Extending Private-Collective Innovation:

A case study[*][+]

Matthias Stuermer

Sebastian Spaeth

Georg von Krogh

ETH Zurich

The private-collective innovation model proposes incentives for individuals and firms to privately invest resources to create public goods innovations. Such innovations are characterized by non-rivalry and non-exclusivity in consumption. Examples include open source software, user-generated media products, drug formulas, and sport equipment designs. There is still limited empirical research on private-collective innovation. We present a case study to 1) provide empirical evidence of a case of private-collective innovation, showing specific benefits, and 2) to extend the private-collective innovation model by analyzing the hidden costs for the company involved. We examine the development of the Nokia Internet Tablet, that builds on both proprietary and open source software development, and that involves both Nokia developers and volunteers who are not employed by the company. Seven benefits for Nokia are identified, as are five hidden costs: difficulty to differentiate, guarding business secrets, reducing community entry barriers, giving up control, and organizational inertia. We examine actions taken by the management to mitigate these costs throughout the development period.

Keywords: Open source software, open innovation, innovation, private-collective innovation, product development, research and development

---

[*] Corresponding author: Sebastian Spaeth, `sspaeth@ethz.ch`, +41-44-6328837

[+] The research was supported by the Swiss National Foundation (grants 100012-101805 & 105512-106932). The authors wish to thank all interview partners for their time and comments, as well as Eric von Hippel and Gideon Markman for providing us with insightful comments and reviews.

Extending Private-Collective Innovation:

A case study

The private-collective innovation model proposes incentives for individuals and firms to privately invest resources to create public goods innovations. Such innovations are characterized by non-rivalry and non-exclusivity in consumption. Examples include open source software, user-generated media products, drug formulas, and sport equipment designs. There is still limited empirical research on private-collective innovation. We present a case study to 1) provide empirical evidence of a case of private-collective innovation, showing specific benefits, and 2) to extend the private-collective innovation model by analyzing the hidden costs for the company involved. We examine the development of the Nokia Internet Tablet, that builds on both proprietary and open source software development, and that involves both Nokia developers and volunteers who are not employed by the company. Seven benefits for Nokia are identified, as are five hidden costs: difficulty to differentiate, guarding business secrets, reducing community entry barriers, giving up control, and organizational inertia. We examine actions taken by the management to mitigate these costs throughout the development period.

1 Introduction

In a private investment model of innovation, firms use internal processes to create ideas, knowledge, and technologies and commercialize these in the market place. Firms appropriate returns from private investment in innovation through intellectual property rights (Granstrand, 1999). This model is contrasted with the private-collective model of innovation, where firms and individuals expend private resources to create public goods innovations. Such innovations are characterized by non-rivalry and non-exclusivity in consumption (von Hippel and von Krogh, 2003; 2006). It is similar to "open innovation," pertaining to models of innovation where firms frequently exchange ideas, knowledge, and technology with outside firms and individuals (Chesbrough, 2003). However, open innovation does not assume that intellectual property rights are forfeited and the resulting innovation is offered to the public for free. Examples of private-collective innovation model include collaborative composing of music on the Internet by many musicians, the open and collective development of a drug formula for treating malaria, or the creation and sharing of new designs for sporting equipment among sports enthusiasts. An oft-cited example of the model is open source software development resulting in products such as Linux, MySQL, or Apache. Open source software comes with licenses that make it non-exclusive: the software is free for all to download, use, modify, and redistribute. Open source software is also characterized by non-rivalry as one person's use of the product does not diminish anyone else's benefits from using it.

Although researchers have examined individuals' motivations to participate in open source software development, to date there has been limited empirical examination of a firm's incentives for private-collective innovation. Moreover, the literature has emphasized the benefits the model brings to the innovator rather than the costs and has not discussed how the latter could be mitigated. Research has shown that the implementation of new models of innovation often have unintended consequences, including "hidden costs" (e.g., Crawford, 1992), and there is a need for more empirical work on the benefits and costs of implementing private-collective innovation.

In this paper, we advance empirical research on the incentives and costs of the private-collective model of innovation. A case study design permits an in-depth investigation of benefits, costs, and actions only partly discussed in prior work (von Hippel and von Krogh, 2006). Our case draws on quantitative and qualitative data from the creation of the Nokia Internet Tablet. Nokia based the development of this product mainly on open source software and made a large part of the research and product development transparent and accessible as a "public- goods innovation." Outside contributors involving firms and individuals, unpaid by Nokia, expended a significant amount of private resources on its development.

The paper is structured as follows. In the next section, we discuss the private-collective model for innovation incentives. The third section describes the research design, and the fourth section contains the case description. The fifth section presents the findings organized along the topics of benefits and costs incurred in the implementation of the model and strategic actions to mitigate these costs. Finally, we conclude the paper and outline implications for management practice and future research.

2 The private-collective innovation model

There are two predominant models of innovation incentives in the technology and innovation management literature. The private investment model assumes that innovators step forward and invest in innovation if and when they can appropriate returns from these investments. Intellectual property rights is a necessary condition for the model because it safeguards returns appropriation (Arrow, 1962; Dam, 1995). In contrast, the collective action model assumes that innovators, provided with the right public subsidy, contribute to public goods innovations (David, 1992; 1998; Stiglitz, 2006). Public goods are characterized by non-rivalry and non-exclusivity in consumption. Innovations are made freely available to all as public goods. Science is often cited as an example of this model. However, companies have the option to free-ride on public goods innovations, such as, for example, a biotechnology company commercializing scientific knowledge on genetics without contributing research back to the scientific community. Therefore, society elects to subsidize the activity of innovators, e.g., university-based research on the human genome funded by the government.

Recently, a third model, called the private-collective model of innovation incentives, has been suggested where public subsidy is absent and where the innovator expends private resources for public goods innovation (von Hippel and von Krogh, 2003). The model is counter-intuitive: why should I make my innovations available to all and why pay for something that anyone else can use for free? Generally speaking, in the case of private-collective innovations the innovator receives higher benefits when contributing to the public goods creation than by only free-riding on its production by others. One aspect is the privately retained tacit knowledge innovators receive through the production of freely available knowledge which distinguishes them from pure users of the explicit knowledge. This implies that firms receive certain benefits during the process of creating publicly available innovations, while the mere application of such knowledge bears less incentives. (see also Grand et al., 2004; Gächter et al., 2006)

2.1 Benefits when applying private-collective innovation

A closer examination of the model outlined in von Hippel and von Krogh (2006) and other literature reveals six complementary benefits for firms to innovate in this manner: the cost of controlling knowledge, learning, reputation gains, and fast and widespread diffusion of innovations, as well as lower cost of innovation and manufacturing. First, in the long run, the cost of protecting knowledge needed to innovate (Liebeskind, 1996) might outweigh the benefits of doing so. Often extensive investments in knowledge management systems are needed to protect information which ultimately and inevitably spills over to the public (Foray, 2004; Alavi and Leidner, 1999). For example, although the source code for Sony's robot dog Aibo was protected, it was ultimately hacked and published by Sony's customers.

Second, innovators that contribute to collective goods innovations benefit from learning from their own and others' contributions. In addition to benefits garnered from the public goods itself (products and services), innovators also benefit from learning in the process of creating it (Allen, 1983; Nuvolari, 2004; Baldwin and Clark, 2006). Thus, it should come as no surprise that many contributors to open source software projects are computer science students. By providing open source software to the public, contributors may get others to use it, test it, and provide feedback on how to improve it (Lakhani, et al. 2002). Some authors have even referred to open source software projects as "epistemic communities," where people create shared knowledge of software development (Edwards, 2003).

Third, innovators may gain a positive reputation by privately expending resources for public goods innovations (Allen, 1983; Lerner and Tirole, 2002; Muller and Penin, 2006). For example, PricewaterhouseCoopers gains a positive reputation amongst regulators, peers, and customers when they provide research to the public on practices of corporate governance. The firm's reputation is further enhanced when regulators actively use and reference the research during standard setting in principles of auditing and corporate governance (PricewaterhouseCoopers, 2005).

Fourth, being the first to contribute a public goods innovation increases the likelihood of benefiting from fast and widespread adoption of the innovation (Allen, 1983). As a consequence, firms may gain an advantage over competitors stemming from network effects (von Hippel and von Krogh, 2003). Establishing a "dominant design" or "open standard" onto which the firm can fit other technologies and even preempt the introduction of competing technology may provide the firm with additional advantages (Economides, 1996; see also Economides and Katsamakas, 2006)

Fifth, by contributing to public goods innovations, the firm may lower the cost of innovation. Chesbrough (2003) argued that involving outside firms, organizations, and individuals in the development of products reduces the direct labor cost in innovation. In addition, when the firm contributes to public goods innovations, such as open source software, it can also effectively reuse existing technology found in the public sphere. Research has shown that, in software development, the reuse of open source software is considerably higher than the reuse of firm-internal software, which should have a positive impact on the cost of innovation (see Haefliger et al., 2008). However, much of this software comes with restrictive open source licenses, which requires the firm to make any combination between this and other software adhere to this license. Hence, while the reuse of such products may reduce costs of innovation, it also "forces" the firm to contribute to public goods innovations.

Sixth, the supply of public goods innovations to the market makes it possible for manufacturers to learn about these innovations and reduce manufacturing-incurred fixed costs related to research and development. Free access to innovations may incentivize manufacturers to ramp up manufacturing capacity, pursue economies of scale, and reduce the price of manufactured products. Additional benefits to the customers of the manufactured product may include enhanced product quality and product warranties (Kotha, 1995; Harhoff et al. 2003).

2.2 Empirical evidence and hidden costs

Empirical research on the private-collective innovation model is mainly found in the field of open source software development where the focus has been on individual contributors and projects (e.g., Shah, 2006; Baldwin and Clark, 2006; Roberts et al. 2006; Gambardella and Hall, 2006). Research on the application of the model by firms is rare with some exceptions: Dahlander (2004) explored the network effects available to firms that provide open source software to the public. Jeppesen and Frederiksen (2006) explored users' motives to contribute voluntarily to the development of media products by firms. Henkel (2006) investigated firms that revealed open source software embedded in their devices to other firms and found incentives for them to do so. Given the focus on public goods innovations in these works, the authors have tended to focus on the cost of forfeiting intellectual property rights in the private-collective innovation model.

Innovation research has pointed to several types of “hidden costs” in the implementation of new innovation models that are not inherent to or captured in the models themselves. For example, Crawford (1992) and Smith (2004) pointed to the costs of implementing the accelerated product development model in US manufacturing industries. Firms often find that the rapid launching of products that have not been properly tested leads to costly recalls, or that fast product development leads to significant delays in pilot- and full-scale manufacturing. Related to open innovation, Kessler et al. (2000) found some indication that the cost of product development rose with increasing dependency on external sources of technology in the innovation process. Empirical research is needed that validates the specific benefits in the implementation of private-collective innovation, examines the costs incurred in such innovation, and identifies firms' strategies to mitigate these costs. We contribute to this research by investigating the implementation of private-collective innovation in the case of the Nokia Internet Tablet development.

3 Research design

The research on the development of Nokia's Internet Tablet focuses on the *process of implementing private-collective innovation*. We investigate benefits in implementation and extend the private-collective innovation model by identifying implementation costs and strategies to mitigate these. Research on implementation processes typically require longitudinal observation (Pettigrew, 1990), prompting a case study design. In order to obtain insights into the development process, we gathered different types of data and performed quantitative and qualitative analysis. Done properly, such combined analysis offers valuable insight, as Shah and Corley (2006) have recently argued. In the following sections, we describe sampling, data sources, and data analysis.

3.1 Sampling

Our research design is a single-case study demanding particular attention to sampling (see Eisenhardt, 1989). There are three reasons for selecting a particular case: fit, distinctiveness, and its revelatory nature (see also Yin, 1999; Siggelkow, 2007). First, Nokia's Internet Tablet development represents a case that both serves to explore and extend the private-collective model. The Internet Tablet was created by private investments by Nokia, building on existing open source products as well as releasing a substantial amount of knowledge (source code).

Second, the distinctiveness of the case is provided through Nokia's unique approach in producing a product based on open source software. While Nokia is not the first manufacturer to create a mobile device based on the open source operating system Linux, its strategy of a conspicuous commitment to open source software and its devotion to building a community of volunteers is unique for a multinational corporation in the consumer electronics industry. In order to determine the distinctiveness of the case, we compared it to a wider sample including Sharp, Motorola, Hewlett-Packard, and Sony.

In the 1990s, Sharp introduced a Personal Digital Assistant called Zaurus which ran on an adapted Linux operating system. Soon after, voluntary developers programmed a variety of applications for the device and started a vivid community of developers (among them one of the interviewees). However, in contrast to Nokia, Sharp did not reveal much of its own source code, scarcely supported the community and its interests, and eventually stopped selling the device in Europe and the United States.

Motorola brought Linux-based cell phones to the market in 2003. The development of their operating system was done by MontaVista, a vendor of embedded Linux software, and TrollTech, the provider of the graphical user interface ‘Qt.’ In contrast to Motorola, Nokia did not rely on a few service providers to implement the software but collaborated with many, mainly small, software firms in an open fashion. By using open source software, Motorola expected to cut costs and speed up software development, since they did not pay per-unit royalties and built application software on the existing open source software (Shankland and Charny, 2003). Similar to Sharp, however, Motorola neither revealed source code beyond legal requirements, nor did the firm provide extensive developer documentation of the software on its devices. Moreover, their Linux appliances granted no administrator access to the user, inhibiting the installation or modification of native applications. Other mobile devices running embedded Linux included HP's iPAQ and the Sony Mylo. However, as we discovered in interviews and in press articles, these companies retained the software's source code and we could find no evidence that these firms attempted to build up a community of outside developers as proposed by the private-collective innovation model (von Hippel and von Krogh, 2003).

Third, given the research gap on the implementation of private-collective innovation, we also searched for a revelatory case. The main criterion for selecting a revelatory case is the researchers' access to a previously inaccessible setting for scientific observation. Establishing ties to Nokia and the developer community surrounding the Internet Tablet, the researchers gained access to a variety of data including documents, interviews, prototypes, and online conversations. Shedding light on the reasons for and effects of this innovation project going open and abstracting these underlying intentions into a model that can be used in future research, as well as raising the attention of practitioners to this mode of product development, motivated the selection of the case.

3.2 Data sources

This study relies on several sources of data. First and most importantly, we conducted semi-structured interviews, allowing participants the opportunity to narrate stories, provide anecdotes, and state opinions. Through an initial reading of the mailing lists, relevant stakeholder groups and data sources in the development project were identified as "Nokia employees," "Nokia-paid contractors," and "independent individuals." Interviews were conducted with participants from all the stakeholders identified. The initial participants were selected from the developer and user mailing lists, and subsequent interviewees were identified through snowball sampling (Heckathorn, 1997). In total, 23 interviews were conducted, 10 with Nokia employees, 5 with contractors, and 8 with unpaid volunteers (see Table 1). In order to protect their privacy, interviewees were anonymized. The interviews lasted on average 75 minutes. The interview guidelines included questions on the firm-community relationship, strategies Nokia used to reveal knowledge and technology to the community, motivation, and other issues (see Appendix for examples of two distinct interview guidelines). The initial interview guidelines were updated and enhanced over time, integrating and building upon the results of interviews already analyzed. All interviews were transcribed verbatim and, using the software Max.QDA, codified using an open coding technique (for a discussion, see also Strauss and Corbin, 1998). This led to the creation of 80 codes, which were subsequently merged and reduced to 12 categories including seven incentive and five cost categories.

f

#	Key	Role	Date	Duration	Contribution, Function
1	N1	Nokia	Nov 15, 2006	89 min	Head Open Source Software Operations at Nokia
2	C1	Contractor	Nov 22, 2006	71 min	Developed Window Manager
3	V1	Volunteer	Dec 6, 2006	55 min	Linux Distribution Release Manager
4	V2	Volunteer	Dec 7, 2006	52 min	Developed Mapping Software
5	V3	Volunteer	Dec 13, 2006	54 min	Developed Music Player
6	C2	Contractor	Dec 14, 2006	89 min	Performance Measurements and more
7	V4	Volunteer	Dec 15, 2006	79 min	Developed Swap Memory Feature
8	N2	Nokia	Dec 15, 2006	58 min	Maemo Product Manager
9	N3	Nokia	Jan 12, 2007	92 min	Software Architecture Team Leader at Nokia
10	N4	Nokia	Feb 19, 2007	83 min	GNOME Desktop Developer at Nokia
11	V5	Volunteer	Feb 20, 2007	85 min	Ported Remote Control Software
12	N5	Nokia	Feb 20, 2007	96 min	GNOME Desktop Developer for Nokia
13	N6	Nokia	Feb 28, 2007	80 min	Multimedia Player Developer for Nokia
14	C3	Contractor	Mar 5, 2007	81 min	GNOME C++ Bindings Developer
15	C4	Contractor	Mar 13, 2007	email	Software Developer at Contracted Firm
16	N7	Nokia	Mar 21, 2007	60 min	GNOME Desktop Developer for Nokia
17	V6	Volunteer	Apr 4, 2007	71 min	Developed Geolocation Software
18	C5	Contractor	Apr 10, 2007	email	CEO of Contracted Software Company
19	N8	Nokia	Apr 11, 2007	69 min	X Windows Developer for Nokia
20	N9	Nokia	Apr 12, 2007	77 min	Maemo Community Manager
21	N10	Nokia	Apr 12, 2007	91 min	Testing Team at Nokia, Volunteer in Browser Project
22	V7	Volunteer	Apr 15, 2007	email	Linux kernel patching for Maemo
23	V8	Volunteer	Apr 23, 2007	66 min	GNOME Foundation Board Member

Table 1: List of interviews including identifier keys, role, date of the interview, duration, and the person's contribution or function in the context of this case study

The second data source consisted of the project's user and developer mailing list. The monthly archives were downloaded from their inception in May 2005 until the end of December 2006. The archival data was examined using the statistical software 'R' in order to indicate the size and activity of the community. The resulting statistics are presented later in the text.

Third, one co-author followed the developer mailing list over the course of several months and observed the project's chat channel on IRC (Internet Relay Chat) which is not publicly archived. Often, informal discussion takes place on IRC giving the researchers a feeling for what "really happens" in the community. Such online participant observation is rather uncommon in the research on open source software development but is deemed both necessary and helpful in understanding the unfolding dynamics of the Internet Tablet development. No formal analysis was done with the data but being immersed in the community helped to interpret mails and understand issues raised in the interviews.

Fourth, secondary sources, such as news reports, blogs of Nokia members and volunteers, and corporate Web sites of Nokia and contracting firms were included in the case study database. For example, these sources provided additional information on the extent to which Nokia sponsors other open source projects. Some Web articles were used to get background information on the Nokia device and potential competitors. We also searched an independent Web forum for discussions and opinions of users of the Internet Tablet.

4 Nokia and the development of the Internet Tablet

In this section, we present a short overview of the history of Nokia and the Internet Tablet development, and we provide a descriptive analysis of Maemo, the community for the Internet Tablet software platform. The purpose of this analysis is to confirm the correctness of the case to examine the implementation of the private-collective model for innovation incentives.

Nokia was originally set up in 1865, producing pulp and paper. It underwent a series of remarkable transformations in its business model. In 1967, it merged with the Finnish Rubber Works Ltd. and the Finnish Cable Works, forming the Nokia Corporation with four major businesses: forestry, rubber, cable, and electronics. A diversified company, with a product portfolio ranging from tires to television sets, it first started producing mobile phones in 1981, manufacturing car phones for the first international cellular mobile phone network. The first hand-portable phone sets were introduced in 1987. During the 1990s, Nokia focused on telecommunications, especially on mobile phones based on the then emerging GSM standard, which had been published by the European Telecommunications Standards Institute (ETSI) in 1990. During this period, the company also divested its other businesses. In 1998, Nokia overtook Motorola as the world's largest mobile phone manufacturer (ICFAI, 2005). R&D operations at Nokia have always been scattered across the world, working in a dispersed, non-hierarchical structure, allegedly to prevent the development of ’tunnel vision.’ In 2003 and 2004, Nokia suffered a decline in market share of its mobile phone business. The company had misinterpreted the market demand for ‘clamshell’ devices and camera phones and had failed to adapt fast enough to these new developments (ICFAI, 2005). However, in 2007, Nokia posted EUR 51.1 bn of net sales and an operating profit of EUR 8.0bn, spending EUR 5.6 bn on research and development.

4.1 Internet Tablet History

According to N1, Nokia started to experiment with incorporating open source products, specifically based on the Linux kernel, into their devices in 2000. At the same time, the company sought to develop a device that would take advantage of the increasing availability of wireless access points and give access to Internet appliances everywhere. N1, the head of the software development team, summarized the vision for this new type of mobile device as follows:

> “*At the same time, totally independently, we had another stream of thought which was this kind of category of Internet Tablets. The big idea behind that was really the same way mobile phones liberated voice. Not only houses or offices have phone numbers, but people have numbers. So you can take the phone wherever you go. We have the same vision that we want to do the same thing with the Internet and Internet use cases. You don’t need to fire up a PC and you don’t need to go to your desk. You have this very light portable device that gives you access to the Internet. Whether that is browsing, email, chat, VoIP.*” (N1)

Inspired by this vision, Nokia designed an overall software architecture of the operating system based on open source components and partly adapted these components themselves and partly contracted developers for specific implementation tasks. In 2002, individuals who were active in architecture-crucial open source projects were approached by Nokia and asked to perform some tasks as contractors for the company (e.g., C1, C2, and C3). They had to sign a Non-Disclosure Agreement (NDA) which prevented the leaking of much information until the Internet Tablet went public.

The prototype device was first publicized on May 25, 2005. At the GNOME User and Developer European Conference on May 31, 2005 Nokia announced that it would give away 500 devices for about a third of the regular sales price to selected software developers and donate the sales to the GNOME foundation, a not-for-profit organization dedicated to supporting the GNOME graphical desktop environment (Nokia, 2005). The sales of the product did not commence until November 2005 when the 770 Internet Tablet was officially released with a price of USD 439. Customers could order the product via a Nokia Internet Tablet dedicated Web site or through the official Nokia Web shop. However, little effort went into promoting the Internet Tablet to the wider public. The product was not available through other distribution channels such as local Nokia mobile phone shops[1].

While the Internet Tablet 770 was still being sold throughout 2006, its successor was announced by Nokia's CEO Olli-Pekka Kallasvuo on January 8th, 2007, at the International Consumer Electronics Association show in Las Vegas. The successor product was added to the N-series of Nokia devices, a popular brand for Nokia's major communication products, and marketed to the 'mainstream' public as the N800. On October 17, 2007, the third

[1] A brief anecdote: in 2006, when one of the co-authors of this paper asked for accessories in such a shop, none of the shopkeepers were even aware of the product's existence.

generation of the Internet Tablet, the N810, was presented in San Francisco at the Web 2.0 Summit. Nokia executive VP and head of the company's multimedia business unit, Anssi Vanjoki, stated *"The N810 is the first of these devices targeted at a 'normal' consumer group, beyond the geeks."* (Martin, 2007)

The hardware of the Internet Tablet differs in one main aspect from other Nokia products: it does not contain mobile phone functionality. It offers a 4.13” display with – given its size – an unusually high resolution of 800x480 pixels which, using a stylus, can be utilized as a touch screen. It connects wireless through Bluetooth (connecting to a mobile phone) or through a common WiFi to the Internet. It is also possible to connect to another PC through the integrated USB port. The device has no hard disk, but flash-based storage is included which can be extended with external flash storage media. Since the N800, a VGA webcam has been integrated and in the N810 a GPS receiver is also included.

4.2 The Maemo Community

The operating system and the software of the Nokia Internet Tablet are based on the Maemo software platform, an effort led by Nokia and announced the same day the device was launched. The intention behind the platform was to provide open source components usually deployed on Linux desktop distributions and to adapt and enhance these for the environment of handheld devices (Maemo.org, 2006). Nokia had downloaded the open source GTK graphical toolkit and other components such as the GStreamer framework for multimedia and modified these to fit the needs of an embedded device with restricted hardware resources. Nokia also added their own software developments and parts from independent software vendors as proprietary software, protected by commercial software licenses and released as binary code only. Figure 1 gives an overview of the software architecture distinguishing between software published under an open source software license, commercial software components published by third-party vendors, and Nokia's proprietary components.

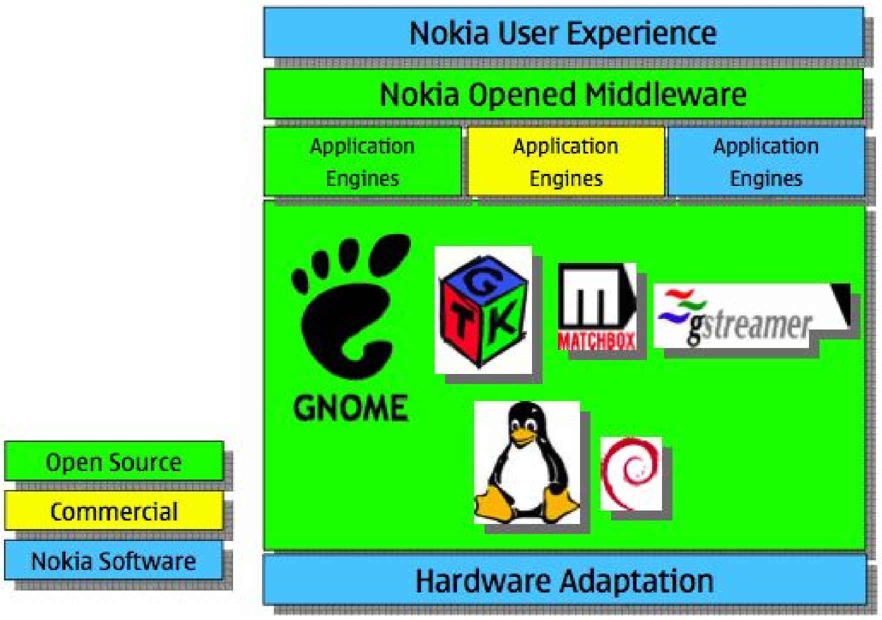

Figure 1: Maemo software stack

The Maemo platform uses its own infrastructure, such as a revision control system, a software bug tracker, and mailing lists that allow communication between developers and users. In November 2006, the Maemo.org site had 54,000 unique visitors. While the operating system contains proprietary software (some hardware drivers and applications such as the Opera Web browser), the main software platform, Maemo, was open source and developed by the Maemo community.

A source code repository allows developers to add new software components to the product and upload improved software that resolves problems and bugs in previous versions. The Maemo source code has its own repository of source code. As of January 2007, our descriptive analysis shows that 33 developers added more than 7.2 million lines of code to this repository, forming the core of the operating system (although much of this is unmodified code from other open source projects). Figure 2 visualizes growth in the source code over time. It should be noted that the developers were exclusively employees of Nokia or formally affiliated with the Maemo project, indicating that Nokia retains tight control over the actual changes that happen to the core system in the software architecture (see also Kuk, 2006).

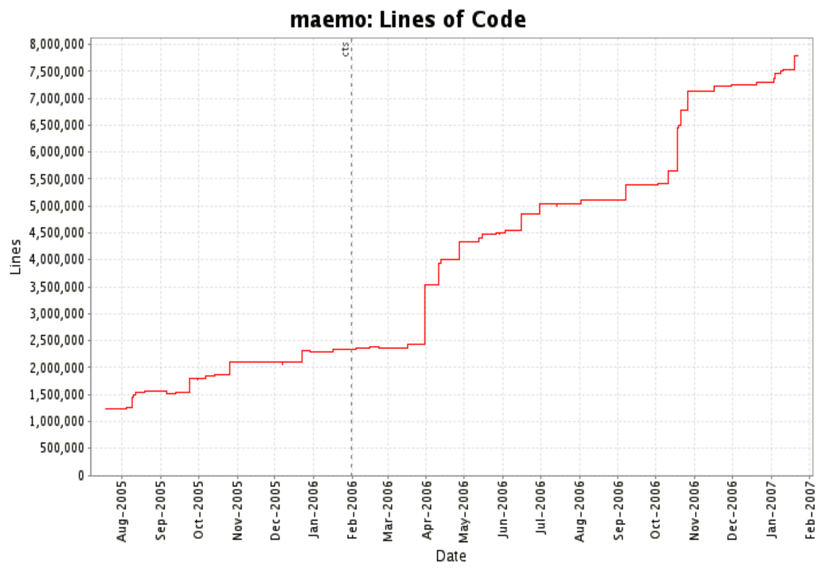

Figure 2: Lines of code in the Maemo Subversion code repository

In order to help identify and remove software bugs, Maemo also has a so-called “bug tracker” of its own. This is used to enter software errors, - or bugs, and keep track of the bug-fixing process. Our analysis shows that in June 2008, this tracker contained 3228 bugs of which 1133 bugs were marked as “open,” meaning that they were waiting to be fixed.

The Maemo project infrastructure also offers a repository, garage.maemo.org, where people independently of Nokia can register their projects and use the developer infrastructure for free. Our analysis shows that in June 2008, Garage consisted of 615 active projects and 12,446 registered users. Projects led and contributed to by independent individuals ranged from GPS navigation solutions and system administration tools to an Electronic Flight Information System for use with small aircrafts.

Most communication happens through mailing lists. For discussion purposes, Maemo has a developer list where actual technical development issues are discussed and a user mailing list for discussing issues related to the operation and use of the product. We examined the Maemo developer mailing list from its inception in June 2005 until December 2006. In total, 832 participants identified by their email address contributed to the list, with 79 participants (9.5%) posting from an official @Nokia.com or @maemo.org email address. The monthly number of postings to the list ranged from 98 to 548 with a median of 328 (mean 339.8). 19.6% of all mails were sent by Nokia addresses. Out of the top fifteen posters, only four had an official email address from Nokia/Maemo. Communication on mailing lists is organized in so-called "threads." Often, a thread is started by someone asking a technical question, followed by replies by others who attempt to discuss and answer the question. 183 threads were started by Nokia affiliates, while 931 mails were replies to an existing thread. Non-Nokia participants, on the other hand, started 1,670 threads and posted 4,010 replies. A chi-square test of the frequency contingency table confirms that Nokia participants differ significantly in the thread start/reply frequencies from non-Nokia participants: Nokia members were more likely to post a reply to an existing thread rather than start a new one.

Besides discussing development, there is also a Maemo user mailing list which serves to support general user discussion. This list follows similar patterns although it is less frequented than the developer mailing list. Its number of postings per month ranged from 3 to 253, with a median of 133 and a mean of 127 mails. As can be expected, user-related discussions started to pick up after the device was introduced to the market in November 2005, whereas development discussions preceded this point. Our analysis shows that 511 participants contributed to the list, 6.46% of them from Nokia. Only two out of the top fifteen posters identified themselves as belonging to Nokia. 10.9% of all mails were sent from Nokia employees. These started 29 new threads and replied to 220 existing threads. Non-Nokia participants, on the other hand, started 775 new threads and sent 1,510 mails to existing ones.

Also on the user mailing list, a chi-square test confirmed that Nokia affiliates were significantly more likely than non-Nokia participants to reply to existing threads rather than start new ones.

Our brief examination of the Maemo community, including source code development and technical and user discussions, supports the statement that the development of the Nokia Internet Tablet is an implementation of private-collective innovation. Large parts of the product are a public goods innovation and Nokia employees as well as external individuals and organizations expend considerable private resources (time, knowledge, and technology) to contribute to the innovation. In the mailing list geared towards development, more than 80% of all emails were sent by non-Nokia affiliates. In the next section, we present the findings from the case study.

5 Findings

The following section presents findings on the implementation of private-collective innovation in the case of the Nokia Internet Tablet development. The aim of this section is twofold: first, we illustrate a case of private-collective innovation with empirical data, providing specific benefits for the company involved. The findings confirm the six conjectures on benefits derived above from the existing literature and additionally identify a benefit in the case: faster time-to-market. In addition, we extend the model of private-collective innovation, highlighting the hidden costs related to the implementation of private-collective innovation and strategies to mitigate these costs. The benefits and costs of this extended model are summarized in Tables 2 and 3.

Benefits	**Findings in the Nokia case**

Low knowledge protection costs	Revealing source code rather than protecting it; however, undetermined costs for revealing.
Learning effects	Collaboration with external firms and individuals
Reputation gain	Increased attraction of Nokia as an employer and for building their own developer community
Adoption of innovation	Standard setting of the platform configuration
Increased innovation at lower costs	Reuse of open source software, outsourcing of software testing and bug fixing and maintenance to open source communities. Experimentation and contributions of new applications by lead users
Lower manufacturing costs	No licensing fees for software platform
NEW: Faster time-to-market	Tapping of distributed technology expertise and high flexibility of software platform

Table 2: Benefits in the implementation of Private-Collective Innovation and findings in the Nokia case

Cost	**Findings in the Nokia case**	**Mitigation strategy**
Difficulty to differentiate	Released source code can be reused by competitors	Partial revealing of source code to retain control of look and feel
Guarding business secrets	Plans for new products	Selective revealing of future plans and protection of information through NDAs
Reducing community entry barriers	Investments for Software Development Kit, preview version of platform, device program, staff for community management, and increased communication effort	Sharing the costs with other actors in the community
Giving up control	Development direction such as scope of functionality of open source projects is controlled by external parties	Hiring of key developers and participation in upstream communities. No single vendor controls platform
Organizational inertia	Required internal restructuring of processes	Adapting and opening up processes

Table 3: Costs and possible mitigation strategies in the implementation of Private-Collective Innovation

5.1 Benefits in implementation

First, considering the **reduced cost of knowledge protection**, product development managers at Nokia were conscious that costs stemming from protecting proprietary knowledge and costs of voluntarily revealed knowledge imply a trade-off. For the Internet

Tablet platform, Nokia decided to keep some components proprietary, while developing other components as open source software. According to the interview with N1, about 25% of the software is unmodified open source software. Another 50% consists of existing open source code to which Nokia made improvements or adaptations and which was again released under an open source license. Only 25% comprises closed code that Nokia either implemented from scratch or which is closed commercial (proprietary) software from a commercial vendor. N1 stated a reason for not releasing source code under an open source license was that Nokia wanted to keep control of the look and behavior of end-user applications, rather than fearing a loss of valuable knowledge. Disclosing the knowledge was not free either, all source code had to be checked by Nokia's department to ensure that Nokia did not reveal intellectual property or patents they did not have the right to. Altogether, it remains unclear whether Nokia saved costs by revealing knowledge rather than spending effort to protect it. Nokia seems to see it as beneficial:

> *"We have evidence that some of our competitors are now looking at our code and they are investigating if they could use our code in their products. You might say that we help them now to get their products out fast. [...]But if we had not put it out then we could not have used the OSS communities who have already helped us to develop that code."* (N1)

Second, Nokia obtained **learning effects** by experimenting with open source software technologies and by collaborating with contractors, firms, and volunteers. The platform had started as a research project within Nokia, seeking to explore new possibilities with open source components in order to build new competences. Although the company could have chosen a commercial open source software integrator such as Red Hat or Montavista, delivering ready-made solutions in order to release a new product quickly, Nokia decided to learn about the novel technologies internally in order to gain the potential for future innovations. N3 reported:

> "*[We wanted to make] something that is more than just putting a product into the market through learning about the possibilities of leveraging open source in deeper and more substantial ways. As a result, we didn't take the quick and simple approach, we went a little bit deeper and we learned more.*"

Building upon technologies developed mainly by external contractors and volunteers implied that many outsiders, rather than the internal research and development staff, had the most intimate knowledge of certain software components. The interview with contractor C2 revealed that in the beginning, Nokia tried to solve technical issues internally. However, when the company faced severe time pressure on a product launch date, it would outsource the request to a trusted external firm which specialized in the area of technical development. For specific parts of the Internet Tablet software, Nokia contracted several small enterprises with experience in open source components (see Table 6). Working with these experts and listening to feedback from the Maemo community helped Nokia to rapidly learn new open source technologies and how those were produced by their communities.

Company	**Expertise**	**Country**
KernelConcepts	GPE and Embedded Linux	Germany
OpenedHand	Matchbox	United Kingdom
Collabora	Telepathy	United Kingdom
Imendio	GNOME and D-BUS	Sweden
Fluendo	GStreamer	Spain
Movial	Scratchbox	Finland

Table 6: Some contractors to Nokia in the software development process

Nokia's strategy was deliberately focused on small companies, rather than on a few, large contractors. Collaborating with small firms all over Europe enabled Nokia to fill gaps in the company-internal technical knowledge needed for development, while still retaining control of the core software architecture by having internal experts in these areas (C2).

In addition to learning about the technologies in use and acquiring the skills to work with them, Nokia learned to cooperate with a diverse community of employees, volunteers, and contractors rather than relying on contractors only who could be forced to keep schedules and timelines:

> *"It's all about the process... You develop this openly within the communities and you try to synchronize your own work with the heartbeat of the communities. Some companies now understand this better than others. We certainly have done our learning. We have made some mistakes too on this front.*" (N1)

For example, one of Nokia's major learnings concerned the early decision to independently continue development of the GIMP Toolkit (GTK), a collection of software constituting the core of the graphical interface. Among others, C3 stated *"the disadvantage of doing this of having a forked or large patch was experienced by Nokia."* By following a separate development stream Nokia became disconnected from the code maintenance effort by the community and, thus, realized they had to move their changes back into the main open source project as architecture team leader, N3, reported:

> *"It was also a learning field that we had to go through. If we had known then what we know today, we would have been able to do it without such a large patch. We would have been able to do more directly upstream in technically better ways with less effort for changing code."*

Third, the growing commitment to open source software development led to a **reputation gain** for Nokia, and interviewees V2, V3, and V4 suggested this led to the increasing attachment of volunteers and recruiting benefits. Particularly the openness of the Maemo platform encouraged volunteer users and developers to buy such a device, to improve the operating system, and to create applications on top of it. Nokia was accredited with pioneering development of an open embedded platform by users who enjoyed running applications from third parties as for example V6 stated:

> *"Yeah, I definitely have a more positive view of Nokia this way. Especially, I think they are handling the open source interaction quite well. I think they are quite a good open source citizen."*

For Nokia, recruiting active open source contributors enabled them to select the best individuals based on their prior contribution to the various projects that constituted the development of the Internet Tablet. In some cases, Nokia directly contacted skilled open source software developers and invited them to apply for jobs. Thus, most of the Nokia employees working on the Internet Tablet were previously active participants in open source projects and known to the Nokia managers through prior collaboration. This fact reflects that for many, Nokia became an interesting company to work for, as C1 testified:

> "*At that time, it was a dream come true. I did not have a good job. I was spending all my spare time hacking Matchbox [window manager of the Internet Tablet]. I was really enjoying it. And then you're given a chance to get paid to do that full time. It was pretty fantastic and an amazing piece of luck.*"

According to interviews with Nokia development managers, Nokia employees selected through the community were highly motivated to continue to work on technologies they already knew. Possibly intrinsic motivation, such as fun - often a primary cause of contributions to open source development - played a role in their continued high-level efforts (see e.g., Luthiger Stoll, 2006; Torvalds and Diamond, 2001).

Fourth, the private-collective innovation model proposes that being first to contribute a public goods innovation increases the likelihood of fast and widespread **adoption of the innovation**. According to the interviews, since the community was already familiar with underlying technologies, adapting existing applications from other projects to the Internet Tablet platform proved a relatively easy task. Nokia also invited competitors to participate in the creation and use of their platform, citing a "the more the merrier approach." By initiating a vendor-independent embedded software platform intended for use in other mobile devices, Nokia made it easier for volunteers, contractors, and competitors to contribute. Spreading the innovation and inviting others to participate was seen as crucial:

> *"We believe the world is changing and the competitive advantage comes from how many others you can get to participate in this network.*" (N1)

In fact, in July 2007, Intel announced they were adopting the Internet Tablet's user interface framework Hildon into their new product category called Mobile Internet Devices (Paul, 2007), which will eventually lead to a higher developer and user basis of Nokia's Maemo platform.

Fifth, by contributing to public goods innovations firms can **lower the cost of innovation**. Building on existing and mature technologies which could be integrated into the new hardware, Nokia enabled the development of a solid, yet cheap operating system for embedded platforms. Collaborating within existing open source projects allowed Nokia to benefit from the collective programming efforts:

> *"So what is happing in the D-BUS, in the GTK, in the GStreamer, in the Linux kernel is that I put two guys there, IBM puts two guys, Motorola maybe puts one*

guy, maybe Novell puts a couple of guys. So for the price of two guys, I get four or six guys working on the same problem." (N1)

Additionally, the company benefited from voluntary contributions leading to enhancements of the device. The volunteers contributed several innovations including applications, user interface improvements, translations, bug reports and fixes, testing different peripherals, and making feature requests. Letting volunteers experiment with the software also created "proof of concepts" which enabled innovations previously seen as unrealistic by Nokia engineers (V3). One example of this is the swap memory enhancement (using the flash memory as extended virtual memory) which was initiated by volunteer V4 and included in a subsequent official version of the Internet Tablet's operating system. Thus, the likelihood of finding a "killer application" (see e.g., Downes and Mui, 1998) in the process (by evolution or sheer luck) increased. An open source software developer observed:

"*I think from my point, if you let people change things [...] and document them and open them up so people can hack their own stuff, you never know what is going to happen, what kind of things people are going to write for your device which ultimately could make it sell millions and millions if someone writes the killer application for it.*" (C1)

As such, both the costs of innovation were kept low and ideas which would not have been developed otherwise could be tested and integrated:

"*I can develop, say, twenty ideas a day and this community can develop a hundred ideas a day. So it's more important to be part of the community with a hundred ideas than by yourself with twenty ideas.*" (N1)

Investigating the software architecture, we found most contributions by volunteers were separate applications which could be installed independently from the core operating system. The interviews showed that volunteers who made such contributions showed high commitment to and responsibility for their work, listening to user feedback and, in some cases, when others demanded it, even enhancing their software against their personal belief of the usefulness of the features (explained by V2). Through their contributions and feedback from users, volunteers slowly gravitated towards more development work in the community.

Sixth, supply by anyone of public goods innovations to the market enables manufacturers to learn about innovations and, thereby, **reduce costs in manufacturing**. This conjecture indicates particular benefits private-collective innovation in software offers to computer hardware manufacturers (von Hippel and von Krogh, 2003). Through choosing a software platform which is available under an open source license as indicated, Nokia reduced fixed costs related to research and product development. Nokia manufactured and sold the Internet Tablet, but the product's functionality and, thus, ability to fulfill user needs were to a large extent shaped by the users themselves. User-developed applications, such as mapping and navigation software, could easily be installed by the end users themselves for free, keeping Nokia's costs down. Interestingly, in addition to fixed cost reduction, Maemo also has a positive impact on variable costs in manufacturing since Nokia did not have to pay a per-device license fee to an intellectual property owner. For example, at the beginning of 2006, a comparable proprietary operating system, Symbian, demanded USD 7.5 per device for the first 2 million units.

In addition to confirming theoretical conjectures on benefits, one more benefit emerged in the study that should be considered crucial for private-collective innovation: **faster time-to-market**. Using external, modular technologies not only impacted on costs, it also led to the creation of a new operating system working on a new hardware platform in a short time. According to interviews with N1 and N3, this fast development created flexibility which, combined with user feedback of pre-releases, allowed for a quick time-to-market compared to other devices the company had launched.

However, integration of the numerous open source components into a executable environment is very challenging as, for example, C3 explained. One way to rapidly tap into this knowledge of open source communities was achieved by Nokia's strategy of contracting open source developers and small firms – the "bridges between Nokia and the communities" (N3):

> *"It'll get done quicker and probably better if they pay us to do it. [...] Over the years we have been involved, we have so much experience and knowledge, we know all the tricks. We know how to get things like X and Matchbox up and running quickly on hardware. We are just basically selling that knowledge as well. Although they could very well likely figure it out themselves, we just can get them there a lot quicker."* (C1)

5.2 Costs of implementation and strategies to mitigate these

While the benefits of private-collective innovation have been spelled out previously, the hidden costs of implementing the model have been neglected in previous work or remain unknown. In this section, we present the findings from the case study along five categories of costs together with Nokia's strategies (where applicable) to mitigate these. First, when software is freely available even to direct competitors, it is possible for current and future competitors to design clones that look like and behave in a very similar way to the original product. Competitors are in the position not only to imitate but to replicate the product (see

Kogut and Zander, 1992). This potential **lack of differentiation** of products represents a cost to the firm as it forfeits an opportunity to gain competitive advantage (Granstrand, 1999). Such reuse of open source software is completely legal and, in fact, took place as explained in this illustrative example:

> *"The OpenMoko actually bases on our stuff. [...] the Moko window is a kind of base window for their applications I think. There in the comments of the source file you can find that this is based on the Hildon window by Nokia. They even have this copyright "Nokia corporation." So they are based on our stuff. But, on the other hand, our stuff is not that revolutionary. I mean that's how it goes. We base on somebody else's stuff too."* (N4)

Nokia's strategy to mitigate this cost was to selectively open up its software development. Figure 1 (see above) visualized how Nokia revealed the middle layer of software, the basic infrastructure, under an open source license, while keeping parts of the bottom layer (hardware specific software) and much of the user-visible applications under their own proprietary license. N1 argued that it wanted to ensure a unique "Nokia look" by retaining control over applications, for example of the device's email program. In doing so, they kept crucial parts closed (e.g., power management and other hardware drivers) and prevented replication of these parts by competitors. In all the interviews conducted with volunteers, the respondents demonstrated understanding for Nokia's decision to selectively reveal source code, although some stated their strong preference for releasing all software under an open source license.

Second, Nokia was concerned with potential costs stemming from **losing business secrets**, such as plans for future devices. In order to mitigate these costs, Nokia revealed knowledge in terms of software but kept future key product innovations and business figures, such as devices sold, number of employees, or investments expended, confidential. For the development of new software features, they contracted several small enterprises or even motivated individual volunteers with unique knowledge critical to the development of the Internet Tablet to sign a NDA. The NDA protected Nokia against the leakage of information about planned new product developments. The agreements effectively created a three-tiered community of people "who knew," which meant Nokia insiders, "those with a clue," which meant contractors, and the "regulars" (C3). However, this information imbalance led to strong tensions for Nokia with its external community members. Thus, Nokia hired N9 e.g., in order to update the technical roadmap of the Internet Tablet with all the information necessary for software developers.

Third, in order to facilitate the increasing involvement of volunteers, Nokia needed to carry the costs of **reducing community entry barriers**. The company invested in the creation of a Software Development Kit that enabled new volunteers to easily start development for the Internet Tablets. It is common practice for software manufacturers to provide such an SDK at high costs (Jacobson et al., 1999). However in the case of the Internet Tablet, Nokia offered the development tools for free. In order to allow volunteers to adapt their software for upcoming platform releases, Nokia also offered a development snapshot of their work-in-progress (often including software for yet unannounced features) which could be used to ensure that an application would also run on future releases. Employees were sent to related conferences in order to increase awareness of the platform and answer questions from current and future volunteers and contractors. Additional staff, such as "community representative" N9, were hired in order to communicate between Nokia internal developers and the external community members. In order to mobilize more volunteers to join the Maemo community, Nokia sold 1,500 heavily subsidized devices to active open source developers. While such direct costs by Nokia cannot be mitigated easily, investments in community building, knowledge diffusion, and marketing may be lower in the future through sharing the effort with other community members. Some interviewees explained that, for example, with increasing popularity, well-integrated Maemo community members started to support new volunteers who were getting involved in the development process.

Fourth, by contributing source code to open source projects which were not managed by the company, Nokia **gave up control** of the future development direction of core technologies deployed in the company's hardware. According to N1, the company traded having full control of the technology for participation in joint development, thus benefiting from sharing the cost of innovation with outsiders. For example, GTK was originally intended for use on desktop PCs, and according to interviewees N1, C1, V2, V4, and N3, it was necessary to adapt GTK to the low resource environment of the Nokia Internet Tablets by decreasing memory consumption. In order to regain some control of these critical software components, Nokia hired key developers from the GTK community and contracted small enterprises with deep knowledge in this area. According to N3, the contribution of code improvements and modifications, combined with a meritocratic organization of the projects involved, gave Nokia enough influence on the direction of software development. Yet, the practice of contracting developers from incumbent communities such as GNOME in order to gain reputation and control raised concerns from some Maemo community members as to how Nokia would influence the future of the projects. For example:

> "*Obviously when they are sponsoring a project, then they are going to have some control over the direction and what gets into it. [...] I think, they truly want to work with the community and want to keep them happy as well, so it's all a bit of give*

and take, I suppose." (C1)

The situation was different for projects that were initiated and controlled by Nokia itself. According to C3, Nokia granted only write-access to its software repository to its employees, thus retaining control of the actual published source code. It is worth noting that this strategy created some tensions in the community, with the non-Nokia members complaining that they could not help or contribute if they did not know about the future direction of the software development or about the estimated timeframes of future software releases. The Maemo community tried to influence Nokia's behavior, mostly through providing intense feedback on mailing lists. For instance, in 2000 Nokia released a new version of the Maemo platform as a binary download as V4 remembers. However, Nokia did not publish the source code at the same time but staved off the community by arguing that the legal department needed three more weeks to clear the code for reasons of intellectual property protection. A strong, negative reaction from the community taught Nokia to proceed differently next time, releasing both binary and source code simultaneously. Sanctions on the part of the community were mainly "withdrawal of love" and the "threat of forking.[2]" When attempting to balance control and openness, Nokia considered the threat of defection of volunteers to other open platforms used by emerging competing "open devices."

Fifth, since the private-collective model of innovation incentives breaks with the traditional private-investment model that is prevalent in industry, it is reasonable to expect that the implementation of the model in an established firm incurs costs of **organizational inertia** (see Sorenson and Stuart, 2000). Since the Internet Tablets also include software written by third-party vendors, Nokia needed to ensure that their software revealed to the

[2] A so-called "fork" results when "dissatisfied programmers" copy the original source code from a project and continue its development in an alternative, competing project.

Maemo community did not infringe on intellectual property rights. As mentioned above, interviewees commented that Nokia's internal approval of open source software was slow and bureaucratic. In addition, the complex internal processes of a large multinational organization often made it difficult for Nokia's developers to collaborate with external open source projects. One interviewee commented:

> "*The biggest problem is that Nokia is a very big company and that Maemo is a very small group. I think it's like ten or twenty people in total working on it. A lot of software which they use on the 770 is developed in some other groups in Nokia. For example, the movie plug-ins and MP3 plug-ins. Those are the same I think as they use on their phones in Symbian. So that's a Symbian decision. You can't tell the Symbian people to use that bug system instead of the internal bug system. They would say, 'Why?'* " (V1)

Nokia employees also commented that the internal Nokia firewall would not allow connection to the official developer chat room of the Maemo community, which was located outside Nokia's network. In order to mitigate the costs of organizational inertia, Nokia employees stated that they would work on a case-by-case basis to remove obstacles. Employees were, for example, assigned to make sure that entries in the external, publicly accessible bug tracking system would be paid attention to.

Altogether, Nokia managers were conscious of the trade-offs between revealing knowledge and technology and the benefits from participating in private-collective innovation. The following statement summarizes well the experiences had regarding the trade-offs between cost and benefits in the model:

> "*Some people might say that one of the problems is that you are leaking and giving out your secrets and so forth, but it's more like a trade-off. What is more important*

to you: to give some of your secrets an internal work-out or how much help in creating these products you get for free. I think, if you calculate, you are far more on the positive side when you decide to share." (N1)

6 Discussion and conclusion

In this paper, we identified a research gap in the literature on private-collective innovation (von Hippel and von Krogh, 2003; 2006): little is known about the implementation by firms of the private-collective model of innovation incentives. We argued that the implementation of the model will be associated with benefits, "hidden" costs, and strategies to mitigate these. In order to examine and extend the model through empirical work, we employed a case study design. Using quantitative and qualitative data, we demonstrated that the development of the Internet Tablet is a case of private-collective innovation. Next, we analyzed data from several sources in order to identify benefits and costs incurred in the implementation of private-collective innovation and strategies by Nokia to mitigate these costs.

Nokia launched the Internet Tablet as a private-collective innovation project and as a low-cost probe (Brown and Eisenhardt, 1997). At the time of product launch, neither a product category nor a market for these devices existed. Rather than following existing market demand, Nokia targeted technology pioneers to find out who would use the Internet Tablet and how it would be used in real-life applications (similar to what Zander and Zander, 2005, called "exploiting the inside track'). Nokia opened up the product's software using externally developed open source technologies, allowed for and encouraged contributions by outsiders and, in the process, created a new market for a product it had envisioned. When the product proved successful, Nokia moved from targeting technology pioneers towards the mainstream market with the subsequent release of the Internet Tablet N800 and N810.

This study confirmed most incentives to innovate identified in previous literature (von Hippel and von Krogh, 2003; 2006). It remains inconclusive whether Nokia saved knowledge protection costs by revealing most of their software. However, the company gained skills and knowledge through collaborating with outside volunteers and contractors and, thus, also acted as a system integrator coordinating a loosely coupled network of component providers (see also Brusoni et al., 2001). As the main contributor in the project, Nokia enjoyed reputation benefits both as an attractive employer and as an "open-source friendly company" amongst open source software developers who contributed software to the platform. By creating the vendor-independent GNOME embedded platform and inviting competitors to contribute and use the software platform, Nokia facilitated the adoption of the software as a common platform for embedded devices. Moreover, Nokia built upon existing technology and took advantage of users' previous contributions to open source software projects. By taking advantage of existing open source components, Nokia managed to create a complete operating system with only a handful of developers and was able to integrate ideas and improvements from other Maemo community members. In terms of costs, Nokia's manufacturing could benefit from low cost software development and avoided paying the common per-device license fees. Finally, our study found that increased flexibility and a faster time-to-market is a benefit in implementing private-collective innovation.

Thus, although the six benefits at first glance make it rational for the firm to choose private-collective innovation amongst alternative models, previous work also raised the awareness of unintended consequences or "hidden costs" resulting from implementing private-collective innovation. For example, in order to obtain outside contributions, a firm may need substantial investments in documenting the released software, training potential contributors, and developing online tutorials. These costs of implementation may offset the benefits to private-collective innovation[3].

We briefly reviewed literature that indicated "hidden costs" associated with the implementation of novel innovation models (Crawford, 1992; Kessler et al. 2000; Smith, 2004). The study found that the implementation of the private-collective model of innovation incentives in Nokia's development of the Internet Tablet incurred costs and that the company found strategies to mitigate these. In particular, the potential lack of product differentiation as well as revealed business secrets incurred costs to the company. Nokia mitigated these costs by selectively revealing knowledge and technology. Another cost concerns the lowering of entry barriers to the Maemo community. The company invested in several measures to reduce such barriers, including discounted devices and a free SDK, as well as allocating employees responsible for community communication in order to attract further volunteers. Moreover, using technologies that are partly maintained externally has the advantage of shared innovation costs but implies giving up full control of the future development of that technology. Through hiring key developers of software for the product, Nokia regained some influence and control. Internal processes sometimes proved inadequate to enable a transparent and open development process, incurring some delays and costs as well as frustration in the community. Nokia acknowledged this challenge that the interviewees described as a "learning process."

The extended model of private-collective innovation provides additional insights for researchers. First, implementing private-collective innovation may enhance organizational learning and renewal, in addition to being a form of "open product development" (Chesbrough, 2003). During the development of the Internet Tablets, Nokia adapted and

[3] In addition, Osterloh and Rota (2004) pointed out that the mere presence of a firm in private-collective innovation may "crowd out" intrinsic motivation by voluntary contributors (e.g., fun and enjoyment).

learned to work with a community of volunteers. An open research question is to what extent such learning can enable firms to work with outside volunteers across generations and categories of products. Second, Harhoff et al. (2003), modeling the payoff for innovators to freely reveal their innovations find that one of four conditions, “greater generality of the knowledge,” reduces the likelihood of free-revealing by the innovator (see also Muller and Pénin, 2006). However, this case study showed that much knowledge revealed proved to be generic frameworks: it laid the foundation for a generic embedded Linux desktop environment. More specific components, such as power management and some end-user applications were kept proprietary. This contradicts Harhoff's et al. findings and the issue of what kind of knowledge is likely to be revealed under what conditions requires more attention in future research.

Limited by its design, the current study can only generalize findings to theory. The extended private-collective innovation model provides a set of benefits, costs, and mitigation strategies that must be tested on a larger sample in future research using cross-sectional as well as longitudinal designs. Thus, it will be important to garner insights on technological, industry, and market conditions that provide different levels of benefits and costs of innovation. For example, in industries of non-virtual goods or where product development constitutes a minor share of fixed costs in production (e.g., cement manufacturing), companies may find it more attractive to pursue private-investment innovation. Moreover, if innovation is largely based on tacit knowledge acquired through extensive and costly apprenticeship (e.g., luxury goods), volunteers who join product development products may be rare. Future research will also have to investigate the impact of company age and size on the innovation incentives in the private-collective model. As we found in the case of an established company, the process represented costs of organizational inertia.

Managers who want to experiment with flexible solutions, while keeping their own product development costs low, should investigate ways to implement private-collective innovation. Sharing development costs and enabling contributions from third parties, as well as boosting organizational learning, are powerful reasons why the model is attractive in practice. However, there are potential “hidden costs” in implementing the model. Learning from Nokia's successful approach, managers should think ahead about possible costs and create strategies to mitigate them. The experience from the development of the Internet Tablet provides possible mitigation strategies.

7 References

Alavi, M. & Leidner, D. E. (1999), 'Knowledge Management Systems: Issues, Challenges, And Benefits', *Communications of the AIS* **1**(1), 1-37.

Allen, R. (1983), 'Collective Invention', *Journal of Economic Behavior and Organization* **4**(1), 1-24.

Arrow, K. J. (1962), 'The Economic Implications of Learning by Doing', *The Review of Economic Studies* **29**(3), 155-173.

Baldwin, C. Y. & Clark, K. B. (2006), 'The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?', *Management Science* **52**(7), 1116-1127.

Brown, S. L. & Eisenhardt, K. M. (1997), 'The Art of Continuous Change: Linking Complexity Theory and Time-paced Evolution in Relentlessly Shifting Organizations', *Administrative Science Quarterly* **42**, 1-34.

Brusoni, S.; Prencipe, A. & Pavitt, K. (2001), 'Knowledge Specialization, Organizational Coupling, And The Boundaries Of The Firm: Why Do Firms Know More Than They Make?', *Administrative Science Quarterly* **46**(4), 597-621.

Chesbrough, H. (2003), *Open innovation. The new imperative for creating and profiting from technology*, Harvard Business School Press.

Crawford, C. (1992), 'The Hidden Costs of Accelerated Product Development', *Journal of Product Innovation Management* **9**(3), 188-199.

Dahlander, L. (2004), 'Appropriating the Commons: Firms in Open Source Software', Technical report, Chalmers University of Technology.

Dam, K. (1995), 'Some economic considerations in the intellectual property protection of software', *Journal of Legal Studies* **24**(2), 321-377.

David, P. (1998), 'Knowledge spillovers, technology transfers, and the economic rationale for public support of exploratory research in science''Background Paper for the European Committee for Future Accelerators'.

David, P. (1992), 'Knowledge, property, and the system dynamics of technological change', *Proceedings of the World Bank Annual Conference on Development Economics*, 215-247.

Downes, L. & Mui, C. (1998), *Unleashing the Killer App: Digital Strategies for Market Dominance*, Harvard Business School Press.

Economides, N. (1996), 'The Economics Of Networks', *International Journal of Industrial Organization* **14**(6), 673-699.

Economides, N. & Katsamakas, E. (2006), 'Two-Sided Competition Of Proprietary Vs. Open Source Technology Platforms And The Implications For The Software Industry', *Management Science* **52**(7), 1057-1071.

Edwards, K. (2003), 'Epistemic Communities, Situated Learning And Open Source Software Development', Technical report, Technical University of Denmark.

Eisenhardt, K. M. (1989), 'Building Theories From Case Study Research', *Academy of Management Review* **14**(4), 532-550.

Foray, D. (2004), *Economics Of Knowledge*, MIT Press.

Gambardella, A. & Hall, B. H. (2006), 'Proprietary Versus Public Domain Licensing Of Software And Research Products', *Research Policy* **35**(6), 875-892.

Grand, S.; Georg von Krogh, D. L. & Swap, W. (2004), 'Resource Allocation Beyond Firm Boundaries: A Multi-Level Model for Open Source Innovation', *Long Range Planning* **37**, 591-610.

Granstrand, O. (1999), *The Economics and Management of Intellectual Property*, Edward Elgar, Cheltham.

Gächter, S.; von Krogh, G. & Haefliger, S. (2006), 'Private-Collective Innovation and the Fragility of Knowledge Sharing'.

Haefliger, S.; von Krogh, G. & Spaeth, S. (2008), 'Code Reuse in Open Source Software', *Management Science* **54**(1), 180-193.

Harhoff, D.; Henkel, J. & von Hippel, E. (2003), 'Profiting From Voluntary Information Spillovers: How Users Benefit By Freely Revealing Their Innovations', *Research Policy* **32**, 1753-1769.

Heckathorn, D. D. (1997), 'Respondent-Driven Sampling: A New Approach to the Study of Hidden Populations', *Social Problems* **44**, 174-199.

Henkel, J. (2006), 'Selective Revealing In Open Innovation Processes: The Case Of Embedded Linux', *Research Policy* **35**(7), 953-969.

von Hippel, E. & von Krogh, G. (2006), 'Free revealing and the private-collective model for innovation incentives', *R&D Management* **36**(3), 295-306.

von Hippel, E. & von Krogh, G. (2003), 'Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science', *Organization Science* **14**(2), 209-223.

ICFAI (2005), 'Nokia and the Global Phone Industry', Case study by ICFAI Center for Management Research, Hyderabad, India.

Jacobson, I.; Booch, G. & Rumbaugh, J. (1999), *The Unified Software Development Process*, Addison-Wesley, Boston.

Jeppesen, L. B. & Frederiksen, L. (2006), 'Why Do Users Contribute to Firm-Hosted User Communities? The Case of Computer-Controlled Music Instruments', *Organization Science* **17**(1), 45-63.

Kessler, E.; Bierly, P. & Gopalakrishnan, S. (2000), 'Internal vs. External Learning In New Product Development: Effects On Speed, Cost, And Competitive Advantage', *R&D Management* **30**(3), 213-224.

Kogut, B. & Zander, U. (1992), 'Knowledge of the Firm, Combinative Capabilities, and the Replication of Technology', *Organization Science* **3**, 383-397.

Kotha, S. (1995), 'Mass Customization: Implementing the Emerging Paradigm for Competitive Advantage', *Strategic Management Journal* **16**, 21-42.

Kuk, G. (2006), 'Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List', *Management Science* **52**(7), 1031-1042.

Lakhani, K.; Wolf, B.; Bates, J. & DiBona, C. (2002), 'The Boston Consulting Group hacker survey', www.bcg.com/opensource/BCGHackerSurveyOSCON24July02v073.pdf.

Lerner, J. & Tirole, J. (2002), 'Some Simple Economics of Open Source', *Journal of Industrial Economics* **50**(2), 197-234.

Liebeskind, J. P. (1996), 'Knowledge, strategy, and the theory of the firm', *Strategic Management Journal, Special Issue: Knowledge and the Firm* **17**, 93-107.

Luthiger Stoll, B. (2006), 'Spass und Software-Entwicklung: Zur Motivation von Open-Source-Programmierern', PhD thesis, University of Zurich.

Maemo.org (2006), 'Maemo website', http://maemo.org, accessed Dec 2006.

Martin, R. (2007), 'Nokia Launches 'Context-Aware' Internet Tablet', *InformationWeek*.

Muller, P. & Pénin, J. (2006), 'Why Do Firms Disclose Knowledge And How Does It Matter?', *Journal of Evolutionary Economics* **16**(1-2), 85-108.

Nokia (2005), 'Nokia Makes Donation to GNOME Foundation', Nokia press release.

Nuvolari, A. (2004), 'Collective invention during the British industrial revolution: the case of the Cornish pumping engine', *Cambridge Journal of Economics* **28**(3), 99-119.

Osterloh, M. & Rota, S. (2004), Trust and Community in Open Source Software Production'Trust and Community on the Internet', Baumann, M./Matzat, U., .

Paul, R. (2007), 'Intel launches site for open source mobile Linux development', *Ars Technica* .

Pettigrew, A. (1990), 'Longitudinal Field Research On Change: Theory And Practice', *Organization Science* **1**(3), 267-292.

PricewaterhouseCoopers (2005), 'Governance, risk, and compliance series: Global Best Practices', www.globalbestpractices.com.

Roberts, J.; Hann, H. & Slaughter, S. (2006), 'Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the Apache projects', *Management Science* **52**(7), 984-999.

Shah, S. (2006), 'Motivation, Governance, And The Viability Of Hybrid Forms In Open Source Software Development', *Management Science* **52**(7), 1000-1014.

Shah, S. K. & Corley, K. G. (2006), 'Building Better Theory By Bridging The Quantitative–Qualitative Divide', *Journal of Management Studies* **43**(8), 1821-1835.

Shankland, S. & Charny, B. (2003), 'Linux to power most Motorola phones', accessed Jan 24th, 2007, accessed January 2007 from http://news.com.com/2100-1001-984424.html.

Siggelkow, N. (2007), 'Persuasion With Case Studies', *Academy of Management Journal* **50**(1), 20-24.

Smith, P. (2004), *Accelerated product development: Techniques and traps*, Wiley, New Jersey.

Sorenson, O. & Stuart, T. (2000), 'Aging, Obsolesence, and Organizational Innovation', *Administrative Science Quarterly* **45**(1), 81-112.

Stiglitz, J. E. (2006), 'Scrooge and intellectual property rights', *British Medical Journal* **333**(7582), 1279-1280.

Strauss, A. L. & Corbin, J. (1998), *Basics of qualitative research: Techniques and procedures for developing grounded theory*, Sage, Thousand Oaks, CA.

Torvalds, L. & Diamond, D. (2001), *Just for Fun*, Texere, London, UK.

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

Zander, I. & Zander, U. (2005), 'The Inside Track: On the Important (But Neglected) Role of Customers in the Resource-Based View of Strategy and Firm Growth', *Journal of Management Studies* **42**(8), 1519-1548.

8 Appendix

8.1 Gaining Insight from Interview Codings

As described in the method section, we conducted interviews with a continuously updated, semi-structured questionnaire investigating different aspects of our research question. After transcription of the interviews, we categorized the quotes in similar statements, eventually leading to 80 different codes and 1,026 text codings. Iterative regrouping and recoding of the statements eventually led to three main themes with a total of 12 patterns. Based on these, we created our extended model of the implementation of private-collective innovation.

Since space restrictions prevent us from quoting too many statements within the article, we explain in the following sections how we extracted a general finding from analyzing the interview statements. For example, in the case of the subsequently merged category “Network Collaboration,” the screenshot of the text analysis software Max.QDA shows that the pattern consisted of three additional codes: “Maintenance/Upstream Project,” “Nokia collaborating with Third-Party Vendors,” and “How Nokia Manages OSS Development.” In sum, 38 codings from the interviews covered statements on "Network Collaboration." Triangulating these statements and interpreting them eventually led to the case study findings in Section 4.

Excerpt of interview with N1:

[...] Suppose that I am a company X and I want to build a Linux-based product. Now I go to the community and I take a snapshot of the Linux operating system today. I think a copy of the version 2.6.18. Then I go into my lab. I work two years with that, doing all kinds of fancy things. And then I again come back to the community and say I took a snapshot two years ago, look what I have done. And the community says we don't care. You should have been participating in the community work all the time so we could have seen what you do to integrate our great things with what you have done. Now we have got two totally incompatible branches. Your branch may be good, but there is our branch and that is certainly good. Do you really want to merge these things? So, corporations didn't really understand and still not all of them do, that OSS is not only about software and what kind of software components you have, but it's all about the process that you develop this openly within the communities and you try to synchronize your own work with the heartbeat of the communities. Some companies now understand better than others. We certainly have done our learning. We have made some mistakes too on this front.

What kind of mistakes?

For example, we did exactly what I described with our email client. We took a suite for -mail from OSS and took a snapshot of that and continued developing that over a year. We didn't synchronize it with the open source project. We are now totally in our own branch and the maintenance costs are increasing. We have two options: we either make a huge effort in merging these two branches or then we just forget what we have done and jump back to where the community is today.

Quotes leading to our abstracted finding, as written in the text:

"Interviewees N1 and N3 pointed out that Nokia learned over time that this is indeed the case and that it was inefficient in the long run to maintain their own software version while the upstream project is continuously improved by a much larger group of developers."

8.2 Interview guideline for a volunteer member of the Maemo community

About the interviewee and his involvement

1.What is your education?
2.What and for whom do you work at the moment?
3.Since when have you been involved in open source software (OSS) development and what did you do?
4.In what way is your paid work related to OSS?
5.Do you have commited access to OSS projects? If yes, which ones?
6.What mailing lists are you subscribed to and follow most of the discussion?
7.When and how did you get involved in the Nokia Internet Tablet project? Why?
8.What are your contributions within the Maemo project?
9.How many hours per week do you spend contributing to Maemo? Has this changed over time?
10.Have you received financial or other material benefits from Nokia for your participation in the Maemo community?
11.What is the main reason for your contributions within the Maemo community?

Knowledge Revealing Strategy

1.In your opinion: Why did Nokia not just integrate OSS components but create a community portal and actively

participate in established OSS projects?
2.How would external participation have evolved if Nokia had not published their own software developments as OSS?
3.Would you have participated in the Maemo community if Nokia had used a proprietary operating system?
4.From your perception: What type of software does Nokia release as OSS and what as proprietary binaries?

Knowledge: Knowledge Reuse
1.What do you think, why did Nokia not chose a "commercial" Linux distribution such as RedHat or SUSE but Debian?
2.Expert's estimation (in %): How many lines of code of the software in the N800 (out of the box) is
    1.unmodified, preexisting OSS (e.g. parts of GTK)
    2.modified, preexisting OSS by Nokia or contractors (e.g., parts of GTK)
    3.newly created OSS by Nokia or contractors (e.g., Hildon)
    4.newly created proprietary software by Nokia or contractors (e.g., Canola)
    5.unmodified or modified preexisting proprietary software by Nokia or contractors (e.g., Opera)
3.What tasks are usually required in order to integrate preexisting OSS for the Internet Tablet?

Knowledge: Distributed Technology Expertise
1.Why is it not difficult for external programmers to develop software for Maemo?
2.Why isn't more software written for the platform?
3.What kind of questions do Nokia developers mostly ask?
4.In your opinion, what technical knowledge do you possess which Nokia employees don't?
5.What are the differences in Maemo community participation (mailing list, IRC...) if somebody is employed by Nokia, working for one of Nokia's contractors or if they are a voluntary contributor...
    1....regarding contributions?
    2....regarding technical knowledge?
    3....regarding form of communication?
    4....regarding helpfulness?
    5....other particular issues?

Knowledge: Guarding Business Secrets
1.Why do external developers demand more knowledge on plans of future Maemo developments?
2.What kind of information could Nokia publish in a technical roadmap while guarding business secrets?

Organization: Network Collaboration
1.Do you as a voluntary contributor feel part of the Maemo community? In what way or why not?
2.What are the benefits for Nokia when their software developments are integrated into established upstream OSS projects such as Linux Kernel, GTK, GStreamer, D-BUS?
3.In which OSS projects is Nokia's software code successfully "integrated upstream"? Why?
4.In which OSS projects is Nokia not able to integrate their developments? Why?
5.Are there certain examples how Nokia learned about the importance of this? If yes, which ones?
6.How does Nokia gain trust in these projects?

Organization: Reducing Network Entry Barriers
1.What is your benefit in participating in the development?
2.How does Nokia encourage external programmers to voluntarily contribute to the Internet Tablet software?
3.What should they improve so you'd spend more time programming for Maemo?
4.How relevant are voluntary contributions so far for the Maemo platform?
5.In which communities does Nokia still need to improve its acceptance?

Organization: Recruiting benefits
1.What is your image of Nokia?
2.Would you accept a job offer by Nokia if you could work at the Maemo project?

Organization: Balancing Control
1.How can Nokia maintain control over source code they integrated in incumbent OSS projects?
2.What type of control is important and which one can be neglected?
3.How does Nokia influence future developments in established OSS projects?

4.What made volunteer developers leave the Maemo community?
5.What are other similar OSS projects volunteer developers could be attracted to?
6.Is it probable that certain components of the Maemo software will get forked by the community or by other software or hardware vendors? Why? Why not?
7.How transparent is the development process of the Maemo platform?
8.Would you like to have more influence in the development process of the Maemo platform?

Organization: Organizational Inertia
1.What organizational aspects make it difficult for Nokia to participate in OSS communities?
2.How is collaboration with volunteers affected by Nokia's organizational structure?

Organization: Organizational Learning
1.From your perspective, what have the key learnings been since the release of the Nokia 770 Internet Tablet?
2.What organizational improvements should Nokia make in the future?

Competitive Situation: Time-to-market
1.Did the volunteer community help to speed up the development process of the Maemo platform?

Competitive Situation: Countering Uncertainty with Flexibility
1.How flexible is the hardware/software platform to implement completely new use cases for the Internet Tablet?
2.How does uncertainty of the future inhibit innovation by Nokia resp. by users?

Competitive Situation: User Contributions
1.What are key contributions from the voluntary developer community?
2.How well are the users aligned with Nokia's plans for the Internet Tablet?
3.Could volunteers develop a killer application for the Tablet? Why? Why not?

Competitive Situation: Difficulty to Differentiate
1.In what way does Nokia lose when it reveals the source code of their own developments?
2.What do you think, for what reasons are certain parts (hardware drivers, graphical user interface, applications) not reveale
3.Do competing projects (e.g. Op,enMoKo, OLPC etc.) using code from Maemo? What parts?

Conclusions
1.In your opinion: What is the greatest benefit and what is the greatest disadvantage of “opening up” software and hardware products?

8.3 Interview guideline for a Nokia manager

Intro
Since when do you work for Nokia?
Since when have you been working for the Internet tablet project?
What's your role within the project?
What are the goals of the Nokia 770?
Is there a difference for N800?

Choices
Why did you choose Linux instead of Symbian/Windows CE? (What would have been different?)
Why did you choose Debian instead of e.g., Suse or RedHat?
Why did you choose the GNOME/GTK stack and not KDE?
Why didn't you contract an Embedded Linux company such as MontaVista to build the operating system for you? (such as e.g., Motorola)
What were the technical and business reasons to break backwards-compatibility of the OS 2007 Edition? What are the benefits and risks of this decision?

Contractors
Why do you contract small firms such as KernelConcepts? (What are their strengths?)
Which external companies did you contract for the Internet Tablet development?
When do you do things internally? (For example...)
What type of knowledge is transferred to Nokia through contractors? (How?)
How long would it have taken you to create Nokia 770 without the help of contractors?
Why didn't you do it all by yourself?
What was the advantage of choosing OSS?
What are the challenges contracting small firms?

Volunteers
How do you motivate volunteer contributions? (For example...)
What is the influence of revealing source code in the context of motivation?
What kind of contributions are made voluntarily and for what do you have to pay money? (For example...)
How business-critical are voluntary contributions really for the functioning of the Internet Tablet? (Looking at the interviews, it seems as if the most difficult, central things are done by Nokia itself or by contracted firms. Armin Warda said his contribution concerning swap memory was negligible.)
On what occasions do you meet volunteer contributors physically? (How often?)
What is the difference between physical vs. virtual collaboration?
When do you decide to pay somebody for a certain development effort? (For example...)
On what criteria of a developer do you decide to hire somebody? (For example...)
What's different when recruiting somebody from the community than somebody from within Nokia? (e.g., in the case of the Maemo product manager)
What changed in terms of employer attractiveness when Nokia started OSS development?
Do Nokia employees ask technical questions to the volunteer community? (For example...)

Communities
Why do you prefer collaborating with the upstream project? (instead of forking your own version)
In what cases is it better to fork an established project? (Why?)
How do you make Nokia accepted within the OSS communities? (For example...)
In what kind of tasks is Nokia strong compared to the strengths of the volunteer community? (For example...)
In what kind of tasks is the volunteer community strong compared to the strengths of Nokia? (for example...)
What are the challenges being a commercial company within OSS communities? (for example...)

Competitors
Are there competitors in the mobile device industry pursuing the same OSS strategy?
Do you know of cases where competitors used your revealed source code?
How do you collaborate with competitors? (for example...)
What is the benefit of setting standards such as the GNOME Embedded Platform? (Don't competitors catch up more easily if they can integrate a standard and develop their own products faster?)