects attract attention, and there are so many free software projects in total that even though only a small percentage succeed, the result is still a lot of visible projects" (Fogel, 2006, p. 1). In effect, we clearly see the OSS model at work, only if we consider successful projects such as Mozilla Firefox or Linux Kernel (Healy, 2003). Taking both projects as reference when studying the transferability of open source principles to the physical realm can be misleading, as both projects benefited from very specific advantages that industrial goods are unlikely to take advantage of. First, because these programs are essential for running computers and the Internet, they have the inherent potential of becoming very widespread. Second, these projects benefited from a critical mass of developers and from network economies. One main factor that may have favored this is the near-monopoly position of one specific software vendor. It is the criticism of Microsoft, which was a major motivational driver for hackers to develop software and to distribute it for free (Lee and Cole, 2003).

Furthermore, software is a low-involvement product. What actually matters for the users of computers is that software works adequately. Software rarely conveys a social status or affiliation, except for particular cases such as gaming communities. In the physical realm, however, goods can be highly significant to their buyers. For instance, people are proud of being possessors of goods produced by certain manufacturers. In some cases, communities and even cultures have formed around brand names. Open source projects may find it difficult to compete in industry branches, in which the product success highly depends on the manufacturer's name. Is it possible for Openmoko to be competitive with Nokia's and Sony Ericsson's mobile phones or that Elphel achieves such a degree of brand awareness like Canon or Nikon? Many commercial firms take advantage of their respectable brand names and their experiences in the industrial sector. Using this line of argument, manufacturer's designs appear to be superior to open designs in the physical realm? Consequently, should managers disregard the open source principles of design? And if not, what does make open designs an alternative to be seriously considered by industry?

There are at least three reasons, making products developed according to the open source principles very attractive. First, a new study by Fueller and von Hippel (2008) shows that community brands are evolving and that their members are willing to pay price premiums for the corresponding products. Thus, community brands are gaining in importance and becoming more appealing to customers. Second, in many industries, consumers are dissatisfied with the manufacturers' solutions and adapt the product to their requirements (von Hippel, 2005). Consumers are sometimes well-versed developers able to introduce technical improvements. Many producers, however, try to block product "hacking" by keeping their designs closed. Reasons for closed-source strategies range from safety and warranty concerns to the fear of a loss of profit. In such industries, those firms opening product designs can provide a high value-added service to customers, who can change the product in a way it exactly fits their needs. For one of our interview partners, this is the main argument why their designs are revealed freely. In general, companies may achieve a competitive advantage, if they

recognize the benefits of openness in their business. Third, commercial firms can use the open source principles to create very innovative products, unavailable so far, while considerably reducing product development risks. Such a project may start with minimal resources, e.g. a few requirements on the final product. Then, the developers' community can use these elementary requirements to develop alternative designs and elaborate detailed specifications. To lead the project to success, however, the firm should create a communication platform, which enables distributed volunteers to work collaboratively on the project.

Recapitulating, we believe that the use of open source development in the physical realm is adequate for technology products whose development can occur at a slower pace than traditional in-house development. People who are interested in the product should be less sensitive to delivery times and accept to wait. They buy the product because they value the possibility of performing modifications.

Furthermore, the success of open source depends on the extent to which people actively contribute to the project. The more developers are interested in it, the higher the likelihood that it will succeed. Project marketing is, therefore, an important factor to attract the attention of potential developers and customers; i.e. open source projects need to make recruiting a high priority. In contrast to big manufacturing companies with large marketing departments, open source projects may not have the required financial resources to increase the awareness level of potential customers. An efficient way, frequently mentioned by interviewees, is to write articles in magazines and to attract the attention of press.

The application of the open source model to the physical realm can, for sure, boost innovation and improve the competitive position of industrial companies. But successful implementation is not guaranteed. Even in the field of software many projects failed, making the extrapolation of this concept to industrial goods very delicate. Therefore, managers should carefully scrutinize the benefits and challenges of using this model in the case at hand before embarking upon an open source project.

8. Conclusions

This paper investigates the transferability of open source principles of product development from the realm of software to the physical world. From a theoretical viewpoint, there is no a priori reason why the concept should be inapplicable to physical goods. First, physical goods are often information-intensive, especially during the development phase. At this stage, no physical entities, only product-specific information is exchanged among developers. Second, user communities around physical products have existed for a long time. In fact, people have collaborated to improve and create physical goods before the rise of the open source model. Third, advances in information and communication technologies facilitate design collaboration among distributed developers.

Although the open source development of physical goods appears theoretically feasible, we pointed out that software code benefits from inherent characteristics that facilitate the application of open source principles and that may not be present in the physical realm. In particular, software can be self-manufactured and -tested; programmers only need a computer and compiler. Furthermore, software code can be instantaneously distributed over the internet. By comparison, physical products are

more challenging; they require a physical supply chain and high up-front capital investments.

Our field investigation showed that a considerable number of open source projects have been launched to date even in the physical world. Our interviews support the theoretical analysis, directly derived from the specific differences between software and hardware. One important finding is that projects related to physical products need not be limited to hardware but may also encompass software development. In some cases, projects only concentrate on the software side and leave the hardware components proprietary. The purely physical realm involves problems that may hinder the project to take off. Therefore, it is recommended, wherever possible, to use programmable hardware translating hardware design task to a software development activity. In this way, flexibility increases and the frequency, at which true hardware changes need to be implemented, diminishes.

The analysis, with its theoretical and practical parts, enables us to derive several conclusions and to propose recommendations for industry practice. At this stage of research, however, it is too early to make generalizations. To investigate the degree to which the principles of open source are transferable to physical goods, not only successful but also unsuccessful projects should be examined. Nevertheless, it is frequently difficult to obtain data on project failures.

To define when an open source project is to be regarded successful is a tricky issue. There are good reasons to state that Linux, Mozilla, Apache web server, and MySQL have achieved an unprecedented success. But there are more than 170,000 OSS projects, and where to draw the line between failed and successful projects is not an easy issue (Comino *et al.*, 2007). None of the examined projects use metrics for the measurement of success. In the RepRap case, for instance, the single measure is whether the machine works and whether it can reproduce itself. To illustrate how difficult the measurement of success can be, one may ask the question: "which is more successful: a project with ten daily downloads over a long period of time, or a project with only 100,000 downloads during the first day of its release (Israeli and Feitelson, 2007)?

A possible direction for future research is therefore the development of suitable metrics that can measure the level of success of open source projects in the physical realm as well as in OSS. The availability of adequate metrics that capture the success open source projects is a fundamental requirement to do research with practical relevance in the future. Using these metrics, it is possible to make a cross comparison among projects in order to classify them according to their levels of success. The analysis of successful projects enables one to detect best practices and the conditions that are conducive to open source development. In addition, a large-scale empirical study can be conducted in order to generalize the findings concerning the challenges that face open source projects and the factors that may facilitate implementation in practice.

During the discussion of the exploratory case studies, we could also derive many research questions that are worth investigating in the future. For instance, what are the least requirements on kernel to start an open source project with high chances of success? Is this dependent on the fact if the project is profit-driven or not? Does complexity have stronger negative effects on project success, as the product moves from pure software to a pure tangible product? What are the strategies that can

accelerate an open source development project? How can responsible persons improve people's awareness with the project to get more developers involved in the design task?

For sure, many other questions are worth researching in the future. What we can retain, however, is that the organization of product development according to the open source principles can provide companies with an access to a powerful source of innovation. This source rests on volunteers who, wherever located, can work collaboratively using an adequate communication platform. We believe that the success of firms in the future will not only depend on their capabilities of creating innovations inside their own walls or getting innovations from outside organizations and institutions, but also on their abilities to accommodate the open source principles of product development in order to generate innovations.

Notes

1. http://finance.groups.yahoo.com/group/decentralization/message/6967 (accessed 8 March 2009).
2. www.konsensmilch.de/Download.ashx?File = 325cb189-ce8b-493d-aae1-73e30c64733c (accessed 8 March 2009).

References

Abdelkafi, N. (2008), *Variety-Induced Complexity in Mass Customization – Concepts and Management*, Erich Schmidt Verlag, Berlin.

Baldwin, C.Y. (2008), "Where do transactions come from? Modularity, transactions, and the boundaries of firms", *Industrial and Corporate Change*, Vol. 17 No. 1, pp. 155-95.

Baldwin, C.Y. and Clark, K.B. (2000), *Design Rules – The Power of Modularity*, The MIT Press, Cambridge, MA.

Baldwin, C.Y. and Clark, K.B. (2006), "The architecture of participation: does code architecture mitigate free riding in the open source development model?", *Management Science*, Vol. 52 No. 7, pp. 1116-27.

Benkler, Y. (2006), *The Wealth of Networks – How Social Production Transforms Markets and Freedom*, Yale University Press, New Haven, CT and London.

Carr, N.G. (2004), *Does IT Matter? – Information Technology and the Corrosion of Competitive Advantage*, Harvard Business School Press, Boston, MA.

Carr, N.G. (2007), "The ignorance of crowds", *Strategy + Business*, Vol. 47, available at: www.strategy-business.com/media/file/sb47_07204.pdf (accessed 8 March 2009).

Chesbrough, H. (2006), *Open Innovation – The New Imperative for Creating and Profiting from Technology*, Harvard Business School Press, Boston, MA.

Chesbrough, H. and Appleyard, M.M. (2007), "Open innovation and strategy", *California Management Review*, Vol. 50 No. 1, pp. 57-76.

Comino, S., Manenti, F.M. and Parisi, M.L. (2007), "From planning to mature: on the success of open source projects", *Research Policy*, Vol. 36 No. 10, pp. 1575-86.

Dahmus, J.B., Gonzalez-Zugasti, J. and Otto, K.N. (2001), "Modular product architecture", *Design Studies*, Vol. 22 No. 5, pp. 409-24.

Demil, B. and Lecocq, X. (2006), "Neither market nor hierarchy nor network: the emergence of bazaar governance", *Organization Studies*, Vol. 27 No. 10, pp. 1447-66.

Dul, J. and Hak, T. (2008), *Case Study Methodology in Business Research*, Elsevier, Amsterdam.

Eisenhardt, K.M. (1989), "Building theories from case study research", *The Academy of Management Review*, Vol. 14 No. 44, pp. 532-50.

Ericsson, A. and Erixon, G. (1999), *Controlling Design Variants*, Society of Manufacturing Engineers, Dearborn, MI.

Filippov, A. (2003), "How to use free software in FPGA embedded designs", *Xcell Journal*, available at: www.xilinx.com/publications/xcellonline/xcell_46/xc_pdf/xc_freesw46.pdf (accesses 18 December 2008).

Fogel, K. (2006), *Producing Open Source Software, How to Run a Successful Free Software Project*, O'Reilly, Beijing.

Fueller, J. and von Hippel, E. (2008), "Costless Creation of Strong brands by user communities: implications for producer-owned brands", MIT Sloan School Working Paper 4718-08, 8/1/2008, available at: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1275838 (accessed 19 December 2008).

Garcia, J.M. and Steinmueller, E.W. (2003), "The Open Source way of working: a new paradigm for the division of labour in software development?", SPRU Electronic Working Paper Series, Paper No. 92, available at: www.sussex.ac.uk/Units/spru/publications/imprint/sewps/sewp92/sewp92.pdf (accessed 8 March 2009).

Gassmann, O. (2006), "Opening up the innovation process: towards an agenda", *R&D Management*, Vol. 36 No. 3, pp. 223-8.

Goldman, R. and Gabriel, R.P. (2005), *Innovation Happens Elsewhere – Open Source as Business Strategy*, Elsevier, San Francisco, CA.

Healy, K. (2003), "The ecology of open-source software development", paper, available at: http://opensource.mit.edu/papers/healyschussman.pdf (accessed 19 December 2008).

Hope, J. (2004), "Open source biotechnology", PhD thesis, The Australian National University, available at: http://rsss.anu.edu.au/~janeth/OpenSourceBiotechnology27July2005.pdf (accessed 8 March 2008).

Israeli, A. and Feitelson, D.G. (2007), "Success of open source projects: patterns of downloads and releases with time", *2007 IEEE International Conference on Software – Science, Technology and Engineering, Herzliya*.

Lee, G.K. and Cole, R.E. (2003), "From a firm-based to a community-based model of knowledge creation: the case of the Linux Kernel Development", *Organization Science*, Vol. 14 No. 6, pp. 633-49.

Lerner, J. and Tirole, J. (2002), "Some simple economics of open source", *The Journal of Industrial Economics*, Vol. 50 No. 2, pp. 197-234.

Malone, T.W. (2004), *The Future of Work – How the New Order of Business Will Shape Your Organization, Your Management Style, and Your Life*, Harvard Business School Press, Boston, MA.

Maurer, S.M. and Scotchmer, S. (2006), "Open source software: the new intellectual property paradigm", NBER Working Paper Series, Paper No. 12148.

Muckelbauer, P.A. and Russo, V.F. (1995), "Lingua Franca: an IDL for structural subtyping distributed object systems", *Proceedings of the Usenix Conference on Object-Oriented Technologies, Monterey, CA, June 1995*.

Nuvolari, A. and Rullani, F. (2007), "Curious exceptions? Open source software and 'open' technology", in St Amant, K. and Still, B. (Eds), *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives, Information Science Reference*, Hershey, New York, NY, pp. 227-39.

Osterloh, M. and Rota, S. (2007), "Open source software development – just another case of collective invention?", *Research Policy*, Vol. 36 No. 2, pp. 157-71.

Pine, B.J. II (1993), *Mass Customization: The New Frontier in Business Competition*, Harvard Business School Press, Boston, MA.

Raasch, C., Herstatt, C. and Abdelkafi, N. (2008), "Creating open source innovation: outside the software industry", *PICMET 2008 Proceedings, July 27-31, Cape Town*.

Raasch, C., Herstatt, C. and Balka, K. (2009), "The open source model beyond software: comparative case studies on the open design of tangible goods", *R&D Management*, forthcoming.

Raymond, E. (1999), "The cathedral and the Bazaar", *Knowledge, Technology, & Policy*, Vol. 12 No. 3, pp. 23-49.

Raymond, E.S. (2001), *The Cathedral & the Bazaar – Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly, Beijing.

Shannon, C.E. (1948), "A mathematical theory of communication", *The Bell System Technical Journal*, Vol. 27, July/October, pp. 379-423.

Shapiro, C. and Varian, H.R. (1999), *Information Rules – A Strategic Guide to the Network Economy*, Harvard Business School Press, Boston, MA.

Shirky, C. (2005) in Felller, J., Fitzgerald, B., Hissam, S.A. and Lakhani, K.R. (Eds), *Perspectives on Free and Open Software*, The MIT Press, Cambridge, MA and London, pp. 483-8.

Springer, M. (2008), "The future of digital branding", available at: http://camorra.org/fdb/fdb.html (accessed 8 March 2008).

Stallman, R.M. (2002), "The GNU project", in Gay, J. (Ed.), *Free Software, Free Society: Selected Essays of Richard M. Stallman*, GNU Press, Boston, MA, pp. 15-30.

St Laurent, A.M. (2004), *Understanding Open Source and Free Software Licensing*, O'Reilly, Sebastopol.

Tapscott, D. and Williams, A.D. (2007), *Wikinomics – How Mass Collaboration Changes Everything*, Portfolio, London.

Ulhøi, J.P. (2004), "Open source development: a hybrid in innovation and management theory", *Management Decision*, Vol. 42 No. 9, pp. 1095-114.

Vallance, R., Kiani, S. and Nayfeh, S. (2001), "Open design of manufacturing equipment", *Proceedings of the CHIRP 1st International Conference on Agile, Reconfigurable Manufacturing*, available at: www.opendesign.org/CHIRP_Open_Design_Mfg_Equipment.pdf (accessed 8 March 2009).

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

von Hippel, E. and von Krogh, G. (2003), "Open source software and the private-collective innovation model: issues for organization science", *Organization Science*, Vol. 14 No. 2, pp. 209-23.

Weber, S. (2004), *The Success of Open Source*, Harvard University Press, Cambridge, MA and London.

Yin, R.K. (2003), *Case Study Research: Design and Methods*, Sage, Thousand Oaks, CA.

Further reading

Kogut, B. and Metiu, A. (2001), "Open source software development and distributed innovation", *Oxford Review of Economic Policy*, Vol. 17 No. 2, pp. 248-64.

About the authors

Nizar Abdelkafi is a Research Associate at the Institute of Business Logistics and General Management. He holds an industrial engineering diploma from the National Engineering School of Tunis, Tunisia, and a Master degree in Business Administration from the Technische Universität München, Germany. He got his doctoral degree in 2008 with excellence. Nizar Abdelkafi published his work in two books, several conference proceedings and international journals such as: *Electronic Markets, International Journal of Mass Customization*, *Management Decision*, and *IEEE Transactions on Engineering Management*. He also serves as a reviewer for many international journals. Nizar Abdelkafi is the corresponding author and can be contacted at: Nizar.abdelkafi@tu-harburg.de

Thorsten Blecker is a Full Professor at the Institute for Business Logistics and General Management. He holds a Master degree in Business Administration (with honors) and a PhD (*summa cum laude*) from the University of Duisburg, Germany. He finished his Habilitation thesis in September 2004 at the University of Klagenfurt, Austria. Thorsten Blecker is co-Editor and author of several books, e.g. *Production/Operation Management in Virtual Organizations*, *Enterprise without Boundaries*, *Competitive Strategies*, *Web-based Manufacturing* and *Information and Management Systems for Product Customization*. Main research interests: open source innovation, business logistics and supply chain management, production/operations management, industrial information systems, internet-based production systems, mass customization manufacturing systems, strategic management, and virtual organizations.

Christina Raasch is a Researcher and Lecturer at the Institute of Technology and Innovation Management and head of the Open Source Innovation Research Unit at the institute. She studied Economics and Management at St Gallen University, Switzerland, and Oxford University, UK. She received her doctoral degree with excellence from the University of Erlangen-Nürnberg, Germany, in 2006. Her current research is funded by the German Ministry of Education and Research and focuses on open and collaborative models of new product development. She has published her work in several leading peer-reviewed journals as well as books and conference proceedings.

This article has been cited by:

1. Étienne Boisseau, Jean-François Omhover, Carole Bouchard. 2018. Open-design: A state of the art review. *Design Science* **4**. . [Crossref]
2. Julia Amann, Claudia Zanini, Sara Rubinelli. 2016. What Online User Innovation Communities Can Teach Us about Capturing the Experiences of Patients Living with Chronic Health Conditions. A Scoping Review. *PLOS ONE* **11**:6, e0156175. [Crossref]
3. Vinayak Kalluri, Rambabu Kodali. 2014. Analysis of new product development research: 1998-2009. *Benchmarking: An International Journal* **21**:4, 527-618. [Abstract] [Full Text] [PDF]
4. NIZAR ABDELKAFI, SERGIY MAKHOTIN, THORSTEN POSSELT. 2013. BUSINESS MODEL INNOVATIONS FOR ELECTRIC MOBILITY — WHAT CAN BE LEARNED FROM EXISTING BUSINESS MODEL PATTERNS?. *International Journal of Innovation Management* **17**:01, 1340003. [Crossref]
5. Richard Reed, Susan Storrud-Barnes, Len Jessup. 2012. How open innovation affects the drivers of competitive advantage. *Management Decision* **50**:1, 58-73. [Abstract] [Full Text] [PDF]
6. Martin Höst, Alma Oručević-Alagić. 2011. A systematic review of research on open source software in commercial software product development. *Information and Software Technology* **53**:6, 616-624. [Crossref]