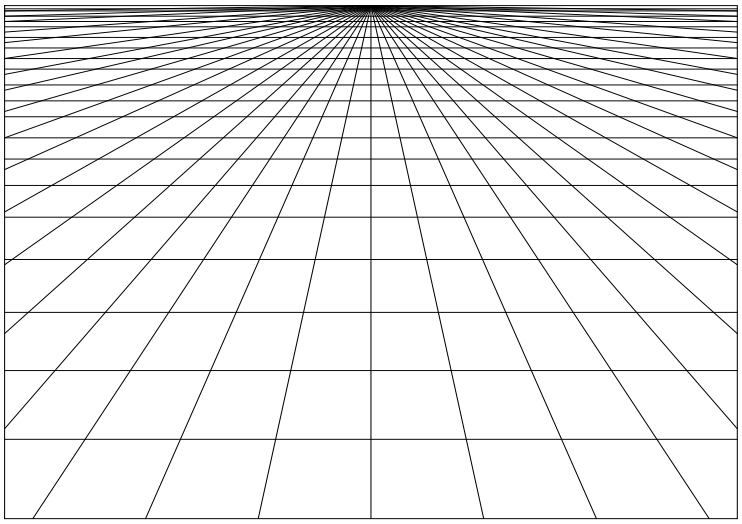
UiO : **Senter for teknologi, innovasjon og kultur**
Det samfunnsvitenskapelige fakultet

Postboks 1108

Blindern

0317 Oslo

www.sv.uio.no/tik



TIK-MA-THESIS

What enables a software firm to persistently reap innovation benefits from a global community of unpaid software developers?

Anders Rekve
08.08.2011

Word count: 31602

Abstract

Since the phenomenon of users innovating scientific instruments was first documented by von Hippel, the aspect of user innovation has been given a lot of attention in innovation literature. With the increased penetration of Information Communication Technologies (ICT) such as computers and the internet, user innovation has become more and more relevant to society. For software firms gaining access to user innovators can be a valuable but also integral part of their innovation strategy. Gaining and keeping access can however be difficult. It is therefore of great interest to learn of examples of how this is achieved. This is an explorative case study of a software firm that has managed this since 1995. By exploring how the firm managed the relationship with a community of unpaid software developers, this paper finds that the firm is able to benefit from user innovators by creating a virtuous cycle. This virtuous cycle relies on a continuous input of software code from user innovators, and a continuous output of new software incorporating the contributed code as well as new code. This paper confirms some previously mentioned managerial challenges to the relationship, such as the importance of using the right licenses. In addition to this, a new managerial challenge has been identified. When the firm becomes a part of a corporation, complexities arise that need to be handled properly. This should be of interest to both managers and innovation researchers.

Acknowledgements

In my application for acceptance for the TIK master program I wrote that I wanted to learn more about the processes where novel innovations make it to become successful commercial businesses. I was particularly interested in innovations relating to Information Communication Technologies (ICT). With this thesis I feel that I have achieved my initial goal and much more. I therefore wish to thank the TIK centre for the opportunity given to me. I also want to extend my gratitude to my supervisor Jarle Moss Hildrum for his help and guidance during the last year. I have deeply appreciated his comments and constructive criticism, and also his encouragement. I would also like to thank the community manager at Qt Development Frameworks. Without his help this research would have been much harder. Lastly I want to thank my dear girlfriend Marthe for bearing with me, and for the support and help she has given me.

Tables and figures

Table of Contents

INTRODUCTION

Development of software consists of writing computer code into a text editor on a computer. This code then becomes the "source code" of the resulting software. "Source code is a list of instructions that make up a "recipe" for software package" (Weber, 2004, p. 4). All software starts out this way. In commercial software the source code is often translated into "binaries", which is a long list of zeros and ones (Weber, 2004, p. 4). This is a way of protecting the work that went into writing the source code, so that potential customers need to buy the product from the firm. In Open Source Software (OSS) the source code is kept visible and accessible for all to see and study (Weber, 2004, pp. 4-5). Surprisingly, there are some firms that employ a business model in which they provide the market with open source software. This means that OSS firms allow users and potential customers access to the computer code it has expended resources into making. Instead of protecting its intellectual product the OSS firm is sharing it with the world. The trade-off is that by giving people an opportunity to study the source code, the firm opens up for giving people the possibility of improving it for them.

OSS is highly innovative (Ebert, 2007; von Hippel & von Krogh, 2009; West & Lakhani, 2008, p. 224). Because OSS firms accept contributions and improvements from users and customers this means one can regard their work and contributions as innovation benefits. Further, the contributions are provided without the exchange of monetary compensation to the contributor (Lakhani & Wolf, 2003, p. 3; S. O'Mahony, 2003). This makes the individual contributor unpaid. Lastly, the contributing software developer is often a part of a larger community of other software developers, such as the developers in the Linux Kernel development (Lee & Cole, 2003). The community is also geographically dispersed (von Hippel, 2005, p. 11). Hence it is a global community of unpaid software developers

It has been shown that software firms are able to reap innovation benefits from so-called user- or innovation communities (Jeppesen, 2005; Jeppesen & Fredriksen, 2006; Jeppesen & Molin, 2003; von Hippel & Katz, 2002). Innovation communities are "[…] nodes consisting of individuals or firms interconnected by information transfer links which may involve face-to-face, electronic, and other communication" (von Hippel, 2005, p. 96). Innovation benefits can be such things as modifications or improvements to the original software. For example, users of a music production software created new graphical designs and added new sound samples to the original software (Jeppesen & Fredriksen, 2006, p. 48). OSS communities are a specific type of innovation community (von Hippel, 2005, p. 11). The

defining characteristic of an OSS community is that it is concerned with developing Open Source Software (OSS), which means that the source code of the software is available and that it can be modified and shared freely (Weber, 2004, p. 5). OSS firms have also been shown able to reap innovation benefits from OSS communities (Dahlander & Magnusson, 2008; Dahlander & Magnusson, 2005; Dahlander & Wallin, 2006). However, OSS communities have strong norms about openness and freedom that can sometimes interfere with the needs of the profit seeking firm (Dahlander & Magnusson, 2005). Apart from two studies (Dahlander & Magnusson, 2008; Dahlander & Magnusson, 2005), there is not a large body of literature about what enables an OSS firm to reap innovation benefits from an OSS community. Building on this literature this paper will therefore address the question of ***what enables an OSS firm in persistently reaping innovation benefits from a global community of unpaid developers.*** The paper will identify a key method that enables the firm to reap innovation benefits, and it will identify a previously unidentified challenge to OSS firms in doing so. As such, the findings of this paper will have some managerial implications.

Background and relevance

The benefit of innovation communities can be explained by drawing on an analogy of 19th century economist Friedrich Hayek's assertion that the market economy was more efficient that the centrally planned economy of the Soviet Union. Hayek "emphasized that at the macro level knowledge is unevenly distributed in society, and that centralized models for economic planning and coordination are prone to failure due to an inability to aggregate this distributed knowledge" . He argued that in the context of economic planning, it is impossible for the planners to collect information about all facets of the economy. There will always be someone else with more information about a particular subject than the economic planners themselves (Hayek, 1945, p. 1). A consequence of this is that Soviet-style 5 year economic plans are inherently inefficient because the economic planners whom decide how to distribute, invest and use the available resources have incomplete information about how to do so most efficiently. A hypothetical example can be if the economic planners decided to enact a 5 year plan to invest in farm machinery, such as tractors. However, farmers in one area might have been more in need of irrigation solutions, instead of tractors. As a result, the increased availability of tractors will only be of marginal help to these farmers, since lacking irrigation is the limiting factor to increased productivity.

Commercial firms may experience similar problems to Soviet-era economic planners. It is impossible for the firm to aggregate all relevant knowledge about the market the firm operates in. As a consequence, the firm might not pick up on changing preferences of its customer base until it is too late, and the customers have gone elsewhere. Also, it is not necessarily so that the firm employs the smartest people available, and thus will not have access to the smartest solutions to issues they find (Lakhani & Panetta, 2007, p. 97). This is why access to innovation communities is particularly valuable to a firm. An innovation community can consists of users of the firm's software whom gather with each other to improve on the product (von Hippel, 2005, pp. 10-11). Users gather in a community because the firm's product does not sufficiently fulfill their needs, and because by joining forces they leverage each others' efforts (von Hippel, 2005, p. 10). Their purpose is to exchange information amongst each other in order to modify the existing product so it will match their requirements (von Hippel, 2005, p. 10). In this manner, innovation communities act as a natural aggregator of market information for the firm. The second reason innovation communities are valuable to the firm is though the fact that the users actively modify and improve the product they are using. A firm may gain access to these innovations through its interaction with the innovation community.

Though interacting with the innovation community is a fairly straight forward process, there are many pitfalls with maintaining a good relationship that allows the firm to use community innovations (Dahlander & Magnusson, 2005, pp. 489-490). At the heart of the problem are the diverging motivations of the people in the innovation community and of the firm. The firm ultimately seeks profit, but the individuals in the innovation community may have other motivations. Their motivation is a mixture of intrinsic and extrinsic motivations such as personal gratification from working on technical challenges or outside recognition for potential future employment (Franke & Hippel, 2003; Lerner & Tirole, 2002; von Hippel & von Krogh, 2009). Also, in the case of OSS communities, there are strong norms and values of keeping community developed computer code open and accessible (S. O'Mahony, 2003). This will sometimes conflict with the firm's goals, which can lead to situations in which the firm's profit motive and the community's openness norms can conflict. The result can be that the community, or parts of it, abandons the software of the OSS firm (Dahlander & Wallin, 2006, p. 1248; von Hippel, 2005, p. 116; Weber, 2004, pp. 239-240). However, in symbiotic relationships, it is clear that both the firm and the community benefit from cooperation (Dahlander & Magnusson, 2005, p. 487). This means that a balance between firm and

community motivation and interests have to be found. Achieving this calls for a community management strategy with a certain amount of *Fingerspitzengefühl* on the part of the firm. This emphasis coincides well with the objective of the TIK course which is to understand the intricacies where technology, innovation and society intersect.

Of the managerial challenges that have been identified include the importance of 1) respecting the rules and values of the OSS community, 2) attentiveness to the effects of licenses, 3) attracting users and developers, 4) contribute resources to managing the community, 5) keeping the community engaged in contributing as the software matures and conditions change, 6) resolve ambiguity about control and ownership and 7) getting acceptance for reaping innovations from the community and avoid direct confrontations (Dahlander & Magnusson, 2005, pp. 489-490). This paper will ask a broader question, not necessarily limited to managerial challenges, of what factors enable an OSS firm to persistently reap innovation benefits from an OSS community. It is suspected that changes over time affect the firm's ability to reap innovation benefits. One reason for this is that the relationship between the firm and the community is an uneasy balance of compromises between core motivations and interests.

Literature contains an anecdote of an instance where the community and the firm's relationship could at least be characterized as strained, perhaps even an instance of direct confrontation (Dahlander & Wallin, 2006, p. 1248; von Hippel, 2005, p. 116; Weber, 2004, pp. 239-240). Upon investigation however, the conflict related in the anecdote is shown to be resolved (Reichard, 2000). Since this was resolved in 2000, there are numerous signs of both a well functioning community-firm relation, increased innovation benefits reaped from the community and an economically healthy firm (Trolltech, 2007). By researching this case it is thought that additional knowledge can be found about what enables OSS firms to persistently reap innovation benefits. The case in question is the Norwegian OSS firm Trolltech[1]. Trolltech has a large innovation community associated with its main software product Qt[2]. The innovation community includes open source developers, hobbyists and firms. However, the largest community is the OSS community consisting of open source developers, which consists of a large base of freelance software developers from all over the world (Qt-Developement-Frameworks, 2011d). Using the case of Trolltech I will examine what factors that facilitate and constrain a software firm's attempts to build up and retain innovation

[1] Acquired by Finnish mobile handset maker Nokia in 2008.
[2] Pronounced "cute"

oriented collaborative relationships with such communities over time. The findings will show that both the firm and the community enjoy a *symbiotic* relationship, which is the core factor as to how Trolltech has been able to persistently reap innovation benefits from the community. Other findings show that community demand led Trolltech to change the licenses it offered after attempts to find compromises was unsuccessful. Also, the case presents alternative solutions to previously known managerial challenges. Lastly, the paper will argue that an acquisition by a larger corporation changed the role of the principal product Qt from being a commercial product to a strategic tool, thereby presenting new managerial challenge.

In relation to literature, this paper extends upon Dahlander and Magnusson's research of Nordic OSS firms and their relationship with OSS communities (Dahlander & Magnusson, 2005). This paper's findings show that Trolltech experienced similar managerial challenges to those found by Dahlander and Magnusson. The findings also present Trolltech's solutions to these problems. The paper gives an account of the circumstances around Trolltech's conflict with the community over licenses, demonstrating that in order to continue reaping innovation benefits Trolltech had to bow to community pressure and offer the community preferred license. Licenses also played a prominent role in the later years when the role of Qt changed. The change of Qt was the transition from being a business in itself, to more of a strategic tool of a larger corporation. Both of these issues had implications for how Trolltech reap innovation benefits, and identify a new managerial challenge to managing a community. Trolltech's conflict with the community over licenses will provide insight into the issue of conflicts between innovation communities and firms, a topic that has been labeled as understudied (West & Lakhani, 2008, p. 229). Further, the findings outline the operational means used by Trolltech in order to handle the relationship with the community. The key finding is found to be verbal and non-verbal communication. Lastly, changing firm structure will be shown to have consequences as to how the firm is able to reap innovation benefits.

The following chapters will be include a literature review were I outline the most important traits of the unpaid software developer. I will also give an account of the literature relevant to my research question. The next chapter will describe my methods for gathering and analyzing the data. I will then present the data in a chapter in my results before I will discuss the results. Lastly I will conclude my findings.

Brief introduction to the case firm Trolltech

Trolltech's main product Qt has been in development since 1991 by the two founders Haavard Nord and Erik Chambre-Eng. Qt is a software development tool that software developers can use when writing new software. The main benefits are that it is faster to program with Qt since Qt contains a library of pre-produced code. This is beneficial because developers do not have to write a program from scratch each time. A fair analogy of Qt is of a Lego set in which some building blocks come in the box, but with the possibility to build a variety of configurations. Another benefit is that once a program is written, it can be made to run on a diverse set of software and hardware platforms. For example, the developer does not need to write a specific version of his or her program to run on Linux, Windows or MacOS. Additionally, the program will also work on different hardware architectures like the x86 architecture (typical in consumer personal computers), and the ARM architecture (typical on low power embedded devices like car computers). To explain why Qt is beneficial one can use the following building analogy; imagine a case in which pre-fabricated houses are made in a factory in Germany and sold throughout Europe. However, every country has its own building regulations. Because of this, the factory in Germany has to build each house specifically for the country in which the house is to be erected. The factory cannot build a house according to French building regulations and expect it to be sold to Norway. Producing pre-fabricated parts for each country's regulation is not surprisingly a cost and time consuming process. The same applies in software production. However, Qt enables software to be written once, but function in many different software and hardware environments. In the words of the community manager at Qt Development Frameworks, "Qt is a cross-platform developer framework" (Jørgenrud & Ervland, 2010). The term "cross-platform" emphasizes the portability across software and hardware platforms mentioned above (Wikipedia, 2011a). The term "developer framework" points to the fact that a developer will use the Qt framework to develop new software.

Qt was made public in 1995, at which point it was offered under both a commercial license and a free license (Blanchette & Summerfield, 2006, p. xvi). To date, Qt is still offered under both commercial and (various) free licenses. This so called dual-licensing is an established business model for open source software firms (Prowse, 2010). To some extent dual licensing solves one of the inherent problems for open source software firms. The firm needs revenue to continue development, at the same time it needs to be adopted by a large number of software developers in order to create a user (and customer) base. By offering a

free license the user base can grow, while commercial licenses generate revenue. However, dual licensing is not without drawbacks. Some industry reports quote the proportional use of dual licensing schemes to have declined from around 20 % to around 5 % in the later years (Prowse, 2010). Dual licensing adds a legal complexity to both the firm's business, product development and to the customer. This complexity is prevalent in the history of Trolltech. Community pages are littered with discussions as to what a particular license allows for.

In 1996 Trolltech had managed to sell 18 commercial licenses, with one of the customers being the European Space Agency (Blanchette & Summerfield, 2006, p. xvi). During the same year, a major OSS project called KDE was started by German programmer Matthias Ettrich. KDE used Qt in developing a Graphical User Interface (GUI) for Linux.[3] The sale of commercial licenses meant that additional developers could be hired, and the KDE project made Qt "the de facto standard for C++ GUI development on Linux"[4] (Blanchette & Summerfield, 2006, p. xvi). These early developments meant that Trolltech had both been adopted by developers in the OSS community, as well as by other commercial firms. The importance of KDE is quite substantial. Getting an OSS community involved using Qt, means there will be more people available to spot "bugs" or errors in the source code. By having advanced users whom have access to the source code, these users can "[…]diagnose problems, suggest fixes, and help improve the code far more quickly than you could unaided (Raymond, 2000, p. 6). As of to date, users contribute to Qt by reporting bugs, offering support on Internet Relay Chat (IRC), translating documentation and contributing to the Qt source code itself (Qt-Developement-Frameworks, 2011a).

The first free license of Qt, the "Free Qt license", did not fit the definition of "Open Source"(Sweet, 2000). Nor did the license comply with the definition of Free Software as was defined by the Free Software Foundation (FSF) (Sweet, 2000). The major fault of the "Free Qt license" was that although anyone could use Qt if the resulting software was issued under a free software license, Trolltech did not offer access to the source code. Hence, Trolltech did not allow any redistribution of any modified source code of Qt itself (Sweet, 2000). As such, Qt was neither free (free as in free speech), nor was it truly open source. From 1999, Trolltech offered Qt under a new license called the Q Public License (QPL). This license complied with

[3] A GUI is a program that "draws" the graphical interface most of us are familiar with using on a computer. Before GUIs, human interaction with a computer was based on writing text based commands to the computer in a command prompt. Microsoft's Windows is a GUI which made human interaction and use of computers much easier by using a computer mouse to navigate visually amongst the windows on the desktop.
[4] C++ is a specific type of computer programming language.

both the newly formed Open Source Initiative (OSI) organization's definition of "Open Source" and the FSF's definition of "Free software" (GNU-Operating-System, 2011a; Open-Source-Initiative, 2011). Distribution of modified versions of Qt was allowed, but with some restrictions as to the technical method of how this was to be done. Because of these restrictions QPL did not comply with FSF's standard license the General Public License (GPL) (GNU-Operating-System, 2011a). During my interview with the former Trolltech CEO, he told me the reason QPL did not comply with GPL "was a fault of GPL, not QPL". By this he meant that Trolltech had already provided the OSS community with an OSS friendly license, and that FSF did not agree was the problem of FSF, not Trolltech. In addition to the QPL, Trolltech had also constructed a foundation called the KDE free Qt Foundation. The foundation was set up in order to secure the availability of Qt for the development of free software. Should Trolltech for some reason decide not to continue developing Qt, or fail to release a new version within a year, the latest version of Qt would be made available under a very permissive license called BSD (KDE, 2011). The former CEO told me that this was meant to reassure the community that they would always have access to Qt. The table below summarizes the free license changes of Qt over the years.

table 1: Qt licenses

Year	Qt version	License name	Qualities of license
1995-1999	v0.90 to v.1.45	Free Qt license	Did not comply with definitions of Open Source or Free Software.
1999-2007	v.2.00 – v.3.3.8	QPL	Complied with the Open Source Definition and the Free Software Definition. But not with the GPL license.
2000-2011	v.2.20 -	GPL	All work that in even just in a small part derives from GPL licensed code has to be licensed in full as GPL.
2009-2011	v.4.5.0-	LGPL	LGPL licensed code can, under some circumstances, be used in closed source applications.

Sources: (GNU-Operating-System, 2011a; Open-Source-Initiative, 2011; Qt-Developement-Frameworks, 2000, 2005, 2007, 2009, 2011b; Sweet, 2000; Wikipedia, 2011c)

From the reader's perspective, the emphasis on what the FSF and OSI define as "free software", "open source" and proper licenses might seem strange. Especially since in many ways the intent of Trolltech with both the "Free Qt license" and QPL is clear; Trolltech wanted users to be able to use and modify Qt for free, for non-commercial purposes. The reason for the emphasis is the considerable normative power and definition power held by both organizations in the OSS community (Perens, 1999). In part, it was FSF's normative power that led to the creation of an alternative to KDE, called GNOME in 1996 (Icaza, 2005). This instance is the aforementioned case of conflict between the OSS community and Trolltech which has been mentioned in literature (Dahlander & Wallin, 2006, p. 1248; von Hippel, 2005, p. 116; Weber, 2004, pp. 239-240). The reason for starting GNOME was that the foundation for KDE, which was Qt, was licensed under the "Free Qt license". According to Miguel de Icaza, FSF founder Richard Stallman him against using KDE for his purposes. As a result, de Icaza started GNOME (Icaza, 2005).

When Trolltech eventually offered Qt under GPL, Trolltech Co-founder Eirik Chambre-Eng explained that the reason Trolltech took so long to offer Qt under GPL was Trolltech's worry that someone might make a "fork"[5] of Qt (Fremy, 2003). According to Chambre-Eng, this would mean that Trolltech could lose control over Qt, and potentially lose their source of "bread and butter" (Fremy, 2003). The same sentiment was shared by the former CEO of Trolltech in my interviews, with the additional comment that any change to more permissive licenses is impossible to reverse. In short, releasing Qt under GPL might have been both damaging and irreversible to Trolltech. Hence, he told me, a thorough discussion internally was conducted in order to balance the needs and wants of the OSS community against the potential negative risks to the firm. Further, he said that one way of negating the risks associated with releasing under GPL was to ensure that both Qt and Trolltech had evolved to a point where any "fork" would be at a future disadvantage. The momentum of the official version of Qt, as well as the company behind it would be hard to match be competitors. In some way, before exposing Trolltech to potential "forks", Trolltech wanted to make the most of its first mover advantage.

In 2008 Trolltech was acquired by Finnish mobile handset maker Nokia. The intent of Nokia was to use Qt and Trolltech as a means to bolster its cross-platform for mobile devices

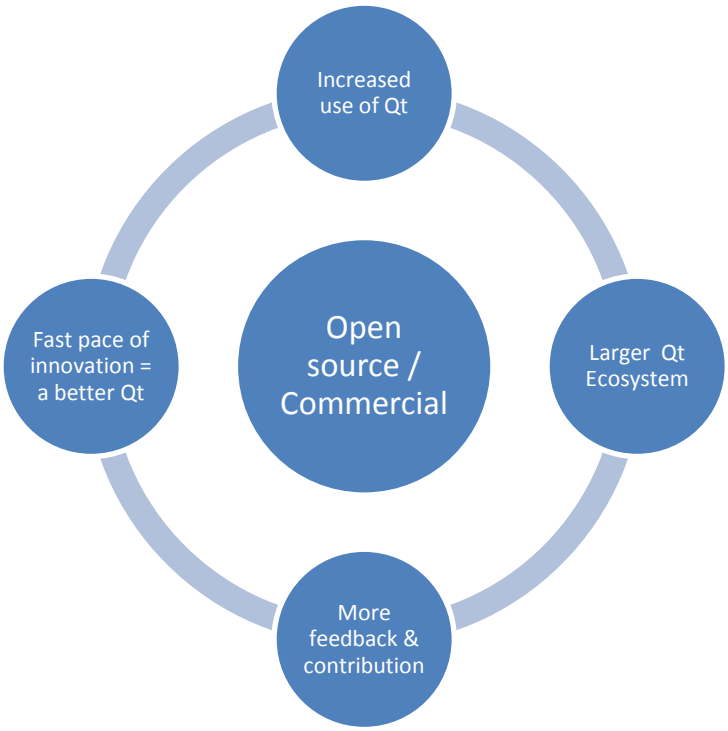
[5] A "fork" is a term used to describe "[…]when developers take a legal copy of source code from one software package and start independent development on it, creating a distinct piece of software" (Wikipedia, 2011b). Under the terms of GPL anyone can take the source code of Qt and start a new project with it as long as the resulting software is licensed under GPL.

and desktop applications (Qt-Developement-Frameworks, 2008a). Some saw this move as a step in Nokia's quest to enable an "app" market for its mobile telephones(Moconews, 2008). In early 2009 Qt was offered under a new license called Lesser General Public License (LGPL). According to my interviews, the LGPL license meant that Qt libraries, or parts of Qt, could be used as parts of proprietary software. The condition is that the parts used would have to remain open source (visible amongst the proprietary code), and that any modification done to the Qt libraries or Qt code has to be licensed under LGPL. The stated reason for this decision was to increase the adoption of Qt (Qt-Developement-Frameworks, 2009). Another benefit of licensing under LGPL related to me during my interviews was that by doing so commercial firms that use Qt, but need to improve parts of Qt in order to do so will have to share this improved code back to the Trolltech.

In 2011, two major developments affected Trolltech; Firstly, Nokia announced that future Nokia handsets would be based on Microsoft's Windows Phone 7 operating system. Following this decision, the commercial sales and support division of Qt was sold to Finnish company Digia. Both decisions spurred a substantial amount of debate on community pages about the future of Qt. However, as of the time of writing this paper, no particular negative effects have been identified in this respect.

Trolltech's innovation strategy depends on a striving community. The importance of a strong and growing community can be characterized as a feedback loop, or a network effect. The network effect is that through a larger community, Trolltech receives more and more contributions and feedback from the community. This again triggers a faster innovation pace of Qt, which increases the quality of Qt, and leads to more users of Qt. More users of Qt lead to a larger community, and so on. Trolltech refers to this as a virtuous cycle. Both commercial and open source Qt benefits from the same cycle since both benefits from programming using a better product.

figure 1: Trolltech's virtuous cycle



Copied from (Qt-Developement-Frameworks, 2010, p. 7)

Name of firm used in this paper

When Trolltech was acquired by Nokia, it was renamed "Qt Software", and subsequently "Qt Development Frameworks". Throughout this paper, I will simply refer to the firm as Trolltech. The first reason for this is that it is simpler than switching between Trolltech, Qt Software or Qt Development Frameworks depending what year it is referred to. The second reason is that Trolltech was the name of the firm for the longest period of time, and since this paper is a longitudinal section study of the firm behind the open source software named Qt. By longitudinal section I mean that I am investigating a section of Trolltech's relationship with the OSS community over time. Lastly, Trolltech is the name of the firm that is already known to previous literature. Combined with the two other reasons; referring to the firm as Trolltech seems the way of least resistance and of least confusion.

The focus in this paper is on reaping innovation benefits from unpaid software developers. Throughout this paper the term OSS community, innovation community or just community is used as a replacement for "unpaid software developer". The reasons for this is are many, and needs some explanation. In short, the individual software developer can be said to have specific traits and qualities that make him innovate and share his innovations. He or she will most likely be a user innovator, lead user and be a holder of "sticky information". Also, the use of ICT such as the internet facilitates cooperation between many individual developers, which leads to new methods of organizing work and innovation, for example through communities. Such communities can have various labels such as user community, innovation community or OSS community. This paper will not concern itself too much with a discussion around these labels. There is still some ambiguity in innovation literature about what labels or terms to use, and this is not the focus of this paper (West & Lakhani, 2008). The following chapter will provide the reader an account of relevant literature with respect to the research question, and gradually introduce innovation literature specific to OSS communities.

The User Innovator and the Lead User

The discovery of the phenomenon of user innovation came through research by Eric von Hippel in 1976 (von Hippel, 1976). Eric von Hippel was interested in the roles scientific instrument users and manufacturers played in developing successfully commercialized innovations. He found that "[…] approximately 80% of the innovations judged by users to offer them a significant increment in functional utility were in fact invented, prototyped and first field-tested by users of the instrument rather than by an instrument manufacturer." (von Hippel, 1976, p. 212). He also observed that the motivational factor for the inventive users (as he called the user innovators) seemed to be twofold. Firstly, some modified and improved the equipment in order to better fulfill the needs they experienced. Secondly, some were motivated by better understanding the operation of the equipment (von Hippel, 1976, p. 235).

Eric von Hippel's research continued to focus on user innovations and creating a better understanding of the phenomenon (von Hippel, 2011). In 1986, he introduced the term "Lead User" which explained a key characteristic of the user involved in user innovation: Common to lead users is that they have strong and specific needs towards a product. These needs are

more advanced than what the rest of the market experience. As the market becomes more advanced with time, the rest of the market will experience the same needs as the lead users once did (von Hippel, 1986). Hence, the needs of the lead user lie in the forefront of the marketplace. Because of this quality in lead users, they can provide the firm with valuable information about future market needs; they become a "[…] need-forecasting laboratory for marketing research." (von Hippel, 1986, p. 791). Additionally, lead users show the capacity to fulfill their needs by acting as a user innovator. As such, lead users may be used to provide product designs to a manufacturer (von Hippel, 1986, p. 791). The definition of a lead user by von Hippel is the following:

> *[…] users (1) who are at the leading edge of each identified trend in terms of related new product and process needs and (2) who expect to obtain a relatively high net benefit from solutions to those needs (von Hippel, 1986, p. 798).*

The concept that lead users will have comparatively more advanced needs than ordinary users should be well known. For example, race car drivers will experience quality and performance related needs of their cars before the general motorist. The race car driver will relay this information to the auto manufacturer on the basis on his experiences. The main difference will be that in motorsports the relationship between the race car driver and the auto manufacturer often will be more pronounced. Typically the driver is either employed or at least sponsored by the auto manufacturer. This was not the case in von Hippel's lead user. The lead user is not employed by the firm responsible for the product he or she is a user of. To continue on the car analogy, the lead user seems to have more in common with an average owner of a car. For example, a car owner may experience some deficiencies with his or her car. As a consequence he or she may try to modify and improve this deficiency. This does not have to be advanced engineering; it can be small things to such as responding to a need for more cup holders. The user of the car can then come up with a novel method to use existing resources in order to solve a problem of too few cup holders. According to the definition used by Schumpeter, this is then an innovation, since the car owner used existing resources and recombined them into something new (Jan Fagerberg, David C. Mowery, & Nelson, 2006, p. 6).

Sticky information

The user innovator is a key component in the innovation community. The user innovator holds a significant knowledge and expertise about the product in which the community is centered on. However, information can be hard to transfer from one holder to another. This is the central idea behind the term "sticky information", as introduced by von Hippel in 1994:

> *We define the stickiness of a given unit of information in a given instance as the incremental expenditure required to transfer that unit of information to a specified locus in a form usable by a given information seeker. When this cost is low, information stickiness is low; when it is high, stickiness is high (von Hippel, 1994, p. 430).*

For example, "learning by doing" is a type of information that sticks to the person doing the activity. For example, the maintenance cost of a particular jet engine had decreased to 30 % of the initial costs after a decade (von Hippel, 1994, p. 432). The reason for this is that the people doing the maintenance had learned how to do it more efficiently over time (von Hippel, 1994, p. 432). In this example, the users (maintenance workers) of the jet engine are the holders of valuable information. But in practice the information is hard to share with others. This can be explained by the fact that it took the maintenance crew 10 years to acquire it. Hence, in cases where information is sticky, problem solving will happen close to the holders of the sticky information (von Hippel, 1994, p. 432). This is exactly what was found in a case of user innovators concerned with innovating mountain biking equipment. The user innovators utilized their own information about their needs, and their own problem solving information in order to arrive at a solution (Lüthje, Herstatt, & von Hippel, 2005, p. 951).

It has been found that user innovators often exhibit the characteristics of a lead user (von Hippel, 2005, p. 4). Hence, the information held by the lead user can be characterized as being sticky information. When user innovators congregate in innovation communities in order to share information amongst each other, these communities are valuable to a firm in the sense that the firm gains access to the user innovators' sticky information. For example, through innovation communities, a firm might gain access to innovations to implement in future products to be sold (Jeppesen & Fredriksen, 2006, p. 45). Access to innovation communities will substantially lower the costs of information transfer, enabling the firm to

learn of users' needs that they might otherwise have to expend large sums to capture through market research.

One medium of knowledge transfer that has helped lowering the cost of gaining access to sticky information is through the use of ICT (Jeppesen & Fredriksen, 2006, p. 45). Computers, but mostly the use of Internet-related communications such as emails and forums, help user-user interaction and user-firm interaction. Increasingly high penetration of computers and internet connectivity begets increasingly less sticky information stickiness. ICT is perhaps the most important underlying factor that has enabled user communities. The lower cost of communication between users enables more communication, which leads to a larger solutions space for existing shared problems. But more importantly, the organizational consequence of ICT is the manner in which user efforts often are organized: into innovation communities.

Free revealing

A key quality of user innovators is that most often they are not opposed to sharing their innovations. As an example of this quality, the following exchange demonstrates this general feature of user innovators. The exchange occurs on the British television show Top Gear, where the show host visits a car modification community in the UK. The background for the exchange is that car manufacturer Citroën has used the community's modifications as input to a new car model.

> *Interviewee: "They're taking all the stuff that we're doing to ours, our older cars, and now they're doing it in production cars, aren't they?"*
> *Show host: "Is that a bad thing?"*
> *Interviewee: "...no"*
>
> Source: (Klein, 2003)

The same willingness to accept, if not allow, firms access to the community's innovations has been extensively documented in innovation literature (Harhoff, Henkel, & von Hippel, 2003; von Hippel, 2005; von Hippel & von Krogh, 2009). Instances of free revealing have been identified between firms, but also between user innovator and firm. With

respect to firm to firm free revealing, an often quoted example is a case discovered by Allen (1983). In this case it was discovered that innovations and knowledge was systematically shared between employees of iron smelting plants in 19th century England. Innovations about the design of smelting plants were shared between engineers in publications and professional meetings. The end result for the firms was a more efficient smelting process. Other than that, the rationale for free revealing by firms was: 1) the firm gains reputation; 2) holding the innovation secret might represent costs to the firm; 3) the innovation might highlight or supplement firm-specific qualities, and as such separate the firm from others (Allen, 1983; Harhoff, et al., 2003, pp. 6-7; von Hippel, 2005, pp. 78-91). Free revealing by user innovators have also been found to be true with respect to innovations in sports equipment (Franke & Shah, 2003). For the individual user it has been argued that increased diffusion of the innovation represent gains for the user innovator in the forms of "network effects, reputational gains, and related innovations induced among and revealed by other users" (Harhoff, et al., 2003, p. 7).

An example of where user innovators reveal their work is within software development, specifically in open source software (OSS). Though it has been pointed out that the free revealing of open source is not quite the same as von Hippel's definition of free revealing, von Hippel argues that the effects are similar (von Hippel, 2007, p. 305). The reason for arguing that free revealing in OSS is different is because OSS projects typically employ licenses to their work that may inhibit firms in using the innovations (Von Hippel & Von Krogh, 2006, p. 297). An inherent principle in an OSS community is that others are to have full access to both the resulting product as well as the computer code that it consists of. The Linux operating system is a well known example of an OSS project. The inception of the principle of free revealing within software production is most often cited with a story about Richard Stallman's irritation that Xerox would not allow him access to the source code of the office printer (Weber, 2004, pp. 46-47).[6] The previous printer at his office had been modified by Stallman to issue a notification when a user's prints were done. Xerox did not allow Stallman access, and as such the time saving feature of print notification was not available any longer (Williams, 2002, p. chapter 1). A few years later Stallman started the Free Software Foundation which was tasked with working for free and open software (Weber,

[6] The dawn of open source software has earlier roots going back to collaborative efforts to write a compiler (a program that translates human symbols like letters and numbers to machine code which is 0 and 1) in the 1950s (Weber, 2004, pp. 21-22).

2004, p. 47). The word "free" refers to freedom, not necessarily at no cost of money. The freedom Stallman wanted was to have the ability to do the following:

1. Freedom to run the program for any purpose
2. Freedom to study how the program works and to modify it to suit your needs
3. Freedom to redistribute copies, either gratis or for a monetary fee
4. Freedom to change and improve the program to the public so others can benefit from your improvements.

   Copied from (Weber, 2004, p. 48)

The prerequisite to this is of course that the software code is open source to begin with, meaning that it is possible to read the code lines that make up the software. When reading about open source software, one will often see the abbreviations FOSS and OSS. FOSS is in reference to Stallman's emphasis on *free*, therefore the abbreviation Free and Open Source Software (FOSS). The abbreviation OSS used in this paper is used because it is perceived by the author as more neutral. For example, the emphasis is on the act of programming with open source software, not necessarily on the normative views on what "rights", or freedoms, people should have. Still, one cannot overlook the importance of Richard Stallman's work on for example software licenses. Stallman, with the help of others, created the General Public License (GPL), a commonly used license with respect to publishing open source software. GPL licensed software ensures that anyone can have the freedoms mentioned above. It must be stressed that OSS communities are very adamant about protecting their work both through strongly held norms in the community and by using OSS licenses (S. O'Mahony, 2003).

For free revealing, the GPL thus provides a legal institution in which programmers can use. However, having such an institution does not speak to the reasons programmers choose to freely reveal their work. In general, the motivations of user innovators to freely reveal can be seen as either intrinsic or extrinsic (Lakhani & Wolf, 2003). The extrinsic reasons can be divided in two; one is the issue of the costs involved by choosing not to reveal, the other is the benefits from free revealing (von Hippel, 2005, p. 77). For example, for user innovators free revealing is often the best course of action because there are few manners in which the user innovator can "[…] protect their innovation from direct or approximate imitation." (Harhoff,

et al., 2003; von Hippel, 2005, pp. 80-81). Hence, no additional value can be achieved from their work if they choose to keep it secret. Also, by freely revealing their innovation user innovators position themselves for the potential positive effects (von Hippel, 2005, p. 10). In relation to OSS, the user innovator can benefit from signaling his or her own competence. By sharing their innovations the user innovator can gain reputation, which may make him or her a more likely candidate for employment by firms (von Hippel, 2005, p. 86). In relation to Trolltech and Qt, this was seen when the founder of the KDE project, Matthias Ettrich, was hired by Trolltech in 1998 (Blanchette & Summerfield, 2006, p. xvi). Although, the fact that Trolltech hired a merited community developer could be seen as self-serving in the sense that in doing so, it may increase the reputation of the firm itself (von Hippel, 2009, p. 17).

The intrinsic reasons for free revealing in specific reference to open source suggest contribution (and thereby freely revealing work) to open source projects is based on a sense of altruism (Kogut & Metiu, 2001, p. 258). On the other hand, Eric Raymond argues that "altruism is itself a form of ego satisfaction for the altruist" (Raymond, 2000, p. 22). Hence, being altruistic is goal in itself. Others suggest that free revealing might be seen as a community norm (von Hippel, 2009, p. 18). There is a certain sense of logic behind this, because influential community members such as Richard Stallman is a vocal advocate of what essentially boils down to free revealing, though with some restrictions. This view is supported by scholars who argue (Lakhani & Wolf, 2003, p. 5) there is a strong adherence to community norms within OSS projects. A related concept is that programmers may want to reciprocate to other programmers because they themselves have benefitted from the work of others (von Hippel, 2009, p. 17). On the other side, developers have been found to be motivated by personally rewarding factors, such as the enjoyment they feel when working on open source projects (Lakhani & Wolf, 2003, p. 3). The joy of solving tasks creatively and overcoming intellectual challenges range among the reasons developers experience joy when working with open source projects (Lakhani & Wolf, 2003).

User Toolkits

User toolkits are a set of tools supplied to the user by the firm. By using toolkits the firm can "[…] *abandon* the attempt to understand user needs in detail in favor of transferring *need-related* aspects of product and service development to users" (von Hippel & Katz, 2002, p. 821). In terms of cost saving, the firm does not need to exert the same effort and money in acquiring users' sticky information since the development is done by the holders of that information. The cost saving aspect however is not absolute, as the firm might need to increase its support to users because of the toolkit. All though, some of this support might be handled by other users, offering user-to-user support (Jeppesen, 2005). An underlying benefit of user toolkits is that they offer firms a practical way of facilitating and to a certain extent manage user innovation (von Hippel, 2005, p. 16). This can happen through adapting the options available to users of the toolkit. For example, the toolkit might be made specifically to create new game scenarios, but be impossible to use to alter other aspects of the game (Bo Jeppesen & Molin, 2003, pp. 369-370). An underlying *feature* of user toolkits is the real innovations that emerge from user communities working with the toolkit (Bo Jeppesen & Molin, 2003, p. 364). As such user toolkits also facilitate for innovation where it would otherwise not be found. Or, it can help facilitate for innovations that would otherwise not happen.

To my knowledge, in the OSS community, no user toolkits have been issued with the goal of increasing the quality of another product. To some extent, this makes sense, as the OSS community by definition is open for all to contribute. There is no need for user toolkits, as there are a multitude of other venues a programmer or user can utilize if they feel like contributing. Still, I find that user toolkits and Qt are very similar. Indeed, one might call Qt a user toolkit, because it provides the user with the tools in which to create new projects or programs, like the KDE-project. However, KDE did not start Qt in order to improve KDE. Instead, the KDE-project utilized Qt to make KDE. The difference lies in that Qt *is* the product, it is not an additional service given to users to improve another product.

From a contemporary viewpoint, one can see evidence that toolkits has become commonplace. For example, a key factor to Apple's iPhone success comes from the number of "apps" (applications) available. To develop "apps", Apple turned to outside resources, much in line with the ideas of Open Innovation (Chesbrough, 2003). By offering a Software Development Kit (SDK), Apple ensured that anyone, both firms and private persons, can develop "apps" to be sold through Apple's "App store". The distinction between Apple's

SDK and user toolkits is that Apple's creation of a market for "apps" meant that it attracted firms too. Although firms can be users of a product, in this example, the role of the firms was more in line with the traditional software firm. Also, the creation of an "app"-market meant that the principle of free revealing is not present.

What is an innovation community?

This paper focuses on what enables an OSS firm to persistently reap innovation gains from a global community of unpaid software developers. These unpaid software developers have certain traits which I have identified above. In the introduction I have used the term *innovation community* and by quoting Eric von Hippel's definition of an innovation community, I have established that an *OSS community* is an innovation community (von Hippel, 2005, p. 96). The following chapter will give an account of the literature relating to the qualities of innovation and OSS communities. First however, I will define the concept of an innovation community.

From an intuitive perspective, most are familiar with the term "community". It is a group of people whom have a characteristic, attribute or quality which is *common* to their group (Dictionary.com, 2011). In this case the term "innovation" specifies what this shared characteristic is. This paper will use the following definition of "community":

> *[…]by building upon the definition of Gläser (2001), we consider a community to be a voluntary association of actors, typically lacking in a priori common organizational affiliation (i.e. not working for the same firm) but united by a shared instrumental goal—in this case, creating, adapting, adopting or disseminating innovations. (West & Lakhani, 2008, p. 224)*

Eric von Hippel defines innovation communities as: "[…] nodes consisting of individuals or firms interconnected by information transfer links which may involve face-to-face, electronic, and other communication" (von Hippel, 2005, p. 96). In von Hippel's definition, the explicit mention of firms being part of the innovation community is relevant to this paper. This is because firms have been shown to be significant parties to the type of innovation community associated with the case chosen (Lakhani & Wolf, 2003, p. 3). The chosen case study is an innovation community consisting of unpaid software developers distributed across the globe. Since this innovation community is concerned with developing software which is open

source, the same community might in some cases be labeled an OSS community. This paper will use the term OSS community or just "community" where it is implicitly clear that the type of community in question is an OSS community. Lastly, it is taken for granted in this paper that the OSS community inherently is innovative. The definition of "innovation" used here is taken from Schumpeter in which an innovation is "new combinations of existing resources" (Fagerberg, Mowery, & Nelson, 2006, p. 6). The aspect of whether or not community co-developed open source software can be classified as being innovation has been debated by both scholars and industry participants according to West and Lakhani (2008, p.1). However, by citing the example of the Apache web server and the reflections of Eric von Hippel, they conclude that there exists ample evidence of open source software being innovative (West & Lakhani, 2008, p. 1).

Many innovation communities use Information Communication Technologies (ICT) such as the Internet to communicate. This means that there is no geographic restriction as to who can participate in the innovation community. It also means that innovation communities, such as the case chosen in this paper, are global communities. The nature of being global communities concerned with innovation means innovation communities in many cases can be described as being instances of *distributed innovation.* OSS communities are described as being "the most fully developed example of the appearance of distributed innovation systems" (Lakhani & Panetta, 2007, p. 98). Distributed innovation systems are "characterized by decentralized problem solving, self-selected participation, self-organizing coordination and collaboration, "free" revealing of knowledge, and hybrid organizational models that blend community with commercial success" (Lakhani & Panetta, 2007, p. 98).

Lastly, the idea of firms using innovation communities as inputs in the firm's innovation process folds into the paradigm of Open Innovation. Open Innovation was coined by Henry Chesbrough in 2003, and refers to the idea that a firm should allow for more permeable borders between the firm the rest of the environment (Chesbrough, 2003). Knowledge is seen as widely distributed, and in order for the firm to advance its technology it needs to cooperate with other firms though acquisitions and/or licensing technology (Chesbrough, 2003). The opposite, and perhaps more traditional view is that the firm needs to keep its innovation process secret and closed, thereby the term Closed Innovation (Chesbrough, 2003, p. xxvi). The following table summarizes the principles of Open Innovation:

table 2: Open Innovation

Open Innovation
Not all the smart people work for you
External ideas can help create value, but it takes internal R&D to claim a portion of that value for you
It is better to build the best business model than to get to market first
If you make the best use of internal and external ideas, you will win
Not only should you profit from other's use of your intellectual property (IP), you should also buy others' IP whenever it advances your own business model
You should expand R&D's role to include not only knowledge generation, but knowledge brokering as well

Copied from (Tidd & Bessant, 2009, p. 295)

Illustration

By observing that innovation literature includes a welter of overlapping terms and definitions of communities in innovation (West & Lakhani, 2008), I am allowing myself to organize some examples in the illustration below. The overarching purpose of this paper is to say something substantive about relationships between an OSS firm and an OSS community's relationship. Therefore, the examples are organized according to the relationship between the locus of innovation (community) and the firm (if present). The purpose is to situate, and create a visual representation, of where Qt is in relation to other examples mentioned in literature.

As I argued above, innovation communities are a form of distributed innovation systems. I also mentioned the case of Apple, where firms and individuals participate in producing "apps" for the iPhone. As the illustration shows, Apple's App Store is characterized as being Open Innovation. Since there are no geographical restrictions as to who can produce new "apps", the App Store is in effect an example of distributed innovation. However, it is not a true example of an innovation community, because I find that most of the "apps" that are sold originate from private firms. User innovation is therefore too small compared to the open innovation paradigm in order to characterize the App Store as an innovation community.

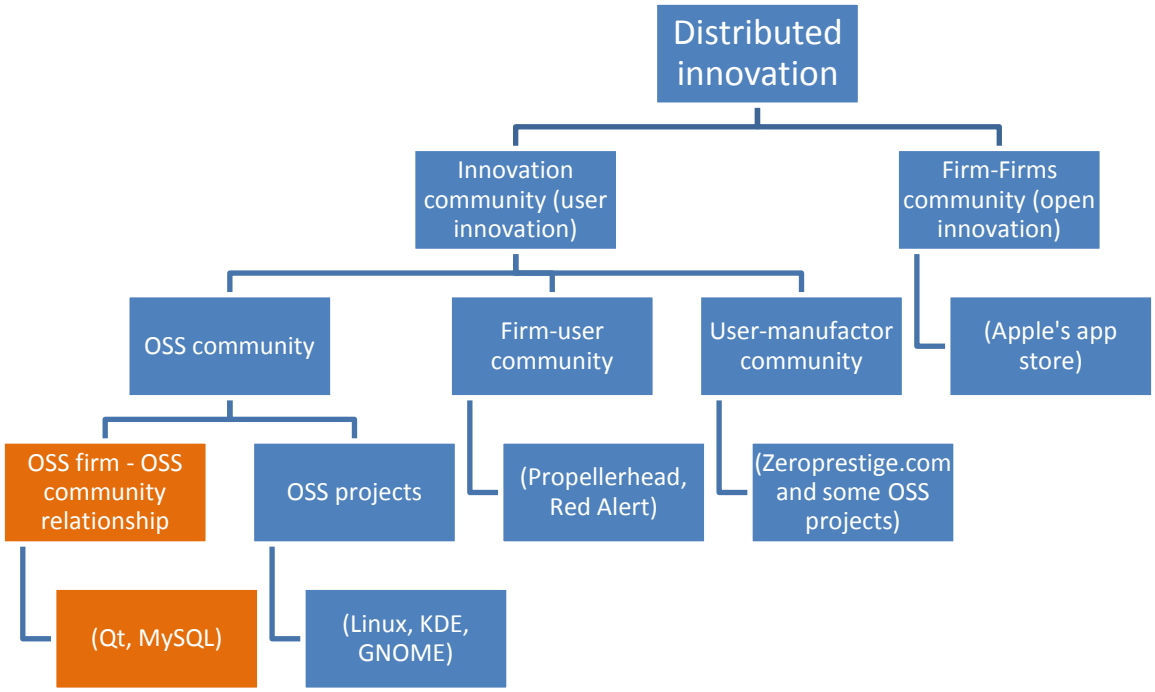


figure 2: Illustration of communities

In the second category, Innovation community (user innovation), all sub-categories rely heavily on user innovation. The three sub-categories here are the OSS community, the firm-user community and the user-manufacturer community. In the case of a user-manufacturer community, the users themselves perform all the functions, from product design to manufacturing. For example, von Hippel (2005, pp. 124-126) mentions the case of kite surfing communities. Kite surfing is a sport in which a person is attached to a large kite which then propels him or her forward. Kite surfing takes place either on water or land, where the kite surfers usually use water-skis or regular skis on their feet. Kite surfers exchange designs for kites that the users have co-developed with each other on websites such as Zeroprestige.com. The kite surfer then takes the design to a sail maker who produce the design, or a manufacturing company may download the design and produce it themselves

(von Hippel, 2005, pp. 124-126). The key aspect of this kind of community is that there is no reliance of the community on any specific firm. Also, to a certain extent, some OSS project also fall into this category since they too can be independent of any firm. The projects are started, designed and written (manufactured) by individuals in the community.

The second category is where there is an explicit firm-user community. Notable examples here are the firm-user community co-development of new game scenarios for the computer game Red Alert 2 (Jeppesen, 2005) and firm-user community co-development of additional functionality for the music production software Propellerhead (Jeppesen & Fredriksen, 2006). In the case of Red Alert 2, the firm responsible for the game offered the users a User Toolkit which enabled them to create new maps, or game scenarios. Red Alert 2 is a war-strategy game in which the players battle against each other in a make believe world. Available to the players in this world are certain resources and topographies that can give strategic benefits to the player that gets to them first. By allowing for users to create new maps, the firm saved the developmental costs of doing it themselves, all though it should be mentioned that the costs relating to supporting the users went up (Jeppesen, 2005).

The third category of OSS communities is divided into two sub categories. Common to both of them is that they involve a community relating to open source software. However, in the case of some OSS projects, there are no firms involved. For example, KDE, GNOME and Linux are community projects that started without any firm involved. Firms may contribute to the projects, as they have been found to do in some cases by employing developers to contribute code to OSS projects (Dahlander & Wallin, 2006). But, they have no hierarchical control over the project (Dahlander & Wallin, 2006), nor did they contribute to the project's inception.

The last sub-category is the one in which Qt falls into. Both Qt and MySQL are open source projects that was started as a firm, but in which the source code eventually was issued under an OSS license (Dahlander & Magnusson, 2008; Dahlander & Magnusson, 2005). Common to both these examples is that the software is mainly developed within the firm, but relies on a symbiotic relationship with the OSS community where the firm gets code contributions from the community in return for providing the software under an OSS license (Dahlander & Magnusson, 2008; Dahlander & Magnusson, 2005).

The research question in relation to literature

A lot of the research into innovation communities and firms employ cases that accentuate a constructive relationship between the two. By this I mean that the relationship results in added value for at least one of the entities. For example, the case of Propellerhead resulted in a better product, which benefitted both the firm and the innovation community (Jeppesen & Fredriksen, 2006). An alternative view is to describe these relationships as symbiotic, in which both parties benefit from cooperation (Dahlander & Magnusson, 2005). But innovation communities can also be exploitive, where their efforts may damage the firm. Such examples have been termed instances of "Outlaw Innovation" (Flowers, 2008). In the same manner, a firm might act as a parasite and gain at the expense of the community (Dahlander & Magnusson, 2005, pp. 487-489).

On the basis of the literature mentioned in this chapter, I am assuming that in order for the firm to reap innovation benefits from an innovation community, the firm needs to establish and maintain a relationship with the community in question. In the case of Outlaw Innovation, no relationship exists. In the case of many OSS projects, no firm is involved. In the case of Trolltech and Qt however, there is a relationship between Trolltech and an OSS community. As such, the case of Qt fits into a tendency of focusing on constructive relationships between firm and community. The additional attribute is that the community is a specific type of innovation community – an OSS community - with an associated set of norms and values.

On the issue of relationships between an OSS-firm and an OSS-community, managerial issues vary depending on the type of relationship between the firm and the community (Dahlander & Magnusson, 2005). It has been found that these typologies could be characterized as either symbiotic, commercialistic or parasitic (Dahlander & Magnusson, 2005, pp. 487-489). The associated managerial issues are listed in the following table:

table 3: Managerial issues with respect to OSS communities

	Symbiotic (firm gains–community gains)	**Commercialistic (firm gains–community indifferent)**	**Parasitic (firm gains–community loses)**
Possibility of influencing community	High	Low	None
Managerial challenge	Respect norms and values Obey licenses Resource consumption of developing community	Respect norms and values Obey licenses Getting acceptance of the community for using its resources in commercial applications	Avoiding direct conflicts
Operational means of subtle control	Attracting developers Aligning different interests Resolving ambiguity about control and ownership Devoting personnel to work in communities Creating and maintaining reputation Fringe benefits Interaction tools Selling development tasks	Devoting personnel to work in communities	

Copied from (Dahlander & Magnusson, 2005, p. 487)

Although the authors stress that the typologies should "[…] not be seen as distinctive categories, but rather as steps on a continuum […]", the case of Trolltech is obviously more in line with a symbiotic relationship than anything else (Dahlander & Magnusson, 2005, p. 487). For one, it is obvious that both the firm and the community gains from their relationship. Although not a proof in itself, this can be deduced from the long history of cooperation between Trolltech and the community, especially seen in the case of the KDE community. It is apparent that KDE would not exist today had it not been for Qt. In my interviews with the former Trolltech CEO and the present community manager, they both stressed that the success of Qt and Trolltech came from the emphasis on creating and maintaining a symbiotic relationship. From recent press releases it is clear that cultivating the community is articulated firm objective (Qt-Developement-Frameworks, 2011c). Also, from the previously mentioned case where Trolltech experienced difficulties with the community over licenses, the historical development shows that Trolltech has gone through great lengths to respect the norms and the values of the community, and to keep relations good (Fremy, 2003).

From this I conclude that Trolltech's work in maintaining a relationship in order to reap innovation benefits from freelance software developers will be based on a perspective of the relationship being symbiotic. The case of Trolltech therefore offers insight into what aspects of the development of both firm and community may jeopardize the relationship.

Trolltech and Qt has gone through several changes in both firm structure and growth throughout the years. By focusing on these developments, I expect to find answers to my research question.

The answers to these questions are thought to offer more detail into the managerial challenges identified by Dahlander and Magnusson (2005). But it will also look for factors that not necessarily can be influenced by management. It is thought that by lifting the perspective up a level, more factors can be identified and explained. The idea is to paint a picture of how innovation benefits, deriving from an OSS firm – OSS community relationship, can be accrued over time. The evolution of the community, the firm and the relationship is thought to highlight factors that may be specific for a given instance. By placing these factors in context it is hypothesized that pitfalls, such as the potential for conflict, can be identified. The issue of conflict is an understudied topic according to leading scholars in the field (West & Lakhani, 2008). It is also thought that the experiences of Trolltech with respect to the licensing debacle at the turn of the millennium, can offer additional insight to licensing issues. Previous research into OSS communities have identified norms and values of the community and licenses as very important in protecting OSS (S. O'Mahony, 2003). Trolltech's attempts at compromising through QPL and the KDE free Qt Foundation will be interesting to contrast with this research.

METHOD

In this chapter I will give an account of my choices for collecting data as well as my strategy for analysis of this data. In this paper I have chosen a case study of the Norwegian OSS firm Trolltech. The data sources are interviews with central people in Trolltech, observation and document analysis. Using this data I have answered the research question of **what enables an OSS firm to persistently reap innovation benefits from a global community of unpaid software developers.** This chapter will outline the methodological choices I have done, and offer my reflections about my choices. The intent is to make my proceedings as transparent as possible. This allows the reader to independently gauge the reliability and validity of my research.

Formulation of a research question

Punch quote Miles and Huberman in that "developing research questions is a valuable defense against the confusion and overload that are possible in the early stages of research" (Punch, 2005, p. 32). In my case, the anecdote of Trolltech mentioned in literature (Dahlander & Wallin, 2006; von Hippel, 2005; Weber, 2004) provided me early with both a case and a potential research question. Trolltech's ability to reap innovation benefits from the OSS community was endangered because of disagreements over licenses with the community (Dahlander & Wallin, 2006). However, observation of the community-firm relationship in 2010 showed that the relationship was very healthy. I therefore started with a research question that was something to the effect of "how does an OSS firm handle conflicts with its associated community". The thought was that the answer to this question would produce knowledge of conflict resolution between the two. This would implicitly say something about what enabled an OSS firm to be able to continue reaping innovation benefits from the community. I ended up modifying my research question as I concurrently started collecting data and developed my methods. This meant that I followed the third of Punch's three approaches of research question formulation, where the researcher "cycles backwards and forwards between questions, methods and some initial data" (Punch, 2005, p. 32). The other two approaches are when the question is developed before the methods are aligned, or where there is a general approach to a topic before the questions and methods are developed (Punch, 2005, p. 32).

Choice of methodology

As mentioned already, previous review of Trolltech in literature provided me with a possible case to study. This case could be used to answer questions about how OSS firms were able to operate, which "had received less attention" in previous literature (Dahlander & Magnusson, 2005, pp. 481-482). Even at the start of formulating research questions, it became apparent that any variation of the resulting research question called for knowledge about how an OSS firm handles its relationship with its community. In order to attain an answer to this type of question, it was important to classify what *kind* of question it is. First of all, it asks for data about the world which is not in the form of numbers. According to Punch this is the definition of qualitative data (Punch, 2005, p. 28). Though, as some point out, the presence or absence of numbers is an insufficient indicator of whether quantitative or qualitative research is warranted (Silverman, 2010, p. 190). Secondly, the research question calls for answers to such questions such as *what* factors are involved and *how* something is solved. These types of questions are well suited for to answer using a qualitative approach (Silverman, 2010, pp. 117-134). After determining that the research question most likely called for qualitative methods, I found that a case study would suit my research question. This was because Trolltech was an interesting example in itself about how an OSS firm was able to continue relations with its community, even after a relatively substantial disagreement with the community. According to Silverman "each case will be something in which the researcher is interested" (Silverman, 2010, p. 138), which I found to be a good description of what the case of Trolltech meant for my research question. These descriptions led me to conclude that the case of Trolltech represented an intrinsic case study "where the study is undertaken because the researcher wants a better understanding of this particular case" (Punch, 2005, p. 144).

One important aspect with respect to intrinsic case studies is that the objective is to learn about that specific case, "not because by studying it we learn about other cases or about a general problem" (Stake, 1995, p. 3). A central tenet of qualitative research is to produce rich descriptions of a situation or a case. It is this description that provides us with a thorough understanding of a complex process or case (Creswell, 2007, p. 40). However, producing descriptions solely for descriptions sake is a poor usage of the data gathered in a case study (Silverman, 2010, p. 140). According to Silverman and Mason, "Qualitative research should produce explanations which are *generalizable* in some way, or which have a wider resonance" (Silverman, 2010, p. 140). But with respect to the attribute of generalization, case studies cannot rely on the same statistical methods used in quantitative research (Silverman, 2010, p.

139). The reason is that the single case represents only one sample, and this sample is not randomly selected (Silverman, 2010, p. 139). In this paper generalization is accounted for by using what Silverman refers to as *purposive sampling*, where the researcher "seek out groups, settings and individuals where … the processes being studied are most likely to occur" (Silverman, 2010, p. 141). In the same manner, I sought out a case where I had a reasonable expectation of finding answers to what enabled Trolltech, an OSS firm, to persistently reap innovation benefits. The question of generalization then becomes a question about whether the findings in my case can be transferrable to other settings and contexts (Punch, 2005, p. 255). An indication that it is transferrable can be seen in my discussion where I argue that some of my findings fit well with previous literature on the subject. If these are similar, then the additional findings of this paper have a reasonable expectation of also being transferrable.

Data collection

Robert K. Yin lists six sources of evidence in case study research; documentation, archival records, interviews, direct observations, participant observation and physical artifacts (Yin, 2009, p. 102). In this paper data is collected using three collection approaches; 1) semi-structured interviews; 2) netnographic research (observation) and 3) documentation. According to Yin it is important to use multiple data sources because data gathered through the use of one method can corroborate data gathered through another method (Yin, 2009, pp. 115-116). This may increase the reliability of the single data collection method, and give an indication of whether the data represent the "true state of affairs" or not (Silverman, 2010, p. 133). The issue of reliability is also taken care of through using a tape-recorder and transcribing the interviews, and by producing notes during the netnographic research. By ensuring to document how the data was gathered and what the data was, reliability is improved. This is because another researcher can arrive at the same conclusions if using the same procedures as outlined here (Yin, 2009, p. 45). Qualitative research is also assessed according to the validity of it. External validity is a measure to which the findings of the research can be said able to generalize (Yin, 2009, p. 40). The discussion above about this case being transferable is a debate about the external validity of the research. A core concept with respect to external validity in qualitative case studies is that generalization or transferability is inferred from theory (Yin, 2009, p. 45). As such case studies such as this rely on analytical generalization (Yin, 2009, p. 45).

Initial data collection

In April 2010 I sent an email and called the community manager at Trolltech in order to assess the potential of negotiating access to key people in Trolltech. This process stretched over several months into autumn and included two meetings with the community manager. Trolltech was positive towards me using Qt as a case study, though the community manager expressed a wish that I chose to conduct a quantitative study in order to attain an answer to how many percent of their users were active contributors. Their best guess at the time was that around 10 percent of their users contributed, but they wanted an affirmation of this. The community manager was understanding and still helpful when I eventually chose not to pursue this potential study. Also, both meetings were very informative about OSS and OSS communities in general. Hence the result of these meetings was that I gained a lot of background information about the case and was able to secure an interview with the community manager to be conducted at a later point in time.

During the initial process I chose to adopt a netnographic approach since I figured I had a good prospect of finding relevant information in the venues available to me on the Internet. The information I hoped to find was two-pronged: 1) facts and background about Qt, Trolltech and the community and 2) information relating to specific qualities of the Qt community. The word netnography is a composite of the word "inter**net**" and "et**hnography**". Netnography is "a specialized form of ethnography adapted to the unique computer-mediated contingencies of today's world" (Kozinets, 2010, p. 10). Through participant-observation of conversations on online forums, the researcher arrives at the "ethnographic understanding and representation of a cultural or communal phenomenon" (Kozinets, 2010, p. 60). The purpose of this ethnographic understanding is to learn the values, behaviors, beliefs and language of a group sharing a culture (Creswell, 2007, p. 68). Hence, using this approach I would learn about the Qt community, and in the process I was more than likely to pick up a lot of facts about Qt and Trolltech. This knowledge would come in handy and be necessary should I use web surveys or interviews to gather data. It would also be valuable background information in other data gathering strategies too.

During this process, I observed discussions on the various online forums and blogs connected to Qt listed below. I chose not to engage in conversation with individuals on these web pages. The three reasons for this was; 1) I needed to learn about what people conversed about, how they did it, and also how interaction worked; 2) at the time I had no sufficiently concise research question to warrant a dialogue and 3) I did not want to participate in

discussions prematurely and potentially create a bad name for myself. Asking questions that may be perceived as irrelevant and bothersome questions could potentially hurt later interaction with members on the forums. I had reasonable fears that this might happen based on examples related by Kozinets (Kozinets, 2010, pp. 75-80). I could have chosen to not identify myself, but I reasoned that it would be unethical to not present myself using my real name. The reason for this was that members often used their full name or nicknames that could easily be traced to a full name through searches on Google.

table 4: Forums, blogs, mailing lists and IRC channels observed

Forum	Description	Hours spent	Time span
http://blog.qt.nokia.com/	Blog	43	2010-2011
http://developer.qt.nokia.com/forums	Official Qt forum	5	June 2010
http://forum.kde.org/viewforum.php?f=108	KDE forum	2	June 2010
http://www.qtcentre.org/forum.php	Unofficial Qt forum	2	June 2010
http://www.qtforum.org/index.html	Unofficial Qt forum	2	June 2010
#Qt related IRC channels	Official IRC channels	168 hours of logs	March 2011
http://lists.qt.nokia.com/mailman/listinfo	Official mailing lists	5	June 2010

In June 2010 I spent two weeks observing the forums, blogs and mailing lists listed in the table. I also spent time observing the same sources in irregular intervals during the following year. I kept track of my observation using two notebooks where I noted reflections, facts and time spent observing. The time spent on the forums is approximate, but the time spent on the blog and IRC have been logged. My assumption was that since the relationship between the Qt community and Trolltech seemed to be fairly open and direct, any discussion relating to my topic of interest could be found in one of these online venues. However, the venues listed in the table above contained thousands of different "threads" (individual discussions). The sheer number of different discussions presented me with a problem of too much information, since I initially had no well formulated selection criteria for choosing what discussions to observe. The reason I did not employ any selection criteria was because of the

explorative nature of my research in this phase. A positive outcome of this was that I discovered that some seemingly irrelevant forum threads turned out to be relevant after all. For example, the topic of the thread might be about something technical, but evolve into a discussion about the implications of the LGPL license. This fact meant that a refined method of selecting discussions by the topic of the conversation would exclude highly relevant discussions emerging from non-relevant topics. For further refinement I could have used the forums' search function to find relevant forum threads. This would have meant I needed to use keywords to identify what was relevant. However, at a practical level this approach was limited also, since the search results were still unrelated to my research questions. Additionally, considering the explorative nature of my research question, a preselected list of words could potentially lead to a skewed set of data, since I would not pick up on relevant discussions where community members used words not included in my list. As Donald Rumsfeld famously said, there are known unknowns and there are unknown unknowns (Rumsfeld, 2002). A preselected list of words would not be able to pick up on what was unknown to me. This would also be a challenge with respect to keeping an explorative approach. Also, committing to a netnographic approach would involve the practical necessity of selecting a few of these venues in which to concentrate the effort. This would introduce an element of selection bias in which I had to account for. On the other side, one could argue that members of the community are not bound to a specific venue, meaning that they frequent multiple venues. This would negate some of the selection bias, but it would still be leave me unknowing of the real effect of selection bias.

The perhaps best and easiest source online was the mailing lists, which at the time was available without the need for registration. However, here too most of the questions posed were technical. But in many ways it was easier to navigate around the mailing lists and exclude irrelevant discussions. However, I had never used mailing lists before, and I chose to first pursue venues that were familiar, such as browsing the forums, the blog and IRC. At the time when I felt that these venues were not ideal for my purposes, I started getting interviews with relevant people in Trolltech. As such, I never explored the possibility of using mailing lists. The reason for this was that I at the time experienced time constraints. I also felt that interviews provided me with the data I needed.

My experiences from observing the forums were that discussion was primarily related to technical questions. After a while when my research question had been refined, I chose to pursue a strategy of being more interactive by using Internet Relay Chat (IRC). A

characteristic of IRC is that all communication is instant, leaving the possibility open for more interactivity with community members. At the time when I started using IRC to monitor and log conversations, Nokia released some news that I felt certain would incite discussion relating to the topics I was interested in. The news was that the support division and the commercial sales licenses were being transferred to a Finnish company called Digia. Nokia's sale came few weeks after another big announcement in which Nokia announced it was going to depart from providing Nokia phones with Nokia supplied operating software, and instead use the Windows Phone 7 (WP7) operating system. Both announcements had implications the use of Qt in Nokia phones. Using WP7 meant that Qt was not going to be used in future Nokia phones, as was Nokia's purpose with buying Trolltech in the first place. The sale of commercial licensing to another company was the logical continuation of this decision, as the role of Qt in Nokia's future plans for handsets was practically non-existing. I chose to start monitoring all the listed IRC channels relating to Qt the same day Nokia sold commercial licensing to Digia. My reasoning was that if these venues were at all relevant for my research, it would be revealed in the aftermath of Nokia's announcement.

My experience proved otherwise. In short, the IRC channels seem to be mostly used for technical support in which community members ask and provide technical help to each other. I withheld from "flooding" random people with questions, which would more or less constitute a shotgun approach of firing into the flock, and hoping I'd hit something. The reason for this was firstly that I was using my own name, and did not want to create a bad image of myself, hurting any subsequent research. But the other factor was that I wanted to interact with specific users; lead users that I, on the basis of community meritocracy, could expect to be more central in formulating and knowing of the community sentiment I wanted understanding of. Therefore, I decided to log all IRC conversations in order to ascertain 1) which users discussed the topics relevant for me, and 2) which users were most active. The rational for the latter was that the most active users would have a far higher likelihood of being lead users (Belz & Baumbach, 2010). Eventually I made contact with two users on IRC that I identified as active. I introduced myself and asked an introductory question about their opinions the future of Qt after the recent events. I experienced responses that I had been warned of when talking to Trolltech's community manager. According to the manager the user community's style of communication could often come off as short, rude and perhaps a little impatient. The interaction I had with the two users on IRC resulted in one short but friendly suggestion that I consult the community blog. Another, perhaps a bit more rude

comment was that I should join the IRC channel #Qt-FUD.[7] In retrospect I realize that I could have pressed on more, and tried to ask more people. But at this point I was disillusioned about the prospect of gaining answers to my research question by using community forums and pages. At the same time I was also beginnings to get interviews with relevant people within Trolltech. It should be mentioned that the two interactions I had in the above mentioned were not logged. In retrospect I suspect that the only reason I remembered them was because I found the reference to #Qt-FUD funny. This goes to show the importance of being meticulous in keeping field notes.

Main data collection

Based on these experiences from the initial data collection, I concluded that I was in need of better data sources. Concurrently with initial data collection, I had refined and developed my research question. By "[cycling] backwards and forwards between questions, methods and some initial data", the research question turned to focus more on what the OSS firm did to enable reaping innovation benefits (Punch, 2005, p. 32). I continued with my netnographic research, at the same time as I focused on building a semi-structured interview guide. This guide was to be targeted towards key individuals with experience from Trolltech, and to whom could reasonably be expected to have information about innovation community *and* firm sentiment with regards to how conflicts appear and resolves (or not). A key question with this approach was whether or not I could gain access to these individuals. As such I focused on identifying the relevant individuals based on my netnographic research. Initially, I identified three people which I absolutely needed information from in order to cover the subject satisfactory. These were 1) the present Qt community manager at Trolltech, 2) one of the two founders and former CEO's of Trolltech, 3) someone who both had been employed by Trolltech *and* been a part of the community. The primary attribute of these three was that I reasoned they would all have information about the intersection of OSS firm and innovation community matters. The community manager would have such information by quality of his position at Trolltech. The founders of Trolltech would have information by quality of their long term development of Qt and its community. An additional source of information would be from someone who had made the transition between the actual OSS firm and the

[7] IRC includes thousands of different "channels", denoted by the sign #. They are normally named according to the subject that is discussed. In the case of #Qt-FUD, the "Qt" denotes that the subject is about Qt. The FUD in this instance refers to the abbreviation of Fear Uncertainty and Doubt (FUD). By referring me to #Qt-FUD (a non-existing channel) this particular user essentially declared my questions to be unwanted.

community. I later identified a Norwegian start-up called Cutehacks, consisting of two former Trolltech employees whom had started a business in which they were still a part of the Qt community. By using these three sources I would be able gather information about firm specific considerations and actions in the cases of conflict, as well as information from people whom could be assumed to have a thorough understanding of the community specific issues.

During a period of about 4 weeks in March and April 2011 was able to secure interviews with these three interview objects. As part of my research was explorative in nature, my semi-structured interviews would include an element of the "snowball-approach", where I asked the interviewee if he knew of any other people that I could interview. At the conclusion of the interviews I also asked the interviewed if they knew of other relevant individuals to interview. This approach turned out not to be successful. The drawback was that the relevant objects worked in Trolltech, and forwarded my inquiries to the Trolltech community manager. For example, the former CEO of Trolltech recommended I get in contact with a specific person in Trolltech development. The former CEO also gave me permission to use his name as a reference. Still, I was not able to secure an interview with this person. The Trolltech community manager explained the person had forwarded him the email I had sent. He explained that this particular developer was very busy at the moment and that he was not too interested in inquiries from students.

I ended up conducting 3 interviews using a semi-structured interview guide. I had 100 minutes with Cutehacks, 110 minutes with the former co-founder and CEO of Trolltech and 175 minutes with the community manager at Trolltech. Conducting only three interviews made me uneasy because it represented a small selection of people in which to base the analysis on. Three things eased my fears. Firstly, the interviewees were specifically selected to answer specific questions not too many others could answer. Secondly, the interviews conducted were long and in-depth. According to Silverman, the number of interviews required is determined by the research question (Silverman, 2010, pp. 192-193). My research question called for using purposive sampling where the interviewees had to have inside-information about actions Trolltech took with respect to its community. The former co-founder and CEO of Trolltech and the present community manager are highly relevant individuals in this respect. As such, I prioritized gaining rich descriptions from these individuals at the expense of representativeness. Thirdly, the information gathered from the interviews corroborated each other. There was no contradiction between them, and none of the interviewees presented any significant information in which the others did not do. From

this I had reason to expect that additional interviewees would only marginally improve the quality of the data, though it would improve the representativeness of my selection.
A final comment is that I could have interviewed more community members in order to provide a different point of view. However, I wanted to keep the focus on what managerial challenges Trolltech solved in order to reap innovation benefits. Community members without prior affiliation with Trolltech would not have been able to provide information about this.

The semi-structured interview guide was intentionally developed to focus on conflicts between Trolltech and the community. The reason for this emphasis was that 1) previous literature mentioned a conflict with the community; 2) conflicts highlight managerial decisions gone wrong and conflict resolution highlights managerial decisions gone right; 3) by focusing on conflict the interviewee is forced to answer and explain the situation around the conflict. The first interview with Cutehacks ended up being more of a focus group interview than a one-on-one interview. I had been in contact with one of the Cutehacks employees about the interview, but on arrival the day of the interview, I was surprised to see that the other member of Cutehacks wanted to participate too. The interview was characterized by a lot of discussion amongst the two employees in Cutehacks, which is why I argue it ended up being more of a focus group interview. Still, the information that emerged from this interview was valuable, since the discussion outlined some of the more sensitive issues in the OSS firm - OSS community relationship. The interview with the former CEO proceeded without any surprises. But the interview with the community manager was complicated by a clause in the guest-list registration that said all photos and audio recordings had to be clearer with a corporate communications officer. Luckily one was present and gave me permission within a couple of minutes. All interviewees gave me permission to tape the conversation and quote them as I saw necessary. I have omitted their names because they do not provide this paper with additional function, and because the electronic publication of this paper might be expose their names to a search engine.

The interviews were transcribed and sent back to the interviewees for them to check their statements. Unfortunately, this took longer than I had expected. I found that transcribing the interviews was especially difficult and time consuming. Most probably, this came from the fact that I am dyslexic. The almost 6 and a half hours of tape took me about 65 effective hours to transcribe, which meant that one of the interview transcripts took me 5 weeks to send back to the interviewee. It is hard to say if this was the reason no one sent back any

corrections. However, they were all informed that the resulting paper would be publicized, and no one asked to be anonymous. My own observation in this respect is that the interviewees were adamant about openness and free access to information. My interpretation of them is that they did not mind sharing the information. Mostly, I think that this was because they viewed Trolltech and Qt as a successful OSS firm and project, and wanted to share the experiences they had about it. That being said, I made some ethical considerations as to not using quotes that included personal characterizations of other people in the community. I reasoned that this could hurt the interview object, and it in most cases using such a quote would not add to my analysis.

Lastly, I used documents to help develop questions and attain background information about issues that were relevant. Some of these document sources are mentioned in the bibliography at the end. Still, there are many document sources that are not listed because they do not enter into my analysis or discussion. By using relevant keywords, Google was of great help in finding online information about Qt.

Data Analysis Strategy

In my data analysis I employed the general strategy of Huberman & Miles (1994) when working with the transcripts from my interviews. I embraced the advice of Agar which was to "read the transcripts in their entirety several times" (Agar, 1980, p. 103). In doing so I concurrently listened to the tape when reading the transcript. I found this to be both good advice, but also quite necessary considering the detailed descriptions given. In listening to the tape recordings I remembered, and were able to note, factors such as enthusiasm and intensity of the descriptions and explanations. Attentiveness to this enthusiasm and intensity of particularly one interview was highly beneficial in shaping a narrative in my own mind about how the OSS firm to OSS community relationship works. As a general observation, my interviews as well as my netnography left me with the impression that the individuals involved in OSS communities were all highly enthusiastic. They were especially enthusiastic about the intersection of innovative activities, innovation communities and firm-community cooperation. In retrospect this is perhaps not too surprising. From Eric von Hippel's explanations, the lead user is expected to have a high degree of intrinsic motivation about his or her work (von Hippel, 2005, pp. 134-146). It should not be surprising then that the lead user – either within the firm boundaries, as a result of recruitment from the community, or an

independent individual in the community – will have strong feelings and opinions about issues that affect this intrinsic motivation. On the other hand, it also clearly demonstrates the potential for conflict. Enthusiastic lead users will have the ability to shape the agenda of the firm-community relationship, as will be apparent in the sections below.

As a consequence of the impressions enthusiasm and intensity left on me during the analysis of my interviews, my approach towards the interview data can be called a realistic approach. According to Silverman, a realistic approach to interview data contains both elements "…of positivism (facts) and emotionalism (feelings)" (Silverman, 2010, p. 225). Although advocated by Huberman & Miles (1994), I have not paid much attention to the frequency a particular category has been mentioned by an individual interviewee. I have, however, used frequency of categories between the interviewees as a gauge of the given importance of this category. In this sense, frequency has been more used as marker for reliability through data triangulation between interview transcripts.

When coding, I chose to use categories that emerged from the transcribed interviews. This choice was initially taken long before I started my analysis; it was taken at the point where I designed my research, and decided upon a research strategy. The reason for this is as follows: Yin lists three types of case studies; exploratory, explanatory and descriptive (Yin, 2009, pp. 8-9). This is largely an explanatory case study, since it seeks explanations such as *how* Trolltech is able to reap innovation benefits. However, there are elements of exploration in this paper, since I seek to identify *what* enabled Trolltech to do so. As such, from the design phase of this research, I concluded that I wanted to keep a certain degree of openness as to avoid predefining categories. Again, as a consequence of this, I opted *not* to employ Yin's preferred analytic technique of pattern matching. Using this technique would have meant comparing empirically based patterns with predicted ones (Yin, 2009, pp. 136-141). I felt that efforts into producing *a priori* predicted patterns would have not taken full use of my data material. Since little previous research about the relationship that enables OSS firms to reap innovation benefits from OSS communities existed, I felt that it was more important to keep an element of exploration in my research. I have found a more compelling description of my case to be something in between an *instrumental* and a *intrinsic* case, as described by Stake (Stake, 1995, p. 3). The case of Trolltech is used in order to understand something more than Trolltech itself, namely the dynamic of the relationship between an OSS firm and OSS community. I found that the four forms of data analysis and interpretation advocated by Stake also would make better use of my data and be more befitting for the scope of my research

questions (Stake, 1995, pp. 71-90). Using Stake's four forms, I proceeded with; 1) categorical aggregation, where I hoped issue-relevant meanings would emerge. This would allow for using categories that emerged from the data material. 2) Direct interpretation, where I focused on drawing meaning from single instances, without comparing them to other instances. 3) I focused on finding patterns between categories. And lastly, 4) I focused on developing naturalistic generalizations.

As already stated, my general data analysis strategy follows Huberman & Miles (1994). The following table is taken from Creswell (Creswell, 2007, pp. 156-157). This table allows for a more readable and procedural listing of my data analysis strategy. It is a suitable table to include because the remainder of this chapter is heavily influenced by it, as is the subsequent chapter including the results and analysis. Note that Creswell includes the four forms advocated by Stake (1995).

table 5: Data analysis strategy

Data analysis and representation	
Data managing	Create and organize files for data
Reading, memoing	Read through text, make margin notes, form initial codes
Describing	Describe the case and its context
Classifying	Use categorical aggregation to establish themes or patterns
Interpreting	Use direct interpretation Develop naturalistic generalizations
Representing, visualizing	Present in-depth picture of the case (or cases) using narrative, tables, and figures.

Copied from (Creswell, 2007, pp. 156-157)

Description

This case follows the case of the OSS firm behind the computer software development product Qt. Two Norwegian programmers; Haavard Nord and Erik Chambre-Eng, started writing the software code for Qt in 1991. In 1994 they started the company Quasar Technologies, and subsequently changed the name of the company to Troll Tech, then Trolltech. In 2008 Nokia bought Trolltech, and changed the name to Qt Software, and later to Qt Development Frameworks. It was widely believed that Nokia bought Trolltech in order to

gain access to both the various Qt products, but also to the large base of Qt developers (Digman, 2008). The implementation of support for Qt on Nokia handsets would have meant that developers using Qt could produce "apps" for the wide variety of different Nokia handset models. At the time, Nokia was still the leading handset producer in the world. The availability of "apps" for Nokia handsets was intended to help increased sale of the handsets.

One important note here is that I am looking at a case which stretches over a considerable number of years. The initial instance of a conflict between the community and Trolltech was in 1998, when Miguel de Icaza started the GNOME-project in response to a fraction of the OSS community's objections to the licensing of Qt. However, the more explorative aspect of this paper – which is to identify what enables Trolltech to persistently reap innovation benefits – opens up for references during the whole lifespan of Qt. As such, the temporal restriction of my case was 20 years; from the initial creation of Qt in 1991, to present the present date of 2011. This means that the data material opens up for what Yin calls Time-Series analysis (Yin, 2009, p. 144). One major strength of this approach is the ability to trace changes over time (Yin, 2009, p. 145). Considering my dependent variable was innovative abilities of the community, and the main independent variable was the ability of the firm to reap innovation benefits from the community, a Time-Series analysis might have been a fruitful approach towards analysis of the data material. However, I observed that the incidences that lead to GNOME were a conflict that was time-limited. By this I mean that it did not take too long before it was resolved, thus it did not hinder for Trolltech's ability to reap innovation benefits. It also made sense that a state of conflict had to be somewhat limited in time; too long a conflict would arguably afflict the health of the firm-community relationship, and thus the basis for the symbiotic relationship. In the case of Trolltech, it was clear that the present relationship was healthy and amicable. However, a time-series analysis would have required generating a theoretically significant trend a priori, or at least used a rivaling trend in which to match the empirical trend. I felt that this would detract from the explorative elements of the research, which I felt was needed due to the limited previous research about the relationship between OSS firms and OSS communities. This is not to say that no theoretically significant trends could be formulated. For example, one could hypothesize that a in order to have a continuous growth in the user base of Qt, and thus the level of innovativeness, the OSS firm would have to gradually approach the community in matching their ideals. The ideals in which I am thinking of are the ideals of a fully open and free Qt, licensed in the standard FOSS licenses of GPL and LPGL. Based on previous

research, we already know that OSS communities adhere strongly to the ideals of free and open access to source code (S. O'Mahony, 2003). In doing so, the firm would remove an important source for conflict, thus enabling continued reaping of innovation benefits. In the case of Trolltech, this hypothesis could be proven true. And from this we could deduce that in order for an OSS firm to be able to persistently reap innovation benefits, two things need to happen: the OSS firm needed to be either bought up by a larger company for strategic purposes (as happened when Nokia bought Trolltech) or Trolltech needed to divert from being a commercial company, and become more of a non-profit project which relied on sponsors to finance development of Qt (in the same way the development of Apache does).

The reason I discarded this approach was because I felt that the data itself warranted a more detailed-focused approach. The focus on trends would certainly be interesting, but I would be discarding the possibility of fully using the insider-insight given to me by the interviewees. This insight left me with a good opportunity to document other aspects that affect a firm's ability to reap innovation benefits, apart from the issue of licenses.

Categorical Aggregation

In the first stage, I employed the advice of found in literature (Agar, 1980, p. 103; Creswell, 2007, p. 151; Miles & Huberman, 1994). I read through the transcripts, wrote memos in the margins, and highlighted issues that were relevant in the text. On beforehand, netnographic research had left me with an idea of what some relevant topics were. I then used my memory of this when I read through the transcripts. This meant that I had some level of data triangulation when going through the first interview transcript. On the subsequent interview transcripts, this data triangulation was strengthened, since all interviewees shared a lot of opinions and observations. However, this was an iterative process, meaning that I did not catch all the relevant topics on the first read-through. After the third read-through however, I could not find more topics in which to highlight.

The second stage of the coding process involved aggregating the topics and passages I had highlighted, and writing them down on large A3-sized sheets of paper. I did this for each interview transcript. On conclusion of this process, I started to circle the common topics each interview transcript had had. In practical terms, this meant finding a large table, and placing the sheets of paper in a manner in which I could see them all at the same time. This process was helped by the fact that I had used the same general structure on each interview. This

meant that there was a certain sense of chronological match between the three interviews, which in turn helped me structure the process of matching topics between them. For each match, I wrote down the topic on a separate sheet of paper. Some topics, however, did not match, some were irrelevant to my research questions, but others were highly relevant. In these cases, I wrote a memo in which I noted my reflections about the importance of the additional information this topic had.

The third stage involved a process of ordering the topics into groups that might be related. This process was both time and paper consuming. I iterated the process several times trying to order the topics in various ways, so their meaning matched with an encompassing category name. An introspective observation in this respect is the apparent difficulty I had with deciding which *grouping of topics* each topic should fall into. For example, the topic of "Licenses" fit into two distinct *groups* of topics, which – at the time – had no apparent connection. I was so focused on structuring what I had written down on paper, that I did not at first realize, that it was completely possible that the topic of "license" indeed *did* fit multiple places. The reason for this somewhat mundane oversight on my part can be found in the initial process of structuring my data material. For example, I observed that the license offered by the OSS firm might dependent on what kind of company structure it had, or fit into. At the same time, I found that the type of license offered by the OSS firm also functions as a form of communication with the community. However, in my work to structure and order my data, I only listed "Licenses" as *one* data point. As a consequence, when I started developing categories, "Licenses" became *one* category which I had problems with assigning into the appropriate theme. In retrospect I would have paid more attention to the advice of Huberman & Miles (1994), which is to keep tabs on the *frequency* of how often issues are mentioned. By doing so, ordering the data and structuring it into categories and themes would have been considerably easier. Additionally, this demonstrates the benefits of using computer assisted coding. As a consequence of using such coding software, frequency would have been accounted for in a much more automatically. The implication of this oversight was that I had to analyze my material one more time, this time accounting for frequency. This was one of the reasons for why this process became so time consuming. As a side note, I should mention that the reason I opted not to apply coding software to my analysis, was the drawback of spending time learning the software. After my discovery that frequency had been overlooked, significant time had passed already; hence I did not attempt to learn the software then either. In retrospect, these experiences show that the rational for computer aided coding is highly

relevant, and should have warranted a more thorough contemplation than it did at the initial stages of data analysis.

The result of my categorization, after the deficiency of my initial oversight of frequency was amended, was the categories in the next chapter. The categories are grouped into two main themes. One of the themes has two sub-themes.

RESULTS

To recapitulate on the research question: I have used the case of Trolltech and their product Qt in order to answer ***what enables an OSS software firm in persistently reaping innovation benefits from a global community of unpaid software developers***. Netnographic research and existing documentation has aided me in formulating a semi-structured interview guide which was used to interview co-founder and former CEO of Trolltech, community manager at Trolltech, and two former employees of Trolltech now self-employed developers in the Norwegian software firm Cutehacks. 6, 5 hours of interview data was collected.

The following section contains a listing and an analysis of the interviews. It is divided in two main sections according to each of the two themes identified; communication and structure respectively. Under each theme I will present what the data suggests are factors that impede or promote a software firm's ability to persistently reap innovation benefits from a global community of unpaid software developers over time. Within the first theme, "communication", I have grouped the categories into two groups, or sub-themes. Although these two groups emerged during the coding process, I have chosen to include them in my presentation of the data because they offer additional detail into the act of communication for the firm. The two groups highlight that communication is both verbal and non-verbal. The second theme identified is "structure", specifically the structure of the software firm, and the effect this has on which factors impede or promote the firm's ability to reap benefits of the community of software developers.

By presenting the results of the interviews in this manner, I intend to give the reader an opportunity to gauge the validity of my findings. It also puts "meat on the bone" in terms of telling a story about the factors that can impede an OSS software firm from accruing innovation benefits from global community of unpaid software developers. The data and quotes used are from the interview transcripts, and are translated from Norwegian to English by myself.

Theme nr. 1: Communication

Sub theme nr.1: non-verbal communication

Licenses

The issue of licenses is very prominent in the case of Trolltech and Qt. Because of this, it was treated at great lengths in the interviews too. This was to be expected since previous references to Trolltech in literature were about the licensing disagreements the firm had with its community (Dahlander & Wallin, 2006; von Hippel, 2005; Weber, 2004). Also, since Trolltech is a Nordic OSS firm, the managerial difficulties of Trolltech were expected to be the same as the ones found by Dahlander and Magnusson (2005, p. 487). In this paper they emphasize the importance of "[…] the social norms and values that are diffused across users and developers" (Dahlander & Magnusson, 2005, p. 489). Licenses serve as a legal tool to uphold these norms and values in the OSS community (S. O'Mahony, 2003). If a firm refuses to adhere to community preferred licenses, the case of Trolltech shows it would definitely jeopardize its relationship with the OSS community. This can be seen from the fact that GNOME was started as an alternative to KDE, because both an influential spokesperson in the OSS community (Richard Stallman) and others deemed the QPL license to be improper (Dahlander & Wallin, 2006; Icaza, 2005). One can therefore see adherence to OSS licenses as a way of adhering to community norms and values. This obviously has a signaling effect to the community, or as I have called it, non-verbal communication. However, Trolltech "dragged its feet" with respect to offering Qt under the GPL license. The reasons for this can be summarized in the points below.

1) Licensing under GPL introduced risks to Trolltech
2) Licensing under GPL could not be reversed
3) Licensing under GPL became necessary to preserve a good relationship with the community.

According to the former CEO of Trolltech, members in the KDE community approached Trolltech and explained that a lot of people in the OSS community did not much approve of the QPL license. Specifically, they were afraid that Trolltech would pull the QPL

license at some time in the future, which would undermine the whole KDE project. As the former CEO put it:

> *It was at this point we understood we had a real problem. (Former Trolltech CEO)*

As a response Trolltech created the KDE free Qt Foundation in 1998 which included people from the KDE project and the two co-founders of Trolltech (KDE, 2011). The foundation was given the right to license Qt under a BSD license should the firm behind Qt ever stop licensing Qt under an open source license, or if no new version had been released for more than a year. Under the terms of the BSD license (a very permissive license), this essentially meant that anyone, firm or individual developer, could take the last open source Qt and continue development of it. This is why the KDE free Qt Foundation is called the "poison pill" (Qt-Developement-Frameworks, 2008b). It is called so because Trolltech, or any future owner of Trolltech, cannot rescind the offering of Qt to the OSS community.

The KDE free Qt Foundation alleviated some of the pressure about licenses from parts of the OSS community. But from 1999, Trolltech still experienced a significant pressure to license Qt under GPL, according to the former CEO. The reason Trolltech hesitated was because they feared that someone could take the GPL licensed Linux version of Qt, and make a Windows version of it. Since the Windows version of Qt was where Trolltech made most of their money, a GPL version could have damaging effects on the firm. An additional factor to changing to a GPL license, as explained by the former CEO, was that Trolltech perceived such a move to be an irreversible one. Because GPL was a more permissive license than QPL, any future decision to ditch the GPL for a less permissive license (like the QPL) constitutes a serious *faux pas* with respect to the community. The following exchange explains why:

> *You don't tighten your license and leave the community hanging. It is a symbiotic relationship, so you need to give and take. If you try to interfere with the balance in the symbiotic relationship, for example by tightening the license, the [community] can't use their code any more. That's a serious no-no. You shouldn't do that. (Former CEO of Trolltech)*
>
> *It's the death of the firm? (Me)*

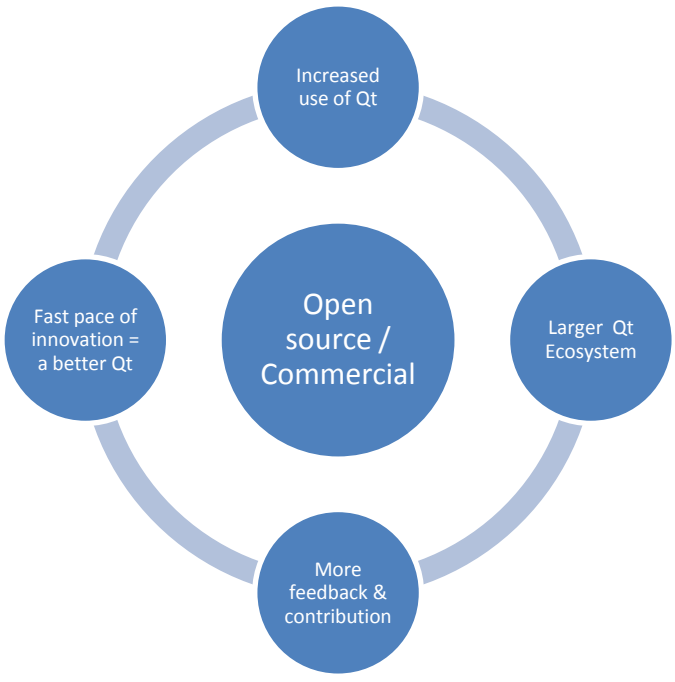
It might not be for the firm; maybe the firm can make money off it [a stricter license]. But you'll shoot yourself in the foot, at least if you intend to work with the community later. (Former CEO of Trolltech)

The interview with the former CEO of Trolltech revealed a surprising finding with respect to the internal debate in Trolltech about whether or not they should license Qt under the GPL. According to the former CEO, it was the internal developers that opposed GPL, while the sales people did not object to it. It was primarily one vocal developer that opposed GPL, on the grounds that it was a philosophically wrong license. The former CEO offered no additional details as to the specifics of why the GPL should be a philosophically wrong license, or the name of the developers. But seeing that this was the view of the developers, it is possible that their objection mirrors the philosophical division between the FSF and the OSI; according to the FSF, the key division between FSF and OSI are on the matter of values. The OSI views open source from a practical perspective, while the FSF views it as an ethical one. Or said in another way: "Open source is a development methodology; free software is a social movement" (GNU-Operating-System, 2011b). With QPL and the KDE free Qt Foundation, the practical aspect of open source Qt was in many ways taken care of. Hence, it is possible that the developers opposed the view of open source as prescribed by the FSF. If this is true, and since Trolltech in 2000 chose to "cave in" to pressure for a GPL license, it does say something of the normative power of the FSF. All though, this is an argument based on an assumption of what philosophical objections Trolltech's developers had; the conclusion would fit a tendency of Trolltech's relationship with the community being influenced by the FSF. Recall, the reason for the GNOME project was the objection to Qt not being GPL licensed, and the advice of FSF founder Richard Stallman was to avoid KDE on this basis (Dahlander & Magnusson, 2008; Icaza, 2005). Hence, in order to continue a working relationship with the community, Trolltech had to appease both the practically minded and the ethically minded parts of the community.

In an interview with community manager of Trolltech, I told him that it seemed that in order for Trolltech to maintain good terms in its relationship with the community; they had to "cave in" to community demands. For example, by eventually having to offer the community with a GPL licensed Qt version. The community manager disagreed with the view, on the assumption that it prescribed a view of position-based negotiations and not interest-based negotiations. He argued that the basis is to find win-win situations, not winning zero-sum

situations. The emphasis on finding win-win situations means that Trolltech's approach to the firm-community relationship is symbiotic, and not parasitic (Dahlander & Magnusson, 2005, p. 487). But, this would mean that both Trolltech and the community gained from Qt being licensed under GPL. By default, I am assuming the community gained, since the community demanded a GPL version of Qt, and got it. With respect to Trolltech, the transcripts mentioned no negative effects, nor have any negative information been found online. From the positive effects, the former CEO claims Trolltech continued to increase earnings after the addition of a GPL Qt. Hence, the negative effects they were afraid of did not come to fruition. In addition, according to a 2010 source, the Qt community had experienced 15 years of growth since 1995 (Qt-Developement-Frameworks, 2010, p. 8). The importance of community growth for Trolltech is best illustrated by the diagram of Trolltech's innovation strategy, or *virtuous cycle:*

figure 3: Trolltech virtuous cycle



Copied from (Qt-Developement-Frameworks, 2010, p. 7)

According to this diagram, a larger community leads to more feedback and contributions from the community. This again leads to an increased innovation pace, which leads to a better quality Qt, which leads to a more users and so on. With regards to the causation between

offering a GPL license, and an increased community, Trolltech community manager offered the following:

> *If what you want as a company, is to get a large community, then choosing a GPL license can be a lot easier to recruit with than other licenses. You also have the Apache-license, and they have a large community, but I don't think they measure up to GPL-projects, the Linux Kernel, KDE, GNOME (Trolltech community manager).*

With the above mentioned in mind, I find that an OSS firm in a symbiotic relationship with its community is better served reaping innovation benefits from its community with a GPL license than any other. The QPL license was in many ways not practically different than the GPL for many community members. And the KDE free Qt Foundation took care of community worries about future accessibility to a free version of Qt. However, the absence of a GPL licensed Qt still remained an issue with parts of the community. These parts of the community were not fringe enough to continue letting down, and subsequently, Trolltech did honor their wishes. In a manner of speaking, the GPL license has a signaling effect in addition to any legal effects. By this, I mean that it signals the firm's commitment to open source development.

The years 1998-2000, the time at which this license dispute emerged, was a time that coincided with a lot of institutional changes in open source and free software. The creation of the OSI in 1997 should be seen as a sign of overall community division over licensing, and thus norms and principles at that time. Trolltech ended up in the wake of this discussion, and it is obvious that it affected the Qt community, for example through the initiation of the GNOME project. The troubles of Trolltech in this process were weighing the perceived benefits and the perceived risk with a GPL. In Trolltech's case, it seems clear that they did not expect the demand for GPL to be as strong as it was. But when they eventually went for GPL, the effects were overall positive. One might expect that the same controversy would not be present, at least not to the same degree, at present time. The argument for this is that discussion over licenses can be expected to have settled over the years. At the same time, firms see that dual licensing using a GPL has been shown to not harm the firm. For example, firms like MySQL and Trolltech have been successful businesses for a while and have subsequently been bought up for large amounts of money. A sign of this can also be seen

when observing that the introduction of the LGPL license in 2009 did not bring with it the same amount of controversy as the QPL/GPL process did. On the other side, the introduction of the LGPL license was initiated by Trolltech and did not subtract from the GPL license. Rather, the LGPL allowed for commercial use of Qt without paying for a commercial license. But under the terms of the LGPL, a firm would have to share its any modifications of Qt back to Trolltech, should they have some. By bringing firms into the community of developers offering feedback and contributions in Trolltech's *virtuous cycle*, one can see that Trolltech, through LGPL, is leveraging increased feedback and contribution against cash flow. This slight change in innovation strategy increases the overall utility for all users of Qt, and does therefore only add to the utility of users of the GPL license. This move however, must be seen in the light of Trolltech now being part of the Nokia Corporation, where Qt functions more as a strategic tool than a business unit in its own right.

In summary, the choice of licenses affects how Trolltech was able to persistently acquire innovation benefits from the unpaid developers. In this respect, the LGPL increased the size of a contributing community. But, it is clear that LGPL would not have been possible had not Trolltech been a part of Nokia. Also, licenses can be seen as a way of communicating intentions towards the community, where half-measures such as the "poison pill" the KDE free Qt Foundation is not sufficient as a communicative tool.

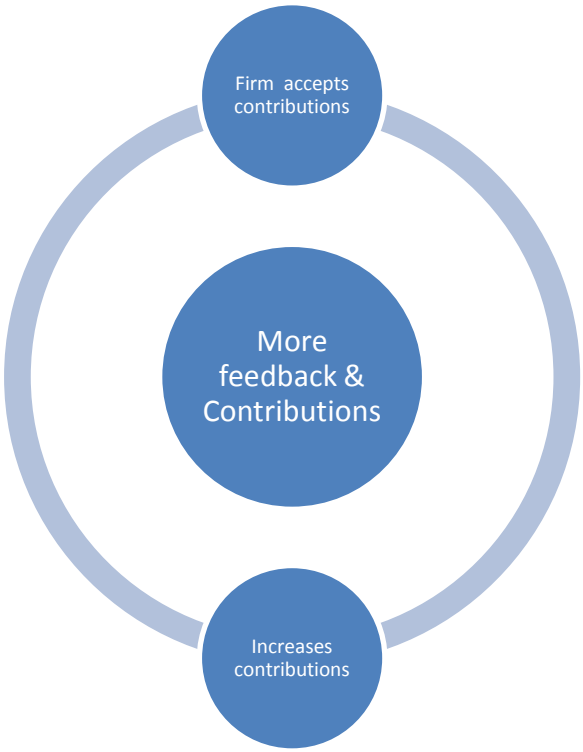
Feeding the community

All the interviewees emphasized the importance of keeping the community engaged and active. This was achieved in two interrelated ways; firstly, Trolltech needed to show willingness to accept both input in the form of code (either new functionality or bug fixes), but also respond to requests for additional functionality. Basically, Trolltech needed to accept input, and provide output of new code. A second feature was that the Trolltech also needed to provide the community with new versions that both incorporated the requested functionality and the provided functionality and fixes.

> *[…] in open source there is something called release early, release often;. You don't want to sit holding on until it is super-done and properly polished, and then release it. You want to release regularly. Show that things happen. What is stupid is to hold back. To hold back, to search for perfection, and to have one release a year, that is no good. In this manner you won't feed the symbiotic relationship. (Former CEO of Trolltech)*

The above mentioned actually represent a network effect within the *virtuous cycle* mentioned above. For example, by showing the community that their voice and work matters, the community is encouraged and continues providing feedback and contribution. The diagram below illustrates this effect.

figure 4: Network effect with respect to feedback and contributions



The community responds to new releases by feeding the firm with additional requests as well as new code, such as patches and bug-fixes. In turn, they expect some of these inputs to be included in the next release. The realization that their work is appreciated in turn makes them more motivated to contribute code back to the firm. This aspect of continuously accepting ideas (code, requests and fixes), and releasing better versions of Qt as fast as they can, embodies an important observation offered about the successes of the Linux Kernel development: "Release early. Release often. And listen to your customers" (Raymond, 2000, p. 7). Raymond explains that by doing this, Linus Thorvalds was cultivating co-developers. He "[…] was keeping his hacker/users constantly stimulated and rewarded – stimulated by the prospect of having an ego-satisfying piece of the action, rewarded by the sight of constant […] improvement in their work" (Raymond, 2000, p. 8). By assuring the community that their contribution matter, Trolltech managed to cultivate the foundation that made them does so in

the first place. Additionally, developers also provided feedback in the sense of bug-reports and bug-fixes (reporting and fixing errors). Hence, Trolltech benefitted not only by assuring developers would contribute, but also by the fact that developers provided quality checking. With respect to spotting "bugs", Raymond emphasizes, "all bugs are shallow given enough eyeballs" (Raymond, 2000, p. 8). Hence, there is an obvious benefit to having a lot of active contributors. By releasing new versions early and by releasing new versions often, Trolltech are able to benefit from this. Conversely, reacting too slow to community contributions has some negative effects:

> *If [the firm] does not fix the problem, and you're still sitting pulling your hair, then you're typically irritated. It is obvious it can be fixed with a little bit of code, and they won't listen to you. […] This means you won't bother taking the time to send bug reports and such. (Former CEO of Trolltech)*

An intriguing part of Trolltech's approach, with respect to Eric Raymond's development model of the Bazaar and the Cathedral, is the successful blending of both approaches by Trolltech. For example, Qt is still mainly developed *within* Trolltech (gitorious.org, 2011), but input effects of the Bazaar model (bug reports, fixes, larger fixes (patches)) can be harvested from the community. The fact that Trolltech has been able to maintain such dynamics over time can perhaps best be explained by the fact that the relationship between Trolltech and the community is *symbiotic*. As argued by Carr (2007), Raymond draws too much of a distinction between the Cathedral and the Bazaar (Carr, 2007, p. 3). Carr argues that without the Bazaar, the Cathedral moves too slowly, and without the Cathedral, the Bazaar is too unfocused (Carr, 2007, p. 3). This view coincides well with the views of the former CEO of Trolltech related to me during our interview. We can also see that there is a clear tendency going from a Cathedral model towards more of a Bazaar model, most notably with the introduction of the LGPL license, and the initiation of Open Governance. The LGPL opens for incorporating larger bits of code to Qt originating from other commercial firms, thereby opening up for more than just individual developers' contributions. Open Governance opens for a transfer of control of development out of the firm, and to the community. The reason for this movement could be interpreted in three different ways. 1) Qt is more a strategic tool for Nokia than it is a business opportunity. Or, 2) by moving towards Open Governance, development might be better and cheaper (for Nokia), since it would

leverage the resources in the community better (as seen from the greater use of the Bazaar model in Linux Kernel development). Or, 3) it could be a sign of the inherent problems of OSS firms with dual licensing and symbiotic relationships with its community. As Prowse puts it: "[…] the decline in dual licensing is being driven by both the inherent challenges of creating and maintaining a code base that is capable of being dual licensed as well as the increasing education and sophistication of end users with respect to OSS licensing" (Prowse, 2010, p. 1). The implication of this argument together with the information given by the interviewees is that there exists a limit to which a single firm can reap innovation benefits from the community. At some point the dual licensing model and leads to a plateau in the ability to both feed the community, and by implication also reap from the community. By moving to a more open business and license model, this problem is overcome. But the problem with this is that the disappearance of income from a more open licensing practice would leave traditional business unviable. It can therefore seem like the increased maturity of the underlying product affects a single firm's ability to feed the community, and therefore the ability to reap from it.

Product Quality

The issue of quality was mentioned frequently by all interviewees. They all claimed Qt was superior to the alternatives, and that this was what made it popular among developers. Taking into account the interviewees' background, it was perhaps not surprising they insisted on Qt being of very good quality. I was therefore a little apprehensive in my interpretations of the meaning of product quality. When engaged in research prior to the interviews I had observed that opinions about Qt were often quite polarized. Most often this was about licenses, but also about the quality difference between Qt and the alternative toolkit GTK. Typical examples of these discussions can be found on the webpage slashdot.com. I hypothesized that the interviewees probably would fall into the category of people that were vocal about the positive sides of Qt. Hence, taking claims by the interviewees about the superiority of Qt at face value was a bit hard. Still, regardless of the factual aspect of whether Qt is better or worse than an alternative, the data shows that product quality matters in reaping innovation benefits from the community. As illustrated by Trolltech's *virtuous cycle*, increased product quality leads to increased use of Qt. Hence, product quality is an important facilitator to gaining innovation benefits from the community. The explanation of why product quality matters was not at all complicated. Basically, developers will use any tool that

makes their job easier and more enjoyable. Qt is a well known toolkit that developers know will perform this function. If a significantly better alternative existed, developers would simply not use Qt. As Raymond emphasizes; if you have users, it can be taken as a sign that you are doing something right. Hence, the question is how you cultivate your users (Raymond, 2000, p. 6). Ensuring that product quality and functionality improves is one way of cultivating the users.

An additional aspect of product quality is the buffer effect it can have in a situation like the one Trolltech had with its community over licenses. According to Matthias Ettrich, founder of KDE, Qt was so much better than the alternative that the KDE community was willing to accept that it was not open source (Duval, 1998). The underlying incentive for the KDE community was producing as good a product as possible. Qt was the best alternative in achieving this. Although there was a lot of intra-community debate about the issue of licensing, it is fair to say that utility benefits from using Qt was able to counteract the normative views of what license should be used, at least for a while. As related by the former CEO of Trolltech, the subsequent decision by Trolltech to release QPL and the creation of the "poison pill" silenced most of this intra-community discussion. Eventually however, it became unavoidable to not license under GPL.

My interpretation of this is that the quality and utility of Qt gave Trolltech some extra time in the license debate. It seems the community is somewhat more flexible towards the firm if it considers the product to be better than the alternatives. Of course, this was not a sentiment shared by Miguel de Icaza and his followers at the time; if not, they would not have started the GNOME project. But it goes to show that the community discussion moved along two distinct lines; the practical emphasis and the ideological emphasis. Some will argue that one or the other is more important. But for the firm, it is clear that quality can in some instances weigh up for licensing issues, at least in the short term. The following exchange demonstrates my point:

> *One thing I am curious of: If you have a really good product, do you have more slack in your relationship with the community? (Me)*
>
> *The answer is yes. One example from our time: We had Qt, and then there was something called GTK. GTK had an extremely liberal license. […] A lot of people liked GTK because of the license. […] But our product was so much better, so a lot of the community members said: "GTK has a better license, but the product itself is crap, we can't use it. OK, Qt has some limits*

with its license. If I have to use it for a job I have to pay for it. But never mind, [Qt] is just so much better". (Former Trolltech CEO)

Recruitment from the community

The positive effects of recruiting employees from the community were a recurring issue. The primary effect it has seems to be in creating a relationship, or a channel for communicating, between the firm and the community. For example, the former Trolltech CEO suggests that a firm that is contemplating on starting an open source project might be wise to hire an open source community member and allow this person to "mull over the project". Acting almost as a consultant, he would then make considerations and suggestions to how or indeed *if* the project should be made open source. His role in this sense is practically as a go-between. He acts, according to the former Trolltech CEO, as a bridge between the corporate culture and the open source culture.

Trolltech hired many developers from the community. For example, Trolltech hired the founder of KDE, Matthias Ettrich, already in 1998. The fact that the firm hires extensively from the community lends weight to the theory that part of the reason unpaid developers contribute to an OSS project is because it might increase their value on the labor market (Lerner & Tirole, 2002, p. 213; von Hippel, 2005, p. 86). But hiring from the community can also have positive effects for the firm. For example, the firm can gain reputation itself by doing so (von Hippel & von Krogh, 2009, p. 17). However, the reason they recruit from the community seems to be more of a straight forward solution to finding qualified labor. According to the CEO:

Some firms, often smaller firms, that employ people from the community, like us, are dependent on a good relationship with the community. Because it is there you recruit people from. (Former Trolltech CEO)

Even if recruiting from the community is a cheap solution to finding human resources, it does have an auxiliary effect in the sense that it imports the norms and values of the community into the firm. The resulting employee values serve the firm in the sense that it incorporates both the firm's and the community's interests.

> *[…] many of the people who work for you do community related stuff on their own time. […] it is a gliding transition as to who community is and who is an employee. When a [new employee] gets inside the door, they often look at things differently, because they know they need a paycheck each week. So they might change views on some things, but they have a very good understanding of what the community feels about important questions. (Former Trolltech CEO)*

Hence, by using the community as a base for recruiting new employees, Trolltech laid the foundation to establishing a symbiotic relationship with its community. This happens through importing the norms of the community though hiring practices, but also through a mutual dependence for employment and labor.

On the other side, recruiting from the community has some intriguing parallels to Dahlander and Wallin's (2006) observation that some firms might sponsor, or pay for, developers on OSS projects. The reason firms do this is because they think that they need a "man on the inside" of the project in order to be able to get access to the community (Dahlander & Wallin, 2006). Similarly, recruiting Matthias Ettrich to Trolltech can be seen as an act with similar intentions. The distinction lies in that Ettrich was not hired to be "Trolltech's man on the inside of KDE", but to develop Qt itself (Bhartiya, 2008). Thus, this act lends more weight to the interpretation that the hiring from the community mainly happens because Trolltech saw developers in the community as highly qualified labor. Still, it would be naïve to dismiss the importance of hiring a central figure in the largest community of Qt users. According to the former CEO of Trolltech, each developer hired brings with them their own network within the community. So in a way Trolltech gained access to both qualified labor and a closer relationship with the community itself. The importance of having a relationship with the KDE community was apparent during the licensing disagreement in 1998-2000. For example, Trolltech and KDE were able to hold a dialog about what the community was afraid of, and what the community eventually demanded. Another hint of the close relationship is the KDE free Qt foundation, or the "poison pill".

My interpretation of Trolltech's heavy reliance on recruiting from the community is that it was primarily done because it was there dedicated and qualified developers could be found. However, by including community members in the firm, an added mutual understanding was created. In terms of reaping innovation benefits from a community, recruiting from the community therefore is a factor that enables communication across

different cultures. This communication facilitates further cooperation and therefore also further access to the community's resources.

The sub-theme of non-verbal communication

The four categories listed in this sub-theme are; 1) licenses (assuring the community), 2) feeding the community (engaging the community), 3) ensuring product quality and 4) recruiting from the community. They are listed together in this sub-theme because they all represent communication in some sense or form. This communication is non-verbal, meaning that they represent an act the firm does in order to communicate with the community. The categories are all facilitating factors to how Trolltech has been able to persistently reap innovation benefits from its community. By eventually issuing Qt under the GPL license, Trolltech communicated its reliance and dedication to the OSS community norms and principles. By feeding the community new releases of Qt in a timely manner, Trolltech stimulated the community to increase their contributions and suggestions for additional functionality. The result was a dialog based on actions that fostered increased innovation to which Trolltech and the community both benefitted. By ensuring that the quality of Qt got better, Trolltech kept its existing users, and welcomed new users. More users lead to increased community engagement, which again leads to a better quality Qt. In this respect, both the firm and the community benefit. By recruiting from the community Trolltech was able to cultivate a firm-culture that incorporated the interests of both the firm and the community.

Central to the outcome of all these categories was that they constituted communicative acts that facilitated the symbiotic relationship between community and firm. The emphasis on creating and maintaining a symbiotic relationship is a key factor to how the firm persistently is able to accrue innovation benefits from the community. These innovation benefits come to pass as network effects in which one effect amplifies another effect, which again amplifies another. The result is an environment in which the firm is able to – in cooperation with the community – reap innovation benefits.

Sub theme nr.2: verbal communication

The following sub-theme lists categories that are all descriptive of the verbal communication Trolltech has used in managing its relationship with its community. If the non-verbal communication treated in the first sub-theme can be seen as what lays the

foundation for a relationship that enables Trolltech to reap innovation benefits, the second sub-theme is more about the day-to-day management of the relationship.

Engineers

The interviews revealed that engineers should play a central part in communicating with the community. All four interviewees were adamant about the importance of this. Considering that they all were software engineers, this is perhaps not surprising. However, developers in the community can be expected to in large part to be software engineers too. Hence, it can be assumed that developers in the community would have the same preferences. One group of people the interviewees had no appreciation for whatsoever, was marketing people. If the preferred communication style could be type-casted, it is apparent that marketing people and engineers are at the opposite ends of the spectrum of desirability to a software engineer; both to an engineer within Trolltech, and to an engineer in the community. As an example of a poor way of communicating, Cutehacks offered the following example from a presentation at Desktop Summit 2009:

> *[…] he drew the UI [user interface] on a board and explained how it worked. […] There was no code, there were no screenshots, there were no devices, there was only a guy up there drawing bad […]. (Cutehacks)*

The objection to this manner of communication style seems to focus on it being irrelevant to an engineer. In illustrating the irrelevancy, Cutehacks compared it to a hypothetical talk about "The new BMW M5", were the presentation is limited to explaining that "the new BMW has […] four wheels, and a chassis and […] a steering wheel" (quote Cutehacks). Cutehacks explained that these things were not what interested an engineer. The engineer wanted to know the specifics of the engine, number of horse power, and so forth. Trolltech's communication with the community seems to have embraced that engineers communicate best with other engineers. With respect to the process of communicating "broadcast information" (general news) to the community, the former CEO explained that the person doing this needed to have a good understanding of the prevailing culture. They were either engineers themselves, or at least familiar with the preferred engineers' style of

communication. “In this manner they don’t act as an elephant in a porcelain shop. Not like glossy marketers at least” (Former Trolltech CEO).

From my own initial data collection I observed myself that community talk is strongly influenced by the mentality and style related to me at a later point by the interviewees. For example, conversation on the various Qt IRC channels is almost exclusively technically oriented. My own attempts to engage in non-technical conversation on these channels were politely (and impolitely) rejected. In internet vernacular, I was “off topic”. In common parlance, it would be the same as joining a conversation about cars at the water cooler, and trying to engage the others in a conversation about politics. Basically, what I was saying and asking about was perceived as irrelevant in these channels.

Honesty

Closely related to engineer-to-engineer communication is the issue of honesty. For the most part, the function of being honest with the community is to maintain credibility. For example, by continuously lying or being mischievous towards the community, the end result may be that the community stops believing in what the firm says. The logic is pretty similar to the phrase “Fool me once, shame on you, Fool me twice, shame on me”. Honesty can also have a mitigating function in the sense that the community is able to accept transgressions by the firm if presented with an honest explanation. Cutehacks mentioned a case where the source code for an OSS project called Meego was temporarily closed by Nokia. Essentially, this meant that Nokia took previously accessible source code and made it temporarily inaccessible.

> *This is something that had the potential of exploding, if it was not handled properly and announced by an unknown person […]. It is a typical thing that is not easily accepted in the open source community (Cutehacks).*

Essentially, the decision to close the source code was very similar to the act of “rolling back” licenses. In the previous sub-theme concerning licenses, the former CEO explained that “rolling back” was a serious *faux pas* against the community. The reason the incident did not amount to serious reactions from the community was in part because it was temporary, but

also because of the honest explanation to why Nokia felt it had to temporarily close the source code. The reason given was that there was an upcoming change to the source code in connection with a new mobile handset being launched. Nokia wanted to keep these changes secret until the new product was launched. The community seemed to accept this reason. As Cutehacks explained it:

> *What I think was done right was that they were open about it. They were honest, and that is something people respect […]. Then [people] accept quite a lot actually (Cutehacks).*

Although the closing of Meego was not related to Qt, the issue of Meego was used to explain the importance of honesty between the firm and the community. Another related aspect of being honest is avoiding so-called PR-spins. According to the former CEO, the use of PR-spins is especially negative if it is used to cover something up. With respect to this, Trolltech community manager offered the following advice: "Don't even try it". One reason given by Cutehacks is that the community consists of relatively intelligent people that will see through "sugarcoating". In many ways the focus on honesty is closely coupled with the aforementioned category of engineer communication. The key objection to non-engineer communication was that the information communicated was mostly irrelevant. Trying to conceal relevant information through sugarcoating just seems like a sure-fire way of annoying the community.

Face to Face communication

The use of face-to-face communication between engineers in Trolltech, customers and community members is part of a conscious communication strategy. As emphasized by the community manager, the direct face-to-face communication provides a level of nuance to communication that blogs, screen casts and video casts can't provide. The former CEO emphasized that face-to-face communication was and still is an important facet of community communication. He mentioned the Developers Days in San Francisco and Munich as the most important venues in this respect. At these conferences customers, engineers at Trolltech and community members socialize and talks about what that projects they find interesting. They can meet the developer responsible for a specific project, or a specific subject within programming. In all, the former CEO said that these conferences gave very positive feedback.

According to Cutehacks, Trolltech was very good at face-to-face communication. They also said that it was far better to send the individual developer to these events than it was to send a middle-man. At the conferences this developer could invite community members to join the project. As pointed out by Cutehacks, the informal situation left room for situations where you could tell people to "come to the main office [in Oslo], grab a beer and tell about your project, and perhaps find some help" (Cutehacks). According to Cutehacks, developers in the community are most interested in speaking with developers at Trolltech, not with professional communicators that might be found in a corporate communications department.

Theme nr.2: Structure

From analyzing the transcript data it became apparent that the acquisition of Trolltech by Nokia had brought with it changes that affected what enabled Trolltech to reap innovation benefits from the community. The principal change came in the changed role of Qt itself. For example, as a single firm Trolltech's revenue was generated by selling commercial licenses. When Nokia bought Trolltech in 2008 it was because Nokia wanted access to a developers community to create applications for Nokia handsets (O'Brien, 2008). As such, the revenue from the sale of commercial licenses of Qt was not as important as cultivating the community to achieve this goal. One development in this regard was the offering of a LGPL license in 2009. The former CEO described this decision:

> *LGPL is lesser GPL as some say, but it also stands for library GPL. It was designed for libraries […], frameworks or a platform which you can use freely commercially and in open source. This is natural, because [Nokia] wants to get as many developers using Qt as possible, and they aren't interested in making $50 million or so. It's a lot of money for a small firm, but for Nokia it is peanuts. What [Nokia] wants is to sell handsets, and they want to create an ecosystem of developers making Qt applications. So they made that change […], [a change] we couldn't do at Trolltech because our foundation for income would have disappeared. (Former Trolltech CEO)*

The rationale for LGPL was further explained by Cutehacks:

> *Such as it is at the time, Qt isn't a product for Nokia. It is a development platform in which [Nokia] build products with and gets other firms to build products with. So for [Nokia] Qt*

didn't have any value as a product, to make money on Qt I think they never had any plans of. (Cutehacks)

With respect to reaping innovation benefits, we can therefore see that the LGPL helps by increasing the size of the "ecosystem" in Trolltech's virtuous cycle. As such, it is a facilitating factor to reaping innovation benefits, but one that in Trolltech's case could only be achieved when the role of Qt changed from being a business to a strategic tool. Still, the LGPL only helps if the additional developers in commercial firms also contribute code back to Trolltech, just as freelance developers contribute bug fixes and other contributions. According to the Trolltech community manager, LGPL allows commercial firms to use Qt as a part of any proprietary software they intend to sell. However, if they modify Qt in order to make their software, they also have to share these changes. In this manner, firms get to use Qt for free in proprietary software, but contribute to develop Qt by sharing modifications. As a continuation of this aspect Trolltech has also set up an Open Governance project lead by Trolltech employee Thiago Macieira. The purpose of this project is to create a development model for Qt based on meritocracy (Qt-Developement-Frameworks, 2011e). According to Cutehacks, the purpose is to create a model in which individual firms can, independently of Trolltech, commit changes to the source code of Qt. They offered the following explanation:

Let's assume a commercial customer creates a helicopter simulator. They spent millions of kroner [NOK] on this, and have a lot of modifications to the Qt source code. [With Open Governance] it will be easier to integrate this into Qt without having to wait for [Trolltech] to test the changes. By building up status, the commercial firm can do this directly. (Cutehacks)

Cutehacks goes on by explaining that the primary intention of Open Governance is to create ownership of Qt for commercial firms. They argue that large commercial actors might be skeptical of using Qt because they become reliable on another large commercial actor, like Nokia. As an example, they mention that when Trolltech was bought by Nokia, Motorola (a competitor of Nokia) announced they were not intending to continue using Qt on their mobile handsets. Hence, by creating ownership privileges for other commercial firms, Trolltech is able to bring in more users and potential developers, thereby increasing the possibility to reap innovation benefits. All in all, the Open Governance model is still an ongoing process, and at

the time of the interviews it had not come very far. On the topic of Open Governance, the former CEO explained that he thought it a double-edged sword in the sense that although it might create ownership for both individual developers and commercial firms, the net effect might not be positive. He felt that there would be too many opinions in an Open Governance model, and that this would slow down development. Hence, Open Governance might lead to a larger ecosystem but might also lead to a slower development pace. These two factors work in opposite directions according to Trolltech's virtuous cycle. Still, the general consensus was that the Nokia takeover facilitated for increased innovation of Qt.

The change in structure from a single firm to being part of a large corporation brought with it some negative effects. The key aspects are the ability to maintain credibility through being honest and forthcoming and communicating clearly so as to avoid speculation. After Trolltech became a part of Nokia these factors have become harder to achieve. Most of all this was because Nokia had more than one project involved with the open source community. For example, Nokia has community participation in several projects such as Qt, Maemo, Meego and Symbian. The problem with this is that any change to one of the projects can affect the expectations the community has for the other projects. Cutehacks argued that Nokia probably does not understand the ill-will they amass in the open source community when they start projects, recruit for the projects and then "kill" the project all too fast. What more is that this kind of behavior is also detrimental to the open source community itself. According to Cutehacks, a large firm can drown a project in resources and thus displace the original developers. So when the firm drops the project a little later, there are no voluntary developers left to continue the project. The result can be irreparable damage to that particular project's community.

Another danger is that Nokia's press releases can create significant uncertainty which may not be possible to counteract efficiently. An example mentioned by the community manager was when Nokia announced that they would drop their own Symbian platform and start making handsets with Microsoft Windows Phone 7 instead. This announcement had implications for the Qt community because application development for Windows Phone 7 would probably use Microsoft's .NET framework and not the Qt framework. Still, it was the manner in which this announcement broke that felt like a shock. Because of the sensitive nature of this kind of information with respect to the stock market, Nokia could not give Trolltech heads-up before the announcement. The result was that when the announcement broke, both Trolltech and the community were equally surprised.

DISCUSSION

The findings from the interviews have identified the following categories, structured under two main themes, with two sub-themes under the theme of communication.

table 6: Themes and categories identified

Communication		**Structure**
Verbal	*Non-verbal*	Single firm project
Engineers	Licenses	Corporation project
Honesty	Keeping product quality	
Face-to-face communication	Recruiting from the community	
	Feeding the community	

During the empirical analysis the theme of *communication* was chosen because it grouped together categories that described communicative actions of the firm with respect to the community. The theme of *firm structure* was an outlier in this respect because it obviously did not perform any action towards the community. However, change in *firm structure* did have implications to the aspect of communication. Specifically it had implications for what kind of licenses could be offered. As related by the former CEO, Trolltech would not have been able to offer the LGPL license without Nokia. For purposes of reaping innovation benefits, the LGPL license widened the basis for accepting contributions by including firms. An interesting subject for future research in this respect would be to see if this in any way affected the contributions from freelance software developers, for example through displacing the efforts by freelancers.

But *firm structure* also had implications for Trolltech's verbal communication with the community. Being honest with the community through news releases became more difficult because some news about the future of Qt was kept from Trolltech. For example, Nokia press releases about Microsoft's Windows Phone 7 introduced uncertainty about Qt and its future, potentially hurting the basis for Trolltech's *virtuous cycle* where more users lead to a larger ecosystem, which leads to more contributions, which leads to higher quality product. As explained by Cutehacks, Nokia had already "killed" the Symbian and Meego projects, which left community members uncertain if the same might happen to Qt.

With respect to the categories deriving from the analysis of the interview transcripts, a key observation is that the categories are heavily inter-related. As the example of the Nokia's press release about WP7 illustrates, Trolltech's ability to be perceived as honest was challenged. This again can affect the *virtuous cycle*, which again is a product of the two categories of keeping the product quality and feeding the community. Three points can be derived from this: 1) the effort to persistently reap innovation benefits from a global community of freelance developers relies on a complicated and inter-related management practice, 2) *firm structure* interferes with this management practice, 3) the differentiation of the categories into verbal and non-verbal might not utilize the data to the fullest extent.

In addressing the last point, I will be drawing on the fact that Trolltech represents an OSS firm. As such one should expect similarities between the findings here and that of the study of Dahlander and Magnusson (Dahlander & Magnusson, 2005). Of the four OSS firms in this study, Trolltech has a similar background and community relationship to that of MySQL. Both firms are in symbiotic relationships with their community, have used dual-licensing, have firm and community established forums for interaction, gives and gets code from the community, receives bug reports and arrange user conferences (Dahlander & Magnusson, 2005, p. 486). As a side note, both firms were acquired by larger corporation in 2008 to serve as a "synergy effect" in the corporation (MySQL, 2008; Trolltech, 2008). Listed in the study (Dahlander & Magnusson, 2005, pp. 489-490) are seven managerial challenges in community-related activities. By comparing the findings in this paper with the list by Dahlander and Magnusson, I am able to accentuate what has enabled Trolltech to persistently reap innovation benefits from its community by showing Trolltech's solutions to these challenges. The following table lists Dahlander & Magnusson's challenges with Trolltech's solution as well as the categories produced under my data analysis. I propose that in comparing my findings with previous findings, it is easier to identify similarities as well as potential new challenges.

table 7: Comparison of managerial challenges, Trolltech's solutions and categories identified

Managerial challenges in community-related activities		
Challenge	*Trolltech's solution*	*Category*
Respecting norms and values of the OSS community	Recruiting from community internalizes the norms and values of the community. It may also provide firm with influential people that can affect community opinion (Matthias Ettrich from KDE). Also, through organizing conferences in Munich and San Francisco, face-to-face social interaction between Trolltech employees and outside developers is achieved.	Recruiting from the community, face-to-face communication and honesty
Using licenses in a fruitful manner	Trolltech tried to avoid GPL through the QPL and KDE free Qt Foundation. It was not accepted by the community. Trolltech were eventually pressured into offering Qt under the GPL license. This process had a negative effect on Qt since some of the user base and developer base started the GNOME project which utilized the GTK development kit. After offering Qt under GPL, the community was appeased.	Licenses
Attracting users and developers	By feeding the community frequent updates and keeping product quality the virtuous cycle creates a network effect that attracts both users and developers.	Feeding the community and keeping product quality
Handling the resource consump-tion related to community development	Trolltech invests in sending engineers to social events like conferences in Munich and San Francisco. Trolltech also employs a community manager.	Engineers and face-to-face communication
Aligning different interests about the nature of the work	Trolltech relies on the symbiotic relationship to align interests of community with firm. Licenses are a key part of this process.	Licenses
Resolving ambiguity about control and ownership	Trolltech did the same as MySQL by dual licensing Qt. After 2000, the license of QPL was changed to GPL.	Licenses
Getting acceptance for using the community-developed software in commercial applications and avoiding direct conflicts	Not directly applicable since Trolltech is not using community developed software, but community contributed bug fixes. Also, depending on the definition used, Trolltech did end up in a direct conflict with the community. In this situation, continued cooperation depended on offering the community demanded license.	Licenses

From this comparison, we can see the solutions Trolltech used in order to solve the managerial problems (Dahlander & Magnusson, 2005). An alternative manner, in which we can further accentuate the solutions utilized by Trolltech, is by comparing the categories from

the data transcripts with what is previously mentioned in literature about them. The objective here is to see if the transcript data can offer additional knowledge about the function of these categories.

table 8: Comparison of literature and findings

Comparison of literature and findings		
Category	*Literature*	*Findings in this case*
Recruiting from the community	The firm can gain reputation (von Hippel & von Krogh, 2009, p. 17). Reaffirms impression that signaling competence may lead to employment (Lerner & Tirole, 2002, p. 213; von Hippel, 2005, p. 86).	According to the former Trolltech CEO, recruiting from the community provides the firm with qualified employees and it internalizes community norms and values in the firm. This may help keep the relationship symbiotic in the long run.
Licenses	Licenses form a central part of how a the community protects its norms and values (Lerner & Tirole, 2002; S. O'Mahony, 2003)	"Half measures" such as QPL and the KDE free Qt Foundation are not sufficient replacements for GPL.
Feeding the community	By releasing early and often developers are constantly stimulated and rewarded (Raymond, 2000, p. 8)	Forms and integral part of Trolltech's innovation strategy through Trolltech's virtuous cycle. The findings demonstrate a clear focus of the firm of integrating and sharing results in order to create a virtuous cycle (Dahlander & Magnusson, 2008)
Keeping product quality	At its most basic level, the product needs to have a desired functionality among users in order to attract users (Raymond, 2000)	Forms and integral part of Trolltech's innovation strategy through Trolltech's virtuous cycle. The findings demonstrate a clear focus of the firm of integrating and sharing results in order to create a virtuous cycle (Dahlander & Magnusson, 2008)
Face-to-face communication	Face to face communication can "help instill greater collegiality, trust and respect among members." (S. C. O'Mahony & Ferraro, 2004, p. 22) Can be seen as a subtle operational means for handling the relationship with the community (Dahlander & Magnusson, 2005, p. 490)	The same
Engineers for communication	Can be seen as a subtle operational means for handling the relationship with the community (Dahlander & Magnusson, 2005, p. 490)	Engineers prefer to converse with engineers. Conversely, they are annoyed by "marketing language".
Honesty	Can be seen as a subtle operational means for handling the relationship with the community (Dahlander & Magnusson, 2005, p. 490)	Being honest can be seen as a manner of maintaining credibility. Can also mitigate temporary transgressions to community norms and values, for example though giving an honest account of why it is necessary.

Through this comparison I find that recruiting from the community has additional meaning than what is previously mentioned in literature. With respect to being able to continuously reap innovation benefits from the community, this action helps by incorporating community values and norms into the firm. It also provides a small firm with a basis for finding qualified labor. As such, recruiting from the community represents a possible solution to the managerial challenge of respecting the norms and values of OSS communities previously identified (Dahlander & Magnusson, 2005, p. 489).

Licenses have been shown to be hugely important with respect to retaining access to the community. It has been found that in the case of Trolltech, attempts to find compromises through the "poison pill" and a self authored license (QPL) was not sufficient in levitating community concerns about intentions of the firm. Thusly, it seems that OSS licenses, such as the GPL, became the de facto standard license any OSS firm needs to be able to offer. Trolltech's experiences can therefore suggest that the managerial challenge of "using licensing in a fruitful manner" does not include attempts to provide the community with compromises such as the "poison pill" and the QPL (Dahlander & Magnusson, 2005, p. 489). From this case, it seems that a firm with the intention of persistently reaping innovation benefits, sooner or later is forced to offer a GPL license. In addition, issuing an LGPL license can increase the firm's ability to reap innovation benefits. But since LGPL is mostly relevant for firms, it is doubtful if it has any effect on the firm's ability to persistently reap innovation benefits from the *unpaid developers* in the community.

Feeding the community and keeping product quality are both integral parts of the virtuous cycle and the innovation strategy of Trolltech. Together they both explain Trolltech's solution to the managerial challenge of attracting users and developers. The key findings is that these two categories are heavily interlinked and dependent on the other, as explained by the virtuous cycle and previous literature such as the findings of Raymond (1999) about releasing early and often. This virtuous cycle is a key explanation to how Trolltech has been able to persistently reap innovation benefits. Dahlander and Magnusson (2005, p. 491) argue that these two categories represent "operational means for handling the relationship to communities". For example, through creating intellectually challenging assignments and frequently improving the software product, the firm is essentially selling development tasks to the community, which is a way of influencing the community (Dahlander & Magnusson, 2005, p. 491). My findings indicate that in relation to being able to persistently reap innovation benefits from a community, these two categories are much more than exercising

"subtle means of control" over the community (Dahlander & Magnusson, 2005, p. 490). They demonstrate focused efforts where the OSS firm uses *assimilation* of community contributions as a method of furthering its ability to persistently reap innovation (Dahlander & Magnusson, 2008)

The last three categories of face-to-face communication, using engineers for communication and ensuring to be honest are perhaps more related to what has been called to "the subtle means of control" (Dahlander & Magnusson, 2005, p. 490). The argument is that the firm has no direct control over the community, but that the firm may use more subtle manners of influencing the community (Dahlander & Magnusson, 2005, p. 490). The use of engineers matches well with devoting personnel to work with the community, which is the first mechanism of subtle control identified (Dahlander & Magnusson, 2005, p. 490). The importance of honesty is correlates well with the emphasis of creating and maintaining a reputation, since being trusted is a necessary prerequisite to a good reputation. In this respect the category of honesty matches well with the second mechanism identified (Dahlander & Magnusson, 2005, p. 490). Lastly, face-to-face communication matches well with "interaction tools", where the argument is that face-to-face communication can create social interactions where OSS firms have it easier to influence the community (Dahlander & Magnusson, 2005, p. 491). An interesting aspect is that these three categories were all in the verbal communication sub-theme found in the data analysis. This is perhaps not surprising, since communication by definition is a necessary aspect of community interaction. However, it does raise the question if not the general rule is that creating and maintaining verbal communication is in itself an operational means of control over the community. This hypothesis might be a valid starting point for further research because it might identify more mechanisms of control, which could have implications for managerial practices of OSS firms.

The acquisition of Trolltech by Nokia led to a change in *firm structure*. From the data we can see that this created opportunities in the sense that it made the LGPL license possible. This in turn led to an increase in the "ecosystem" around Qt by including firms, which according to Trolltech's virtuous cycle facilitates for increased contributions back to Trolltech. As a step in further increasing these contributions, Trolltech has initiated an Open Governance initiative. By doing so, Trolltech is essentially making a change to one of the managerial challenges found by Dahlander and Magnusson (2005, p. 490). Trolltech is not resolving the ambiguity of control and ownership; it is transferring it out of Trolltech, and into the community itself. However, this development is too recent to judge the effects of it, or if it

indeed ever will come to fruition. We see however that the role of the Qt itself has experienced a change from being a business to being a source for reaping synergies from other business models. One interpretation of this is that the associated value of an OSS projects like Qt is greater when used as input to other ventures. Considering that both MySQL and Trolltech were acquired by a much larger corporation for more or less the same reasons, this fact brings forward some interesting questions. For example, it raises the question if not *OSS software itself is a facilitating factor for reaping innovation benefits in firms in general.* This view would coincide well with reports from that more and more commercial software applications include open source software as part of the application (Gartner, 2008). Also, data shows that OSS firms utilizing dual licensing schemes have decreased from around 20% in 2008 to around 5% in 2010 (Prowse, 2010). It might be possible that in order to stay competitive, OSS firms need to increased penetration of their software so as to keep reaping innovation benefits from the innovation community. In this sense dual licensing might be a constraining factor to continued reaping of innovation benefits.

Being part of a corporation may solve the financing concerns dual licensing has been used to solve before. The effect is an increased "ecosystem" through more permissive licenses, such as the LGPL. However, in managing the community of developers, being part of a corporation presents some complexity problems. In this respect firm structure might be said to be an *additional managerial challenge* to those identified by Dahlander and Magnusson (2005, pp. 489-490). For example, in section above I identified "Honesty" as a subtle operational means for handling the relationship. The key function of honesty is to maintain credibility. Maintaining credibility is harder for Trolltech under Nokia than it was as a single firm. The sole reason for this is that bystanders can see that Trolltech itself has a limited say when it comes to Nokia's strategy. For example, when Nokia announced that future handsets would be based on Windows Phone 7; this decision had implications for the Qt community, since it leaves a lot of questions open about the future for Qt in Nokia. Even if Trolltech should communicate assurances to the community that were a hundred percent honest, there would still be left some doubt as to whether Trolltech's communication is a hundred percent accurate. Also, insecurity can enter the Qt community because of how Nokia has handled other OSS projects in the past. For example, Nokia has instituted and subsequently dumped the Meego, Symbian and Maemo projects, both of which recruited freelance developers from the OSS community. The managerial challenge is therefore in

handling the added uncertainty and credibility problems that comes with being a part of a corporation.

CONCLUSION

This thesis set out to explore ***what enables a software firm in persistently reaping innovation benefits from a global community of unpaid software developers***. The impetus of this research question came from reading literature where I discovered that the relationship between OSS firms and OSS communities were an understudied topic. There were several references in literature to a case which was used to illustrate how a relationship between an OSS firm, Trolltech, and an OSS community turned sour. In this case, the OSS firm's ability to reap innovation benefits from the community was therefore challenged. Specifically, this happened through the creation of a competing project, which diverted some of the community's efforts away from the OSS firm. At the time of reading about this case in literature, over 10 years had passed since these events had transpired. Upon investigation the specific OSS firm, I found that since its mention in innovation literature, things had fared well with both the firm and the community. The relationship seemed highly functional and interdependent. I therefore concluded that this case could serve as something in between an instrumental or intrinsic case study that could highlight what it was that enabled the OSS firm to continuously reap innovation benefits from the community members over many years.

After identifying my research interests in relation to existing literature, I proceeded with formulating a research strategy that would enable me to gain answers to my questions. I had at the time no definitive research question, but proceeded anyway by relying on a strategy of "[cycling] backwards and forwards between questions, methods and some initial data" (Punch, 2005, p. 32). I quickly determined that the type of questions I sought answers to called for a qualitative methodology. I felt that literature had already handed me a very specific case in which to study, hence a case study of the OSS firm Trolltech with interviews seemed like a pertinent method. Alongside the interviews I also used netnography of the forums relating to the case and relevant documents that I found. The additional data sources allowed for triangulation of evidence which had implications for constructing validity of my research.

The empirical findings are limited in the sense that it only concerns one single case. As such there are limitations to generalizing from this case in the sense generalizing is used in quantitative research. However, by employing analytic generalization and comparing my empirical findings with previous literature, I have demonstrated that my findings are transferrable to one other case. The fact that my findings are corroborated by previous findings, lends confidence to any additional findings from the same empirical data. Still, it

must be pointed out that the number of interviewees were few. Perhaps the most important limitation of the empirical findings is that it does not include the perspectives of the community members. Indeed the individual unpaid developer, in which the OSS firm is to reap innovation benefits from, has not been consulted in this research paper. They were not consulted because of time considerations and because of the direction the research took.

The answer to the research question of ***what enables a software firm in persistently reaping innovation benefits from a global community of unpaid software developers*** can be explained by how Trolltech managed the community/firm interface. By constantly engaging the community through updates and new versions in a fast pace, the community is kept active. The results is that the community contributes more and more, which feeds into a cycle where the quality of the product improves, more users are attracted because of this, which leads to more developers in the community contributing. Hence, the key reason the software firm is able to persistently reap innovation benefits, is by investing and paying attention to creating and maintaining a virtuous cycle. In relation to previous literature this finding broadens the importance of releasing early and often as remarked by Raymond (1999). Trolltech's virtuous cycle seems to have incorporated Raymond's remarks into a formulated innovation strategy that is heavily reliant on the community. Trolltech's approach therefore has a lot to offer other OSS firms seeking to reap innovation benefits from their respective communities of unpaid software developers. Trolltech's approach also confirms a tactic used by other OSS firms in assimilating contributions from OSS communities, which is to feed code back to the community (Dahlander & Magnusson, 2005).

This case has also confirmed some of the managerial challenges predicted by Dahlander & Magnusson (2005). The most crucial is the importance of choosing which license to use. In this respect is has been found that the OSS firm might as well choose to use the community preferred GPL license from the start. Attempted compromises with vehicles such as the KDE free Qt Foundation and creating a custom license was ineffectual and might be said to have damaged the prospect of reaping innovation benefits from the community. First and foremost because the result was the initiation of the GNOME project, which meant that community, resources (developers) were diverted away from Qt and towards GTK.

With respect to managerial challenges this case has introduced an additional challenge to those identified by Dahlander & Magnusson (2005). It has been shown that when an OSS firm is acquired by a corporation and become a small unit within this corporation, some difficulties are introduced into managing the community. The specific challenge is to maintain

credibility and be perceived as honest by the community. This was difficult in Trolltech's case because its parent organization Nokia previously had cancelled other OSS projects it had been involved in. In addition Nokia also enacted some changes to its core business that introduced uncertainty to the community about Nokia's future dedication to Qt. Although the causality between added uncertainty in the community and a possibility of less innovation benefits is uncertain, the interviewees were all clear on the issue that this was a problem. It would therefore present an interesting case for future research to analyze how uncertainty affects the actions and the *modus operandi* of the individual developer in the community. As explained above, the empirical data in this paper does not allow for any conclusions to this question on the count that no community developers apart from Cutehacks were interviewed. On a more general level, the fact that OSS firms like Trolltech and MySQL are bought by larger corporations is interesting because it enables a different licensing scheme by including the LGPL license. In relation to the ability to reap innovation benefits from unpaid developers in the community, it would be interesting to learn of how the innovative efforts of these developers are affected by the (possible) influx of firm contributions.

Another finding in this paper explains a possible method of addressing Dahlander & Magnusson's (2005) managerial challenge of respecting the norms and values of the OSS community. By recruiting a lot of employees from the community, Trolltech is able to internalize these norms and values. By doing so, one could argue that Trolltech positions itself so as to making any transgression of these norms and values hard because internal debate would identify them before they were (potentially) unwittingly carried out. Another aspect is that the recruited employees will bring with them their contacts within the community, and thus have the ability to influence community opinions.

Lastly, this paper has identified face-to-face communication, being honest and using engineers for communication with the community as what Dahlander & Magnusson (2005) call subtle means of control over the community. These categories coincide well with the findings in previous literature.

REFERENCES

Agar, M. H. (1980). *The Professional Stranger*. New York: Academic Press Inc. .

Allen, R. C. (1983). Collective invention. *Journal of Economic Behavior & Organization, 4*(1), 1-24.

Bhartiya, S. (2008). Matthias Ettrich: The KDE-Man! Retrieved 07.13.2011, 2011, from http://www.efytimes.com/e1/fullnews.asp?edid=25412

Blanchette, J., & Summerfield, M. (2006). *C++ GUI Programming with Qt 4*. Stoughton, Massachusetts: Prentice Hall PTR.

Bo Jeppesen, L., & Molin, M. (2003). Consumers as Co-developers: Learning and Innovation Outside the Firm. *Technology Analysis & Strategic Management, 15*(3), 363-383.

Carr, N. G. (2007). The Ignorance of the Crowds. *47*(Summer 2007). Retrieved from http://www.strategy-business.com/media/file/sb47_07204.pdf

Chesbrough, H. (2003). *Open Innovation*. Boston: Havard University School Press.

Creswell, J. W. (2007). *Qualitative Inquiry & Research Design*. Thousand Oaks, California: Sage publications Inc.

Dahlander, L., & Magnusson, M. (2008). How do Firms Make Use of Open Source Communities? [Article]. *Long Range Planning, 41*(6), 629-649.

Dahlander, L., & Magnusson, M. G. (2005). Relationships between open source software companies and communities: Observations from Nordic firms. [Article]. *Research Policy, 34*(4), 481-493.

Dahlander, L., & Wallin, M. W. (2006). A man on the inside: Unlocking communities as complementary assets. [Article]. *Research Policy, 35*(8), 1243-1259.

Dictionary.com. (2011). Defintion of "community". Retrieved 06.23.2011, 2011, from http://dictionary.reference.com/browse/community

Digman, L. (2008). Nokia buys Trolltech; Gets serious about going cross platform. Retrieved 05.08.2011, 2011, from http://www.zdnet.com/blog/btl/nokia-buys-trolltech-gets-serious-about-going-cross-platform/7747

Duval, G. (1998). Linux Center Interview: a few questions to... Matthias Ettrich. Retrieved 08.01.2011, 2011, from http://www.linux-center.org/articles/9809/interview.html

Ebert, C. (2007). Open Source Drives Innovation. *Software, IEEE, 24*(3), 105-109.

Fagerberg, J., Mowery, D. C., & Nelson, R. R. (Eds.). (2006). *The Oxford Handbook of Innovation*. New York: Oxford University Press.

Flowers, S. (2008). Harnessing the hackers: The emergence and exploitation of Outlaw Innovation. *Research Policy, 37*(2), 177-193.

Franke, N., & Hippel, E. v. (2003). Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software. *Research Policy, 32*(7), 1199-1215.

Franke, N., & Shah, S. (2003). How communities support innovative activities: an exploration of assistance and sharing among end-users. *Research Policy, 32*(1), 157-178.

Fremy, P. (2003). Interview: Trolltech's Eirik Eng and Matthias Ettrich. Retrieved 27.06.2011, 2011, from http://dot.kde.org/2004/04/12/interview-trolltechs-eirik-eng-and-matthias-ettrich

Gartner. (2008). Gartner Says as Number of Business Processes Using Open-Source Software Increases, Companies Must Adopt and Enforce an OSS Policy. Retrieved 07.20.2011, 2011, from http://www.gartner.com/it/page.jsp?id=801412

gitorious.org. (2011). Members in qt-developers. Retrieved 07.16.2011, 2011, from http://gitorious.org/+qt-developers/memberships

GNU-Operating-System. (2011a). Various Licenses and Comments about Them. Retrieved 27.06.2011, 2011, from http://www.gnu.org/licenses/license-list.html

GNU-Operating-System. (2011b). Why “Free Software” is better than “Open Source”. Retrieved 08.07.2011, 2011, from http://www.gnu.org/philosophy/free-software-for-freedom.html

Harhoff, D., Henkel, J., & von Hippel, E. (2003). Profiting from voluntary information spillovers: how users benefit by freely revealing their innovations. *Research Policy, 32*(10), 1753-1769.

Hayek, F. A. (1945). THE USE OF KNOWLEDGE IN SOCIETY. [Article]. *American Economic Review, 35*(4), 519-530.

Icaza, M. d. (2005). The Story of the GNOME project. Retrieved 27.06.2011, 2011, from http://primates.ximian.com/~miguel/gnome-history.html

Jan Fagerberg, David C. Mowery, & Nelson, R. R. (Eds.). (2006). *The Oxford Handbook of Innovation*. New York: Oxford University Press.

Jeppesen, L. B. (2005). User Toolkits for Innovation: Consumers Support Each Other. *Journal of Product Innovation Management, 22*(4), 347-362.

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

Jeppesen, L. B., & Molin, M. (2003). Consumers as Co-developers: Learning and Innovation Outside the Firm. *Technology Analysis & Strategic Management, 15*(3), 363-383.

Jørgenrud, M., & Ervland, P. (2010). Hva er egentlig Qt? Retrieved 23.06.2011, 2011, from http://www.digi.no/837724/hva-er-egentlig-qt

KDE. (2011). KDE Free Qt Foundation. Retrieved 07.07.2011, 2011, from http://www.kde.org/community/whatiskde/kdefreeqtfoundation.php

Klein, B. (Writer). (2003). Top Gear. In A. Wilman & C. Hale (Producer), *Season 3, Episode 6*. United Kingdom: BBC Worldwide.

Kogut, B., & Metiu, A. (2001). Open-Source Software Development and Distributed Innovation. *Oxford Review of Economic Policy, 17*(2), 248-264.

Kozinets, R. V. (2010). *Netnography: Doing Ethnographic Research Online*. London: Sage publications Ltd.

Lakhani, K. R., & Panetta, J. A. (2007). The Principles of Distributed Innovation. *Innovations: Technology, Governance, Globalization Summer, Vol. 2, No. 3, 2007*.

Lakhani, K. R., & Wolf, R. G. (2003). Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. *SSRN eLibrary*.

Lee, G. K., & Cole, R. E. (2003). From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development. *Organization Science, 14*(6), 633-649.

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

Lüthje, C., Herstatt, C., & von Hippel, E. (2005). User-innovators and "local" information: The case of mountain biking. *Research Policy, 34*(6), 951-965.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis*. London: Sage publications Ltd.

Moconews. (2008). Trolltech Acquisition Improves Application Availability On Nokia Handsets: Ovum Retrieved 08.01.2011, 2011, from http://moconews.net/article/419-trolltech-acquisition-improves-application-availability-on-nokia-handse/

MySQL. (2008). Sun to Acquire MySQL. Retrieved 15.07.2011, 2011, from http://www.mysql.com/news-and-events/sun-to-acquire-mysql.html

O'Brien, K. J. (2008). Nokia acquires Trolltech for $154 million. Retrieved 20.07.2011, 2011, from http://www.nytimes.com/2008/01/28/technology/28iht-nokia.4.9555656.html

O'Mahony, S. (2003). Guarding the commons: how community managed software projects protect their work. *Research Policy, 32*(7), 1179-1198.

O'Mahony, S. C., & Ferraro, F. (2004). Managing the Boundary of an 'Open' Project. *SSRN eLibrary*.

Open-Source-Initiative. (2011). Licenses by Name. Retrieved 27.06.2011, 2011, from http://www.opensource.org/licenses/alphabetical

Perens, B. (1999). Open Sources: Voices from the Open Source Revolution. In C. DiBona, S. Ockman & M. Stone (Eds.), The Open Source Definition Available from http://oreilly.com/catalog/opensources/book/toc.html

Prowse, T. (2010). What business models are currently used with open source software? *2010*, from http://www.osbr.ca/ojs/index.php/osbr/article/view/1157/1107

Punch, K. F. (2005). *Introduction to Social Research* (2 ed.). London: Sage Publications Ltd.

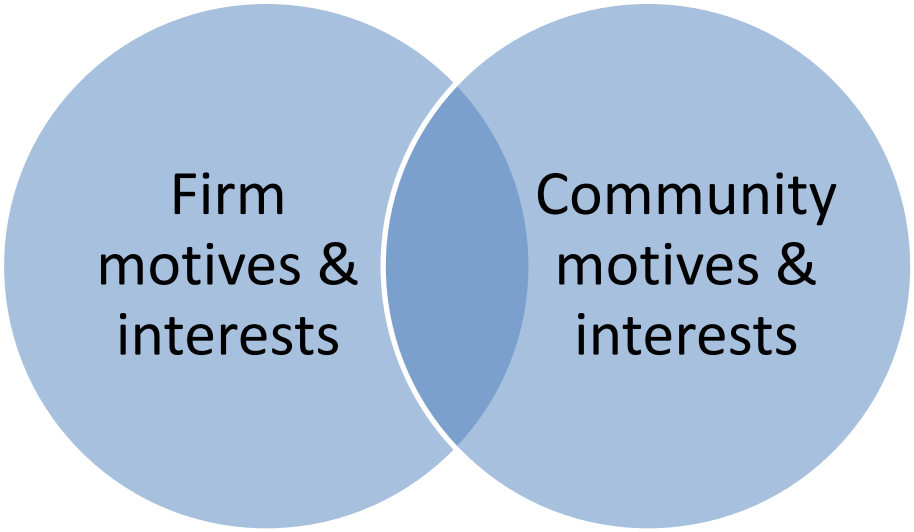
Qt-Developement-Frameworks. (2000). Trolltech offers a choice in licensing with the addition of GPL licensing for the upcoming release of Qt. Retrieved 06.29.2011, 2011, from http://qt.nokia.com/about/news/archive/00000043/

Qt-Developement-Frameworks. (2005). Qt Open Source Edition License Agreement. Retrieved 07.06.2011, 2011, from http://doc.qt.nokia.com/3.3/license.html
Qt-Developement-Frameworks. (2007). Trolltech Releases Qt 3.3.8. Retrieved 07.06.2011, 2011, from http://qt.nokia.com/about/news/archive/press.2007-01-22.4604809587/
Qt-Developement-Frameworks. (2008a). Nokia to acquire Trolltech to accelerate software strategy. Retrieved 08.01.2011, 2011, from http://qt.nokia.com/about/news/archive/press.2008-01-28.4605718236/
Qt-Developement-Frameworks. (2008b). Qt 3 and 4 licensed under GPLv3. Retrieved 07.07.2011, 2011, from http://labs.qt.nokia.com/2008/01/19/qt-3-and-4-licensed-under-gplv3/
Qt-Developement-Frameworks. (2009, 06.29.2011). Nokia to license Qt under LGPL. http://labs.qt.nokia.com/2009/01/14/nokia-to-license-qt-under-lgpl/
Qt-Developement-Frameworks. (2010). Qt & Qt in education. Retrieved 07.10.2011, 2011, from http://qt.nokia.com/partners/qt-in-education/files/qt-and-qt-in-education-program-presentation
Qt-Developement-Frameworks. (2011a). Contribute to Qt. Retrieved 06.29.2011, 2011, from http://developer.qt.nokia.com/contribute
Qt-Developement-Frameworks. (2011b). Frequently Asked Questions. Retrieved 06.29.2011, 2011, from http://qt.nokia.com/about/licensing/frequently-asked-questions#what-versions-of-qt
Qt-Developement-Frameworks. (2011c). Nokia and Digia working together to grow the Qt community. Retrieved 07.06.2011, 2011, from http://blog.qt.nokia.com/2011/03/07/nokia-and-digia-working-together/
Qt-Developement-Frameworks. (2011d, 06.23.2011). Online Communities. http://developer.qt.nokia.com/wiki/OnlineCommunities
Qt-Developement-Frameworks. (2011e). Qt and Open Governance. Retrieved 20.07.2011, 2011, from http://qt-labs.org/index.php/Main_Page
Raymond, E. (2000). The Cathedral and the Bazaar Available from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.120.4974&rep=rep1&type=pdf
Reichard, K. (2000). LinuxPlanet: Trolltech to Release Qt Under GPL. Retrieved 06.22.2011, 2011, from http://www.linuxtoday.com/news_story.php3?ltsn=2000-09-04-002-21-PS-BZ-KE
Rumsfeld, D. H. (2002). DoD News Briefing - Secretary Rumsfeld and Gen. Myers. Retrieved 07.29.2011, 2010, from http://www.defense.gov/transcripts/transcript.aspx?transcriptid=2636
Silverman, D. (2010). *Doing Qualitative Research* (3 ed.). London: Sage publications Ltd.
Stake, R. E. (1995). *The Art of Case Study Research*. Thousand Oaks, California: Sage Publications Inc.
Sweet, D. (2000). KDE 2.0 Development. In D. Sweet & e. al. (Eds.), 19.3. The License Usage by Qt Available from http://openbooks.sourceforge.net/books/kde20devel/index.html
Tidd, J., & Bessant, J. (2009). *Managing Innovation*. Chichester: John Wiley & Sons Ltd.
Trolltech. (2007, 06.23.2011). Trolltech delivered 42% revenue growth. http://qt.nokia.com/about/news/archive/press.2007-02-28.1193625148/
Trolltech. (2008). Nokia to acquire Trolltech. Retrieved 07.15.2011, 2011, from http://labs.qt.nokia.com/2008/01/28/nokia-to-acquire-trolltech/
von Hippel, E. (1976). The dominant role of users in the scientific instrument innovation process. *Research Policy, 5*(3), 212-239.
von Hippel, E. (1986). Lead Users: A Source of Novel Product Concepts. *Management Science, 32*(7), 791-805.
von Hippel, E. (1994). "Sticky Information" and the Locus of Problem Solving: Implications for Innovation. *Management Science, 40*(4), 429-439.
von Hippel, E. (2005). *Democratizing Innovation*. Cambridge, Massachusetts: The MIT Press.
von Hippel, E. (2007). Horizontal Innovation Networks by and for Users. *Industrial and Corporate Change, Vol. 16, Issue 2, pp. 293-315, 2007*.
von Hippel, E. (2009). Horizontal Innovation Networks - By and For Users. *SSRN eLibrary*.

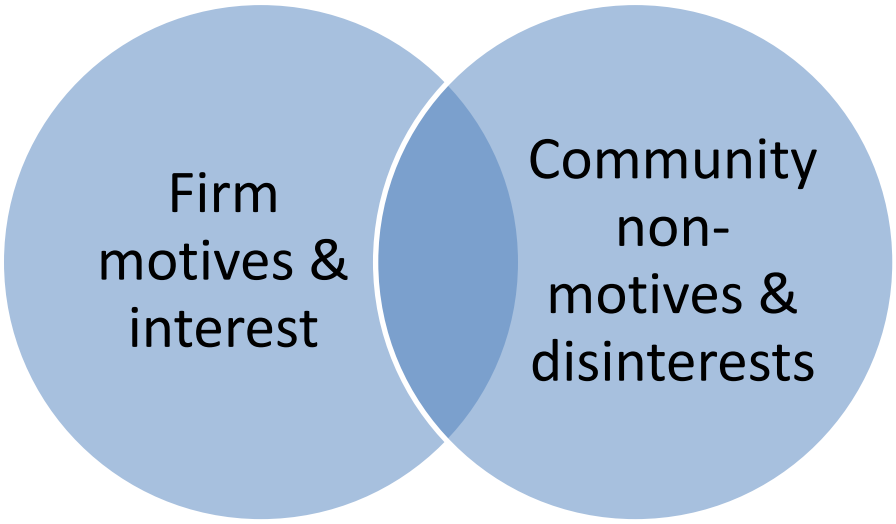
von Hippel, E. (2011). Collected Papers of Eric von Hippel. Retrieved 30.06.2011, 2011, from http://web.mit.edu/evhippel/www/papers/evh-01.htm
von Hippel, E., & Katz, R. (2002). Shifting Innovation to Users via Toolkits. *Management Science, 48*(7), 821-833.
Von Hippel, E., & Von Krogh, G. (2006). Free revealing and the private-collective model for innovation incentives. *R&D Management, 36*(3), 295-306.
von Hippel, E., & von Krogh, G. (2009). Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science. *SSRN eLibrary*.
Weber, S. (2004). *The Success of Open Source*: Harvard University Press.
West, J., & Lakhani, K. R. (2008). Getting Clear About Communities in Open Innovation. [Article]. *Industry and Innovation, 15*(2), 223-231.
Wikipedia. (2011a). Cross Platform. Retrieved 06.26.2011, 2011, from http://en.wikipedia.org/wiki/Cross-platform
Wikipedia. (2011b). Fork (software development). Retrieved 08.01.2011, 2011, from http://en.wikipedia.org/wiki/Fork_(software_development)
Wikipedia. (2011c). Q Public License. Retrieved 06.29.2011, 2011, from http://en.wikipedia.org/wiki/Q_Public_License
Williams, S. (2002). Free as in Freedom: Richard Stallman's Crusade for Free Software Available from http://www.faifzilla.org/toc.html
Yin, R. K. (2009). *Case Study Research: Design and Methods* (4 ed.). Thousand Oaks, California: Sage Inc.

APPENDIX: INTERVIEW GUIDE

It is understood that in general both the firm and the community profit from cooperation and working together. However, the firm and the community will always have divergent interests and motives in some cases. The core motives of the firm are well known, and recent studies into innovation communities have discovered some of the core interests and motives of individual community members. While the firm will always be influenced by a profit motive, the user innovator is more influenced by the act of innovating itself, and the status gaining aspect of sharing their innovation. In the case of FOSS communities, achieving a name for one's self within the community can also lead to job opportunities.



This Venn diagram illustrates the two different sets of motives and interests. The intersection of the two sets represents motives and interests that are shared by both the firm and the community. Within the intersection one can find the common ground held by both parties. An example of one such common goal is the desire to improve on a product of the firm. The firm profits from a better product in which to sell, and the lead users profit from gaining access to a product more in suit with their needs. However, in some cases the firm and the innovation community do not share interests. In some cases the appropriate Venn diagram would look like the one below.



Assuming the situations in which this Venn diagram illustrates do happen, the question arises of what these situations constitute, and how the firm approaches these situations. In short, I am looking for descriptions of what these kinds of situations are and how the firm adapts to them.

1) If these conflicts of interest happen, what community interests and motives are violated?

2) In general: how does the firm relate to the community in cases where their interests and motives conflict?

3) The assumption is that given enough conflicting interests and motives, the firm-community relationship will degrade. What is your take on this assumption?
    a. How would the relationship degrade? (Discussion in community → Fork/alternative →mass migration / on a continuum)
    b. Explain the process within the firm when decisions need to be made that can upset the community. (same during the life of the firm?)
    c. In what circumstances would the firm need to proceed regardless of potential community reactions?
    d. Providing that the firm can predict what the community will object to, how does this knowledge influence the decision process within the firm when it comes to whether or not they should go through with a decision or action?

4) Are there situations when the firm was surprised by a reaction within the community?
    a. If yes, please explain the situation(s)
    b. If no, what factors within the firm do you attribute this to? (honesty, engineers communicating with the community)

5) Are there any “faux pas” with respect to the community?

6) How important is the quality of the product in negating potential community reactions to unpopular decisions by the firm? (Are they more forgiving if the quality is good and not many alternatives exists, or does the firm have a sort of monopoly and can do as it pleases?)

7) Can you speak about the processes that lead to the present licensing structure?
    a. With respect to the process within the firm (how did you go about it when deciding on what to do)
    b. With respect to the community and its reactions
    c. How do you interpret the GNOME project?
    d. The given reason for the GNOME project was licensing, was this widely held (representative) feeling within the community?

8) How does the firm communicate with the community?
    a. Based on your experience, are there any best practices? (engineer-engineer, face-face, no bullshit artist/marketing, sugarcoating)
    b. How does firm size influence how the firm communicates with the community?

9) Several people I have interviewed point to community insecurity of firm intentions and strategy.
    a. What are the factors that lead to community insecurity about firm intentions?
    b. How do the size and the structure of the firm contribute to increased insecurity?
    c. When the business unit (Qt Software) responsible for the product (Qt) is part of a much larger organization (Nokia); how does this affect the work of managing the community? (different business units interfering)
    d. In managing a community, is there any optimal firm organization/type? (size of firm, strategy

10) What is the reason for the Open Governance initiative in your opinion? (commercialistic→symbiotic)
    a. Does it have any effect on lessening the potential for conflicting interests?
    b. If yes, then why?
    c. In your opinion, is Open Governance more of a “traditional” business decision than it was a community management one? Explain.

11) My purpose is to understand how conflicts between communities and firms arise, how they are solved or not solved, and how the firm views conflicts in its decision making process. In light of this, do you feel there are questions which I have overlooked?

12) Regarding this topic and the case of Qt and its community, do you know of any people that it would be helpful to interview?