

OSBR.CA

The Open Source Business Resource

Editorial

Chris McPhee and Michael Weiss

IT Cost Optimization Through Open Source Software

Mark VonFange and Dru Lavigne

Best Practices in Multi-Vendor Open Source Communities

Ian Skerrett

How Firms Relate to Open Source Communities

Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun

Private-Collective Innovation: Let There Be Knowledge

Ali Kousari and Chris Henselmans

Control in Open Source Software Development

Robert Poole

Nokia's Hybrid Business Model for Qt

John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim

Recent Reports

Upcoming Events

Contribute

JANUARY
2011



JANUARY 2011

PUBLISHER

The Open Source Business Resource is a monthly publication of the Talent First Network. Archives are available at: <http://www.osbr.ca>

EDITOR

Chris McPhee
chris.mcphee@osbr.ca

ISSN

1913-6102

ADVISORY BOARD

Chris Aniszczyk
EclipseSource
Tony Bailetti
Carleton University
Leslie Hawthorn
Oregon State University
Chris Hobbs
QNX Software Systems
Rikki Kite
LinuxPro Magazine
Thomas Kunz
Carleton University
Steven Muegge
Carleton University
Michael Weiss
Carleton University

© 2007 - 2011
Talent First Network

Editorial 3
Chris McPhee and Michael Weiss discuss the editorial theme: The Business of Open Source

IT Cost Optimization Through Open Source Software 5
Mark VonFange and Dru Lavigne analyze the use of using free/libre open source software as a cost-effective means for the modern enterprise to streamline its operations.

Best Practices in Multi-Vendor Open Source Communities 11
Ian Skerrett describes best practices for companies that wish to lower development costs and gain access to wider addressable markets through engagement with open source communities.

How Firms Relate to Open Source Communities 15
Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun explore the relationship between firms and open source communities. They illustrate the types of roles and strategies companies can employ to meet their business objectives.

Private-Collective Innovation: Let There Be Knowledge 22
Ali Kousari and Chris Henselmans describe the benefits and risks of the private-collection innovation model, in which an innovator may chose to collaborate with others while still keeping some intellectual property private.

Control in Open Source Software Development 27
Robert Poole examines the perceived loss of control in open source software development and argues that a better understanding of the mechanisms of control will help companies achieve their for-profit objectives using open source software.

Nokia's Hybrid Business Model for Qt 31
John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim describe the hybrid business model used by Nokia for its Qt cross-platform application and GUI framework.

Recent Reports 37

Upcoming Events 38

Contribute 40



Editorial

Chris McPhee and Michael Weiss

From the Editor-in-Chief

The editorial theme for this issue of the OSBR is The Business of Open Source. I am pleased to welcome as Guest Editor, Michael Weiss, Associate Professor in the Technology Innovation Management program at Carleton University.

The editorial theme for the upcoming February 2011 issue of the OSBR is Recent Research and submissions will be accepted up to January 15th.

For the March issue, we have invited authors from the Research Forum to Understand Business in Knowledge Society (<http://ebrf.fi>) to contribute to a special issue on Co-creation. The Guest Editors will be Stoyan Tanev from the University of Southern Denmark and Marko Seppä from the University of Jyväskylä.

For subsequent issues, we welcome general submissions on the topic of open source business or the growth of early-stage technology companies. Please contact me if you are interested in submitting an article (chris.mcphee@osbr.ca).

Chris McPhee

Editor-in-Chief

Chris McPhee is in the Technology Innovation Management program at Carleton University in Ottawa. Chris received his BScH and MSc degrees in Biology from Queen's University in Kingston, following which he worked in a variety of management, design, and content development roles on science education software projects in Canada and Scotland.

From the Guest Editor

An open source business is a business centered around an open source offer. Companies can engage with open source projects in different ways: they can release code as open source and hope to increase the adoption of their solution; they can contribute to community-initiated open source projects and leverage the solutions the community develops; they can offer complementary services and products that add value to an open source product; and they can reduce the cost and risk of product development by pooling their non-core efforts with other companies.

This issue contains six articles. The first two articles discuss cost reduction through open source, and best practices for multi-vendor open source communities. The remaining articles were contributed by graduate students in a class on Open Source Business in the Technology Innovation Management program at Carleton University in Ottawa (<http://www.carleton.ca/tim>). This course explored why companies participate in open source projects, how companies manage communities around their open source offers, and how companies make money from the open source projects they initiated or contribute to.

Mark VonFange, Professional Services Manager at iXsystems, and Dru Lavigne, Director of Community Development for the PC-BSD Project, provide evidence of the increasing use of open source solutions in enterprise IT infrastructures. With its advancements in availability, usability, functionality, choice, and power, free/libre open source software (F/LOSS) provides a cost-effect-

ive means for the modern enterprise to streamline its operations. The article quantifies the benefits associated with the use of open source software.

Ian Skerrett, Director of Marketing at the Eclipse Foundation, identifies five best practices for multi-vendor open source communities. Multi-vendor open source communities such as Eclipse, Apache, or Linux enable companies to lower development costs and gain access to wider addressable markets. The article also discusses the importance of foundations in implementing multi-vendor open source communities.

Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidoku explore the relationship between companies and open source communities. The article identifies the ways in which companies can participate in open source communities and how they can benefit from engaging with the community. It also asks how open a company should be.

Ali Kousari and Chris Henselmans weigh the benefits and risks of companies moving from a private to a private-collective innovation model. In this model, a company collaborates with other companies by making its project public and, in turn, may benefit from higher-quality, decreased time to market, and maximized revenue.

Robert Poole discusses how companies that rely on open source may fear losing control over the

execution of their product development strategy. Understanding the mechanisms of control inherent in open source projects and the benefits of hybrid approaches helps companies articulate those fears and make appropriate strategic decisions to match their business objectives.

John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim present Nokia's Qt product (initially developed by Trolltech) as an example of a hybrid business model. They illustrate how the hybrid approach was implemented and the extent to which the approach has been effective for Nokia. The Qt story illustrates how F/LOSS business models were developed during a period when participants were just beginning to understand how to make money with open source.

Michael Weiss

Guest Editor

Michael Weiss holds a faculty appointment in the Department of Systems and Computer Engineering at Carleton University, and is a member of the Technology Innovation Management program. His research interests include open source ecosystems, mashups/Web 2.0, business process modeling, social network analysis, and product architecture and design. Michael has published on the evolution of open source communities, licensing of open services, and the innovation in the mashup ecosystem.

IT Cost Optimization Through Open Source Software

Mark VonFange and Dru Lavigne

“Waste neither time nor money, but make the best use of both. Without industry and frugality, nothing will do, and with them everything.”

Benjamin Franklin

The cost of information technology (IT) as a percentage of overall operating and capital expenditures is growing as companies modernize their operations and as IT becomes an increasingly indispensable part of company resources. The price tag associated with IT infrastructure is a heavy one, and, in today's economy, companies need to look for ways to reduce overhead while maintaining quality operations and staying current with technology. With its advancements in availability, usability, functionality, choice, and power, free/libre open source software (F/LOSS) provides a cost-effective means for the modern enterprise to streamline its operations. iXsystems wanted to quantify the benefits associated with the use of open source software at their company headquarters. This article is the outgrowth of our internal analysis of using open source software instead of commercial software in all aspects of company operations.

The Coming of Age of Open Source Software

Until recently, purchasing proprietary (closed source) software was considered to be the de facto industry standard. Organizations need applications that are robust, mature, and supported, and building long-term relationships with software vendors was simply the best way to achieve this. As long as proprietary software led the field in market share, performance, and reliability, enterprises could not afford to take the risk of using alternatives that had low adoption rates, lacked professional support, and provided inferior technology.

However, much has changed in the world of IT in the past few years. As seen in Table 1, F/LOSS is now widely used and accepted for both infrastructure services and desktop applications. The reasons for widespread adoption include an overall increase in product reliability and per-

formance as well as superior security and cost. Enterprises have been integrating open source solutions into their IT infrastructure at an increasing rate and reaping many benefits.

Reliability and Performance

Over the past several years, open source projects have matured as adoption rates have increased. As companies adopt and integrate F/LOSS into their core technologies, they have the opportunity to drive development through participation in the associated open source communities. This creates a self-perpetuating cycle whereby increased adoption brings about improvements in open source technologies which, in turn, brings about even further adoption. As the market matures, it has become more desirable for businesses to offer professional support and services for open source applications. In the current landscape, businesses can choose F/LOSS applica-

IT Cost Optimization

Mark VonFange and Dru Lavigne

Table 1. F/LOSS Market Share as of August, 2010

F/LOSS Software	Market Share
Apache web server ¹	55%
Sendmail, Exim, and Postfix Internet mail servers ²	68%
BIND DNS server ³	79%
Firefox web browser ⁴	46%
OpenOffice office software suite ⁵	>1.2M downloads per week

1 <http://news.netcraft.com/archives/2010/08/11/august-2010-web-server-survey-4.html>

2 http://www.securityspace.com/s_survey/data/man.200907/mxsurvey.html

3 <http://ftp.isc.org/www/survey/reports/current/fpdns.txt>

4 http://www.w3schools.com/browsers/browsers_stats.asp

5 http://wiki.services.openoffice.org/wiki/Market_Share_Analysis

tions for their mission-critical operations because support infrastructures exist to provide commercial, enterprise-level support.

The very nature of F/LOSS development necessitates the use of solid development practices to manage the contributions of developers dispersed throughout the world. To give an example, the FreeBSD Project (www.freebsd.org) possesses a highly organized team of developers who direct code implementation. The Project manifests a conservative professional approach to software development (<http://tinyurl.com/nxxhkt>), a vetted codebase that has been under constant development for decades, and a product release cycle that focuses on code stability and quality assurance. This allows FreeBSD to run in an optimized manner for reliability and stability and is a great example of how F/LOSS projects can provide superior, professional-level technology.

The reliability advantage extends beyond operating systems. According to the Coverity Scan Open Source Report (<http://tinyurl.com/2bv2yox>), participating F/LOSS projects have reduced their static analysis defect density by 16% over the course of the last three years. The report has marked substantial increases in the number of projects that meet the qualifications of its three-rung system of evaluation. Also, open source projects tend to be more responsive to known software vulnerabilities. According to Veracode's State of Software Report (<http://tinyurl.com/29gxsav>), 36 days was the average turn-around time to fix a known defect in open source software, compared to 82 days for commercial software.

Not only has F/LOSS become more dependable, it is highly competitive against leading proprietary products in the area of performance. There have been many similar studies that demonstrate the performance advantages of open source distributions in head-to-head testing against proprietary distributions in a number of areas (<http://tinyurl.com/c8dnrh>; <http://tinyurl.com/39nsvtf>). These advantages extend beyond desktop applications. For example, nine of the top 10 most reliable Internet sites run an F/LOSS operating system, with four of the top five running FreeBSD (<http://tinyurl.com/35tzszl>).

Ready for the Desktop

Beyond operating systems and server applications, F/LOSS offers robust applications for personal and enterprise desktop operations. For example, OpenOffice offers nearly all of the features of Microsoft Office, along with ease of use and the ability to handle complex operations. OpenOffice can open any Microsoft Office document, operate with a similar interface, and use the same syntax for spreadsheet and database operations (<http://tinyurl.com/23lhoco>). In

IT Cost Optimization

Mark VonFange and Dru Lavigne

terms of usability, OpenOffice offers an interface that is as polished and intuitive as Microsoft Office, and even superior in some aspects. It offers comparable performance to its proprietary counterpart, support for all major operating systems, superior localization, and better support for Visual Basic macros (<http://tinyurl.com/2agr2c2>).

Open source email clients such as Mozilla Thunderbird (<http://mozilla.com/thunderbird>) and Zimbra (<http://zimbra.com>) offer all the functionality of Microsoft Outlook with the added benefit of built-in security and privacy measures. Thunderbird is well noted for its speed of operation and supports hundreds of add-ons¹ to customize the email experience. Zimbra extends its email features with a collaborative documentation management suite.

In the area of graphics, the imaging software GIMPshop (<http://gimpshop.com>) offers a powerful alternative to Adobe Photoshop. Scribus (<http://scribus.net>) is a Desktop Publishing utility with functionality similar to Adobe InDesign. These open source applications provide a graphical design team with all the core tools necessary to put together quality, professional publications.

Open source web browsers, such as Mozilla Firefox, often outperform their proprietary counterparts in speed and performance benchmarks (<http://tinyurl.com/yfctr8c>). Google's Chrome browser is based on the open source Chromium project (<http://chromium.org>).

There are many mature enterprise-oriented F/LOSS applications that can handle virtually every business-related function. PostgreSQL (<http://postgresql.org>) is a trusted and widely used database program. Samba (<http://samba.org>) and Apache (<http://apache.org>) are popular server applications that are used in enterprise as well as small and medium business

environments. There are several customer relationship management (CRM; <http://tinyurl.com/4wh5vc>) and enterprise resource planning (ERP; <http://tinyurl.com/qdhwu>) software offerings, such as SugarCRM and Support Suite, which offer a wide array of operational management capabilities and support packages.

Putting Open Source Into Play: A Case Study of iXsystems

As an outgrowth of BSDi (http://wikipedia.org/wiki/Berkeley_Software_Design), iXsystems (<http://ixsystems.com>) owes its very existence to F/LOSS. Since its inception in 2002, iXsystems has run all aspects of its operations exclusively on F/LOSS. Today, iXsystems has approximately 45 employees and runs 30 production servers as well as 40 to 80 testing servers. All of iXsystems' equipment runs either the FreeBSD server or PC-BSD desktop operating system, as well as a variety of open source applications for day-to-day operations. This has led to significant cost reductions in all areas of operation.

If iXsystems were to operate at its current level with proprietary software, it would have to purchase operating system and desktop application licenses for 45 desktop machines, 20 laptops, and 30 servers. In addition to licensing costs, there would be the costs of administration, support, maintenance, and periodic upgrades.

The first and most obvious measure of cost is the initial price to obtain all the applications necessary to operate. If iXsystems were to purchase all proprietary software, these expenses would quickly add up into the hundreds of thousands of US dollars. The initial costs would be over \$10,000 for desktop operating systems, \$34,000 for office suite applications, nearly \$140,000 for media and development software, \$1,800,000 for server software, \$23,000 for mail server software, \$69,000 for CRM and ERP applications, and \$180,000 for data archiving. As shown in Figure

IT Cost Optimization

Mark VonFange and Dru Lavigne

1, these costs add up to a grand total of over \$2 million in initial software licensing and support package costs. This is in stark contrast to the total payout of \$500 for all of the open source applications iXsystems uses in its daily operations. This allows for a significant reduction in the operational bottom line, which is certainly desirable for any organization, especially those just getting off the ground.

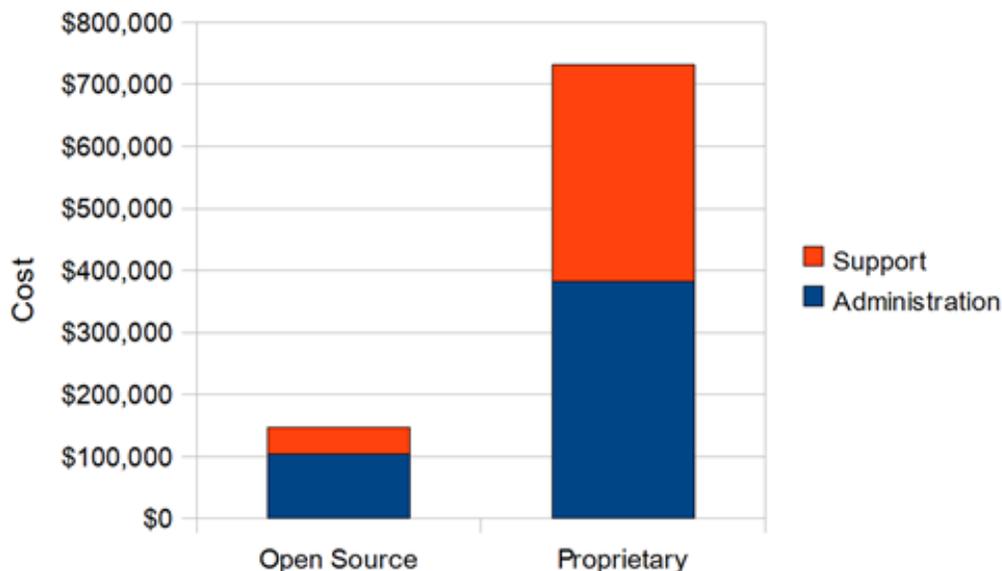
Beyond the initial cost of purchasing enterprise applications, companies must maintain and support their daily operations. This includes application support as well as administrative support costs. If iXsystems were to run on proprietary software, yearly external software support would run upwards of \$350,000. While iXsystems handles all of its own support internally, any company could obtain similar support for its open source applications for less than \$50,000 annually plus administrative costs.

At first glance, systems administration costs seem to favour the hiring of Windows administrators. In Silicon Valley, where iXsystems' base

of operations is located, the median salary for a Unix administrator is \$104,000 versus \$85,000 for a Windows systems administrator (<http://salary.com>). However, Unix system administrators are generally highly experienced and have a diverse skill set. A competent Unix administrator can support upwards of hundreds of desktop, server, and database systems. This is partly a result of the integrated design of Unix operating systems; once the base system is configured, the administrator can script many maintenance and administrative tasks. iXsystems manages to run a mail server, over 40 desktops, around 20 laptops, over 30 production servers, as well as many test servers with one experienced Unix administrator.

In a Windows environment, specific services are provided by different products, with each product requiring a discrete skill set. Due to the complexity of configuring and maintaining these products, it is rare outside of a very small network to find one administrator who is skilled in all of the products required in a typical business environment, for example, MS Exchange, SQL

Figure 1. Comparison of Initial Software Costs



IT Cost Optimization

Mark VonFange and Dru Lavigne

server, IIS, Sharepoint, domain controllers, print servers, and desktop support.

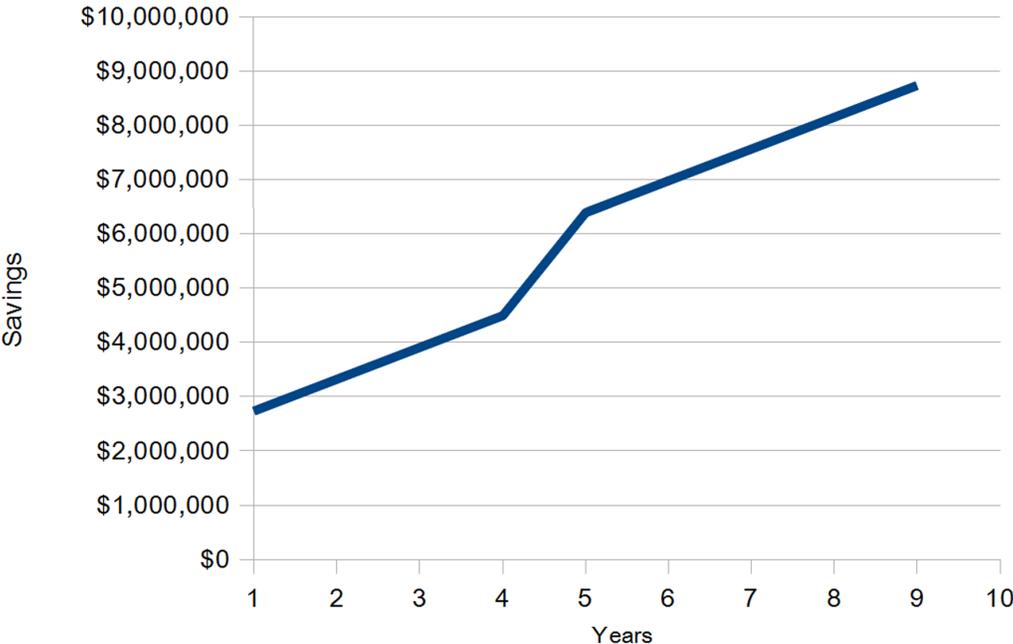
Due to these considerations, iXsystems determined that, if it were to run in a proprietary environment, it would need a dedicated database administrator, an MS Exchange administrator, a desktop support technician, and up to two general systems administrators. This would bring administrative support costs closer to \$400,000 for proprietary software versus a little over \$100,000 for open source. This gives a cost advantage of around \$300,000 per year to iXsystems for administrative support, freeing up further valuable resources for the core aspects of the business and lowering the bottom line.

The next consideration when evaluating software costs is the life cycle of the software. For most enterprise applications, a company will upgrade their software at least every five years, and some applications will have even shorter upgrade cycles. In our example, proprietary software would carry a cost of \$1.3 million every five years for software upgrades. With F/LOSS, the

cost of upgrading software to the most recent versions is negligible and can be considered part of the salary costs of the Unix system administrator. Using open source also carries the hidden advantage of being able to keep up with the latest improvements in software as they are released, rather than waiting until budgetary constraints allow. The end result is increased and improved application features sooner rather than later, without the increased costs.

With F/LOSS, the costs over the course of five years are around \$733,750 for a company with a setup comparable to iXsystems' current operations. If a similar company wanted to opt for proprietary software, they would have to spend approximately \$5.2 million over the same time period. That is five times the cost for the first five years of operation. Over the course of 10 years, a company comparable to iXsystems could save over \$8.7 million in software-related costs (Figure 2). This means that, in the short-run as well as the long-run, companies can save considerable amounts of money by utilizing F/LOSS.

Figure 2. Cumulative Yearly Savings Using F/LOSS



IT Cost Optimization

Mark VonFange and Dru Lavigne

Conclusion

Running operations utilizing open source applications has greatly contributed to the success of iXsystems as a business. The alternative costs to run licensed proprietary applications are simply too great to justify without substantial increases in quality and productivity. While in the past, proprietary software may have provided significant advantages in the areas of stability and support, those advantages can no longer be claimed in the current IT marketplace. Since open source solutions are competitive and mature in all areas of enterprise applications, iXsystems cannot justify such a substantial increase in company expenditures. As demonstrated by iXsystems' experience, open source solutions are now preferable to their proprietary counterparts.

Mark VonFange is the Professional Services Manager at iXsystems, providing oversight and coordination of its FreeBSD, PC-BSD, and FreeNAS support and development services. The Professional Services Team provides services ranging from mission critical support to software and firmware development to private consultation. Mark also develops internal and external documentation for division sales and marketing.

Dru Lavigne is the Director of Community Development for the PC-BSD Project where she leads the documentation team, assists new users, helps to find and fix bugs, and reaches out to members of the open source community to discover their needs. She is the former Managing Editor of the OSBR and author of BSD Hacks, The Best of FreeBSD Basics, and The Definitive Guide to PC-BSD.

This article is based on the whitepaper "IT Cost Optimization Through Open Source" (http://www.ixsystems.com/images/pdf/ixsystems_whitepaper.pdf).

Best Practices in Multi-Vendor Open Source Communities

Ian Skerrett

"If you love something, set it free. If it comes back to you, it's yours. If it doesn't, it never was."

Proverb

Multi-vendor open source communities enable companies to lower development costs and gain access to wider addressable markets. This article describes best practices for companies considering this approach. First, the different types of open source business strategies are examined along the types of participants that contribute to the communities that support them. Next, five best practices are detailed to show how companies can maximize their engagement with open source communities. Finally, the importance of foundations in implementing multi-vendor open source communities is discussed.

Introduction

Software companies are adopting open source strategies to drive lower development costs or expand an addressable market for their products. A key success factor for a company's open source strategy is how to engage with an open source community. It is often the community that will help provide resources to lower the development cost or create an expanded market of potential customers. Therefore, a company needs to understand what the strategies that will encourage a growing and engaged community.

The 451 Group (<http://tinyurl.com/2896jfc>) and Dirk Riehle (<http://tinyurl.com/28pmy3a>) have identified four types of corporate open source strategy and community engagement:

1. Traditional open source: these are the stereotypical open source projects started by a individual developer or group of developers working on interesting technology. Corporate involvement is limited.

2. Open source distributors: based on the success of early traditional open source projects, like Linux, companies began to create business around projects and contribute resources back to the core of these projects. Red Hat (<http://redhat.com>) is the most successful company using this strategy.

3. Single-vendor open source: software companies looking to disrupt an existing market will sometimes use an open source approach to change the market dynamics. These companies will establish an open source project and build a community around the project and technology. Then, they typically sell services and support or value-add products to enterprise customers. The company maintains strong control of the open source project and technology.

4. Multi-vendor open source: this type includes open source communities that involve multiple organizations working together on an open source project. The vendor collaboration typically focuses on areas that are not part of the participating organization's core asset.

Best Practices in Multi-Vendor Communities

Ian Skerrett

This article will focus on the best practices for creating a multi-vendor open source community. More and more companies are realizing that the multi-vendor open source community is the best option for achieving their business strategies. Having more companies involved provides incremental resources to work on the project and creates more incentive for companies to create a market around the technology.

Different Types of Community Participants

There are different types of participants in an open source community that help contribute to its health and success:

1. Users: these participants are individuals or corporations that make use of the technology for their own internal purposes. Typically, their motivation is to improve productivity.

2. Adopters: these participants add value to the project technology. Companies will often incorporate open source technology into commercial products or build services and components that work on top of the open source technology.

3. Contributors/committers: these participants are individuals and companies that actively work to develop and advanced the project technology. This type of participant helps lower the development costs of the project. Typically a subset of users and adopters form the core pool of committers and contributors. A strong community of users and adopters is essential for a strong contributor/committer community.

Successful open source communities appeal to all types of participants. In fact, there is often a progression of users to adopters to committers that helps create more core contributors working on the open source project. Therefore, it is important to consider what best practices and strategies are needed to encourage different types of participants.

Best Practices

The starting point for any successful open source project is a concept that creates something useful and has high-quality code. However, great code is not always sufficient to create an environment of collaboration and contribution. The following are five practices companies need to consider following when creating a multi-vendor open source community. These practices are based on the experience of the author observing the dynamics of open source communities and implementing many of these practices in his work at the Eclipse Foundation.

1. Engage a wider community by using a permissive or weak copyleft license. The choice of an open source license can limit participation. In particular, licensing a project under the GNU General Public License (GPL; <http://gnu.org/licenses/gpl.html>) can limit the number of organizations willing to participate as adopters. For example, a company that wants to use the technology in a commercial licensed product will be excluded due to the terms of the GPL. Using a permissive license, like the Apache Software License (ASL; <http://apache.org/licenses/LICENSE-2.0.html>), or a weak copyleft license (<http://wikipedia.org/wiki/Copyleft>), like the Eclipse Public License (EPL; http://wiki.eclipse.org/Eclipse_Public_License), makes it easier for a wider range of companies to engage as adopters.

2. Earn the trust of the community by not requiring copyright assignment. Mature open source communities will have contribution agreements that stipulate the terms of a contribution. The basic term is under which license the contribution is being provided, but some contribution agreements will request the contributor to assign the the copyright ownership to a vendor. For single-vendor-dominated communities, copyright assignment was required to allow the receiving vendor the ability to create

Best Practices in Multi-Vendor Communities

Ian Skerrett

revenue streams by implementing a dual license for the project code. MySQL (<http://mysql.com>) is the most common example of this strategy. Unfortunately, this approach creates a revenue stream that is unique to one company. In turn, this inequality creates a barrier to involvement by other companies.

Successful multi-vendor open source communities do not require copyright assignments to a single for-profit company. This is because organizations and individuals want to participate as equals in a community. If one entity is aggregating a special right, in this case copyright to the code, it creates a two-tiered system. Copyright to contributions should be retained by the originating contributor or a license should be granted to a not-for-profit foundation. For example, copyright of contributions to the Linux kernel or projects at the Eclipse Foundation remain with the contributor.

3. Be truly open: develop in the open. In an open source community, the development teams need to truly work in the open. They need to be using public issue trackers, public code repositories, and public build systems, and they must not be developing behind a firewall. Multi-vendor communities inherently require distributed development, but committing code to a public code repository once a month is not sufficient.

The vendor's development team also needs to make sure updated project plans are published and technical discussions occur in a public forum, not a hallway in the vendor's office. The development team needs to start believing that they are part of a larger community, not a traditional vendor-oriented development team. True open development allows potential participants to observe and learn how the community operates before beginning to participate. It also allows the existing participants to understand the direction of the project.

4. Have a clear policy on trademarks. The organization that controls the trademark for the project name can ultimately decide how the name is used. For instance, it controls who can use the trademark in a commercial product name, a company name, a service, or even a conference name. In the case of a fork of the project, the organization that controls the trademark controls where the project name can be used after it has been forked.

Successful open source communities define trademark guidelines that describe how the trademarks can and cannot be used. These trademark guidelines should allow all organizations the same rights and privileges. The actual trademark may be controlled by a not-for-profit foundation or a vendor may agree to equal access for all participants.

5. Implement a vendor-neutral governance structure. Successful communities have well-defined rules for making decisions. The rules determine, for example how decisions are made on admitting new committers, who decides the project roadmap and project release schedules, how technical architecture decisions are made, etc. These rules also describe the process to change the rules, strategy, and purpose of the community. Over time, all communities need to adapt, so it is important to define how things can be changed.

A successful open-development community makes these rules visible in the form of a governance document. This allows everyone to know what to expect. A truly vendor-neutral governance structure ensures that no single organization is provided extra decision-making influence within the community.

Implementation: The Role of Foundations

These best practices can be implemented in a variety of ways and to different degrees. Many

Best Practices in Multi-Vendor Communities

Ian Skerrett

successful open source communities are set up and managed by for-profit companies, such as Red Hat's Fedora and JBoss, VMWare's Spring-Source, and Google's Android. However, open source foundations have demonstrated a scalable model for creating multi-vendor communities. Organizations such as the Apache Software Foundation, Eclipse Foundation, Linux Foundation and many others have implemented many of these best practices for their communities. Starting a new open source project under the auspice of one of these foundations allows for a project to start in less time and at a lower cost than without the help of the foundation, plus it also offers the benefit of leveraging an existing community.

Conclusion

Multi-vendor open source communities are the future of corporate open source. This model best allows companies to leverage communities by collaborating on development and thereby lowering their development costs and gaining access to a wider addressable market. To create a successful multi-vendor community, there needs to be a level playing field for all participants. No one company should be in a position to veto decisions, hold an institutional advantage on any potential profit, or control the intellectual property. Development teams also need to work in an open community, not behind a firewall. Vendors that participate in successful open source projects win by giving up control and following best practices to engage with their communities.

Ian Skerrett is the Director of Marketing at the Eclipse Foundation, a not-for-profit corporation supporting the Eclipse open source community and commercial ecosystem. He is responsible for implementing programs that raise awareness of the Eclipse open source project and grow the overall Eclipse community. Ian has been working in the software industry for over 20 years. He has held a variety of product management and product marketing positions with Cognos, Object Technology International, IBM, Entrust, and Klocwork. He graduated from Carleton University with a Bachelor of Computer Science and has an MBA from McGill.

How Firms Relate to Open Source Communities

Michael Ayukawa, Mohammed Al-Sanabani,
and Adefemi Debo-Omidokun

"The key for us is to know what's going on in different communities in order to assimilate that information and use it in our products. We follow a massive number of communities and what they develop, but only have the resources to build competences to use them in a handful."

Interviewee from SOT Finnish Software Engineering Ltd.
Quoted in Dahlander & Magnusson (2008)

This article explores the relationship between firms and open source communities. Open source communities create, adopt, adapt, or disseminate innovation in a manner very different from a proprietary approach. To put this in context, we first define what is meant by open source community and then examine the roles members may play in these communities. Next, we illustrate that a firm can participate in an open source development community in different ways, depending on its level of sponsorship of that community. We assert that the degree of influence desired by the firm should connect to its business strategy and the firm needs to determine how its participation and support can be used to enhance its competitive position and provide new value to its customers. We next explore three main strategies to leverage and engage communities. We also examine how community interactions are affected by the degree of openness when engaging the community and how this relates to the firm's ability to protect the competitive advantage of its proprietary assets. This discussion will help firms with strategic planning when considering how to tap into this source of technical innovation that lies outside their boundaries.

Introduction

The success of free/libre open source software (F/LOSS) has focused attention on open source communities. These communities create, adopt, adapt, and disseminate innovation in a manner very different from the traditional commercial approach. Before analyzing the role and impact of open source communities on creativity, it is important to first understand what is an open source community. Many refer to them as innovation communities, knowledge-production com-

munities, online communities, technical communities, among other names. But such a list does not provide much insight into what differentiates an open source community from other communities. A definition is required so that we may identify open source communities.

In this article, we use the following definition: an open source community is an interacting, self-governing group involved in creating innovation with members contributing towards a shared goal of developing free/libre innovation. This

How Firms Relate to Open Source Communities

Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun

does not mean that active members of an open source community cannot be commercial firms or employees of those firms, but the scope of control, the innovation drive, and the resulting innovation are mainly governed by the rules set by open source community itself.

A firm (a corporation or business that generates revenue from the sale of software products or services) can have three basic relationships with a software development team (Figure 1). In the classic proprietary model, the software development team are all employees of the firm. In the two F/LOSS development models, the development team can be thought of partially or wholly residing outside the firm. The notion extends beyond the affiliation of a given developer to the firm (i.e., an employee-employer relationship) and includes the degree of influence on the decision-making processes in the project. This includes such factors as features, release schedules, and the direction of the project.

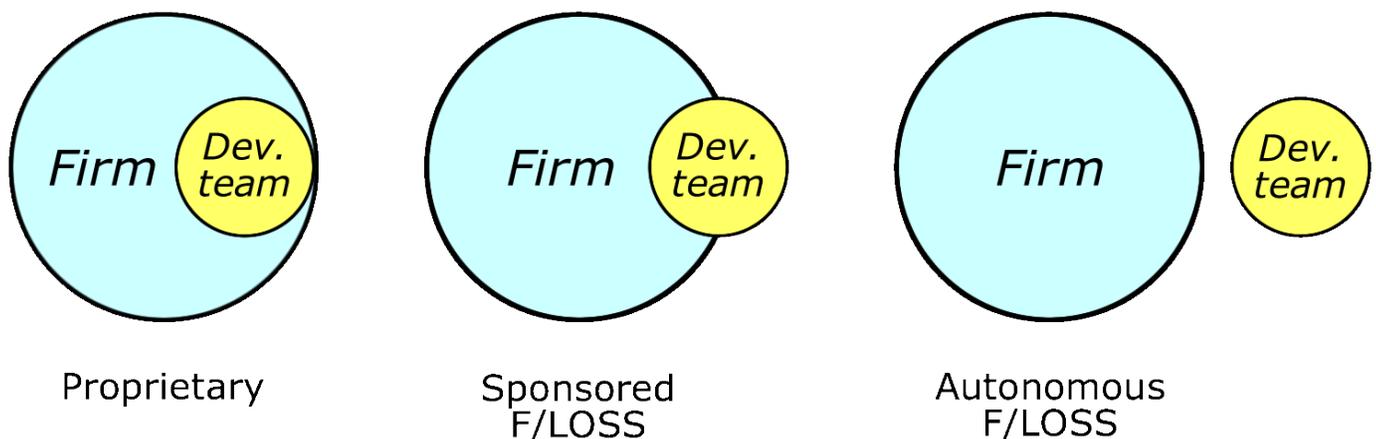
Open source communities are an important source of technical innovation that has helped many firms develop successful market and profit strategies. This further increased the interest in

studying the creation of these communities, and subsequently how firms can strategically leverage this source of innovation. As such, it became important to understand how a firm can relate to and interact with open source communities, to both enhance the firm's opportunities and bring value back to that community. The best place to start is in understanding the fabric of these communities.

Roles in Open Source Communities

Members in an open source community seek different benefits from their participation, but have a common interest in creating the open source solution. The community is made of members that can be either firms or individuals. These members constitute the governing body, sponsors, suppliers, complementors, developers, testers, single and enterprise end users, advocates, promoters, and legal experts. The motivation for joining and contributing to the community can vary. For a firm, it can be to extend their market reach, to provide a non-proprietary solution to their customers, to contribute to the public good, or to learn from and use the resources in the community. By participating, it increases the

Figure 1: Relationship Between Firm and Development Team Under Different Software Development Models



How Firms Relate to Open Source Communities

Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun

firm's opportunity to sell complementary assets and services to users and other members of the open source product.

The momentum of open source is perhaps a component of a more general trend towards openness in innovation. This shift to collaboration and partnering with communities is evident in the many dynamic roles firms have taken in the open-source environment. These roles are mainly:

- supplier partnering and co-development
- customer-based co-creation and crowd-sourcing initiatives
- co-opetition (competitive cooperation) with rivals
- open-source community extensions to internal development resources
- development of consortia
- open-outsourcing design and engineering to instantiated or existing open source communities for initial or further development.

In return, open source communities play important roles within this business environment:

- contributing to innovation and the creative process
- advocacy and marketing of the product of open source design
- complementing competitive strategy and creating alternative business models
- providing feedback, testing, and bug reporting

Because these roles played outside of the firm, their management requires a very different approach from the traditional "command and control" relationships in a business organization. It

is more of a collaborative and open approach to making decisions, an approach that is a result of the historical roots of the open source movement and its emphasis on the community principles of inclusion and meritocracy.

Strategy and Communities

F/LOSS and its communities are tools that provides a new degree of freedom for developing a firms' business strategy. Participation in a F/LOSS development community is an important strategic decision. A firm must first determine how it can leverage the community to enhance its competitive advantage and provide new value to its customers. Table 1 provides some examples of why participation in F/LOSS development may provide new opportunities or cause undesired side effects for a firm.

As an example, a firm may find itself in a weak position competing against a more popular, but proprietary product. By releasing their code base and creating a F/LOSS community, they disrupt the competitive environment. New players are brought into the game, attracted by the lower costs to access this market. Every new entrant adds momentum to the open source alternative and creates new value for end users through an expanding diversity of options and overall lowered cost structure. An implication for the firm is that it will find itself now competing against new entrants in this market, each of which is leveraging the same open source solution and community. The firm must therefore think carefully about what other value they can bring to their customer base and how they will compete and profit in this new environment.

How Open is Open Enough?

There are degrees of openness. A firm must consider how open is open enough with respect to both engaging the community and protecting the competitive advantage that proprietary assets can bring to the firm. This is not an all-or-nothing situation; many firms take a balanced or

How Firms Relate to Open Source Communities

Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun

Table 1: Reasons for a Firm to Engage in F/LOSS Development*

Reason	Possible Strategic Implications
Promoting open development communities that create new business opportunities through adoption by customers and their suppliers	Need to define the firm's role in fulfilling customer needs in a heterogeneous environment
Promoting company branding with an increased mind share with developers in the open source community	Need to connect with the complementary products that support the firm's brand
Sustaining a product line in a small or declining markets and profit from sales of complementary assets.	Need to manage the loss of control in servicing this market.
Promoting an open standard and benefit from the network effect of expanded market share	Need to fit the open standard with product and business goals
Lowering development costs by building from open source components with more effort spent on the differentiating items for which customers are willing to pay	Need to manage the contributions back to community (required or desired) and how this may limit the ability to directly profit from code
Accelerating time to market by reusing open source components with proven functionality making it easier and faster to demonstrate capability	Need to pay attention to licenses and the possible need to rewrite code used in prototyping

*Derived from Capek (2008; <http://tinyurl.com/29qh674>) and Weiss (2010; <http://tinyurl.com/275qtr7>)

hybrid position. A firm can open commoditized layers that no longer bring a competitive advantage and continue to control others that continue to differentiate their product. Table 2 summarizes the options available to firms with respect to the openness of product development.

In determining the openness of its strategy, a firm might want to answer the following questions:

- How does pursuing such a strategy enable value to a broader base of users (e.g., through reliability, quality, cost, variety, and the availability of complementary assets)?
- Will the firm be able to attract a critical mass of developers to serve these users?

- How will sharing increase the overall returns to suppliers of complementary assets (positive feedback due to network effects) and thus justify their participation in the community?
- How will the approach create lasting barriers to imitation?
- Is the approach likely to create a competitive advantage because it add to customer value?

Leveraging F/LOSS Communities

Once the decision has been made to leverage F/LOSS, it is important to consider how to engage and leverage these development communities. Software firms make use of open-source communities that are associated

How Firms Relate to Open Source Communities

Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun

Table 2: Product Development Openness Options*

Product Development	Characteristics
Proprietary	<ul style="list-style-type: none">• pioneering strategy• establishes barriers to imitation• appropriation of returns to platform owner• winner takes all
Open standards	<ul style="list-style-type: none">• favoured when a firm has low market share or limited market power• changes the balance of power based on adoption of open standards and shifting competitive advantages to other layers• more complicated to manage; may cause loss of revenues and lock-in• competition is based on marketing, customer service, product design, and operational efficiency• offensive strategy is to speed the adoption of a standard and other positive network effects
Open source	<ul style="list-style-type: none">• compete based on implementations, rather than erecting barriers to imitation• disclosing the technology prevents appropriation of the returns by any single firm• hybrid is a balanced approach that opens access to commodity layers while controlling or complementing layers to retain the opportunity for differentiation• increases interoperability• the partly open variant should have restrictions but still provide value to customers• limits the ability of competitors to use it directly

*Derived from West (2003; <http://opensource.mit.edu/papers/rp-west.pdf>)

with their use, and benefit from the creative ideas of individuals outside the company. But the inflow of such ideas does not happen spontaneously. The community can help firms increase the resources they can draw upon in the innovation process, and the firms must identify where the knowledge resides and how it can be captured and subsequently used. The table below shows three strategies of engaging communities. They are accessing, aligning, and assimilation:

Dahlander and Magnusson (2008; <http://tinyurl.com/3yvk853>) have identified three strategies

that firms can employ when engaging open source communities: accessing, aligning, and assimilating.

The accessing strategy extends the resource base by creating new communities that attract outsiders to firm's area. Firms in these communities identify and expand niches by developing unique offerings, attracted by the opportunities that mass customization brings. To sustain these communities, it is important to establish critical mass and firms must invest in the community. These communities themselves are a good marketing channel and can enhance a firm's brand.

How Firms Relate to Open Source Communities

Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun

Alternatively, a firm can invest in one or more existing communities without the need to build a new one from scratch. This avoids the cost and risks related to community building but does have a drawback of fewer opportunities to directly influence or control the community.

The second strategy is to align the firm's strategy with that of the community. Alignment is not without its challenges since members are often driven by different motives. One particularly important element to clarify is ownership of the source code, which impacts the basis for collaboration between the firm and the community. Clarity in licensing is needed for mutual trust and to avoid conflicts. Another alignment tactic is to influence direction of the development by providing incentives and creating stimulating challenges. This can be done through competitions and awards for developing specific features.

The third and final strategy is to assimilate the work developed in open source communities. This does require resources to evaluate and select source code from communities. The parameters for selection must include how the license may determine the future degree and scope of control over the entire code base. The firm also needs to decide what tasks are best done internally and how to best leverage the community resources. In a reverse flow, non-strategic source code can be delivered to the community, building legitimacy. Of course, a careful decision must be made as to what remains proprietary and outside the scope of community development, given that competitors will have equal access to the code and cannot be excluded from the distribution.

Firms can use these strategies to effectively scan the environment, evaluate the developments outside the core areas of activities, and rapidly integrate the external knowledge and its components to its products and services. Use of these strategies is fundamental to developing and sustaining a competitive advantage if a

firm shifts from internal development and manufacturing to assembling knowledge and components available through open source communities. When firms rely heavily on communities, the potential for firms' specific knowledge to provide competitive advantage will be reduced. Using communities is a way for a firm to increase the total amount of resources it can draw upon in the innovation processes, but at the same time there is a counter-acting need to appropriate the potential value of an innovation by limiting other firms from accessing to the same resources and information. The distributed nature of open source innovation puts different demands on firms aiming to use the knowledge residing in these communities for their business purposes and calls for new means to coordinate and control the development and use of knowledge over time.

How to Build an Open Source Community

It is easy to start a F/LOSS project but it is difficult to succeed with it. There are many more failures than successes in open source projects, with figures from SourceForge suggesting that 80% of all hosted projects account for a very low proportion (0.5%) of total downloads (<http://tinyurl.com/25yf3cy>). This should not be surprising; many things driven by human social dynamics follow such power-law relationships. Therefore, there is a real need to establish a critical flux of support and activity in the community. In simple terms, this means recognizing that there is competition for resources and the project initiator must pay attention to both attracting and motivating developers for their project.

Joining or forming a F/LOSS community is not without its costs, either to individuals or firms. But in an open source community there is no singular source of binding financial compensation for the membership. Therefore, there is a need to contend with the lack of traditional command-and-control methods and to deal with the generally higher turnover, a usual feature of

How Firms Relate to Open Source Communities

Michael Ayukawa, Mohammed Al-Sanabani, and Adefemi Debo-Omidokun

F/LOSS communities. Understanding the joining process will help manage the barriers to joining that new community members face. Typically joiners must show technical expertise to make a contribution, which follows the principle of meritocracy. The joining script is behavioural and is based on the type and intensity of the activity. Often starting with lurking (quietly viewing) or reporting a bug, users will generally have to find a place to make a contribution themselves. Other project aspects can be reviewed to reduce the barriers to joining, which includes modularization, forum management, documentation, design framework for adding features independent of the kernel, and choice of programming languages. There is also a need to establish trust within the community, which might include commercial firms. This includes making a clear licensing agreement and appropriating and sharing of value among members of the community.

Conclusion

F/LOSS and its communities help firms tap into a source of technical innovation outside the boundaries of the firm. Going down this path will often require revisiting and tuning the firm's product and business strategies so it can profit from the new value that comes from the contributions of the whole community. In this article, we have shown how the creation and strategic cooperation with these communities can both enhance the firms opportunities and bring value back to that community.

Michael Ayukawa is a Master's student in the Technology Innovation Management program at Carleton University and plays an active in several emerging business ecosystem projects, including co-founding Cornerportal Inc., a company that will help bring economic opportunity to more individuals in more communities worldwide.

Mohammed Al-Sanabani is a Professional Engineer and a Master's student in the Technology Innovation Management program at Carleton University. His interests include charitable education foundations. His initiatives include establishing an education foundation for graduate students in Yemen: <http://education4yemen.org>

Adefemi 'Debo-Omidokun is a graduate of Electrical/Electronics Engineering, a telecommunications professional, and a Master's student in the Technology Innovation Management program at Carleton University. He is a motivational speaker, founder and the President of the "Hero Mentors" young heroes development and international mentoring initiative, a non-governmental organization with the vision of raising heroes.

Recommended Reading

- *The Art of Community: Building the New Age of Participation*, by Jono Bacon
<http://www.artofcommunityonline.org/>
- "The Metropolis Model: A New Logic for Development of Crowdsourced Systems," by Rick Kazman and Hong-Mei Chen
<http://portal.acm.org/citation.cfm?id=1538808>

Private-Collective Innovation: Let There Be Knowledge

Ali Kousari and Chris Henselmans

"Man was born to be rich, or grow rich by use of his faculties, by the union of thought with nature. Property is an intellectual production. The game requires coolness, right reasoning, promptness, and patience in the players."

Ralph Waldo Emerson

Many innovators (companies or individuals) opt for a private innovation model. This model uses resources to create a product whose intellectual property (IP) is protected by the firm. At the opposite end of the scale is the collective innovation model, in which innovators collaborate and expend resources to produce a public good. Many free/libre open source software (F/LOSS) projects rely on collective innovation. Some innovators are now combining the two models into a private-collective innovation model, in which an innovator may choose to collaborate with other innovators and spend private resources while still keeping some IP private. For example, a company may release its product's source code to the public in the hope of attracting a community of contributing developers. Such a company commits its own resources to a project, but may still hold on to the intellectual property.

The success of private-collective innovation is dependent on many factors including: project interest and value, company reputation, and project status. There are benefits and risks to private-collective innovation which must be carefully weighed before making a decision to employ this model. Private-collective innovation involves the sharing of knowledge and, in some cases, the sharing of IP that may or may not be patented.

Introduction

During private innovation, an innovator such as a company or individual, commits private development resources and keeps all artifacts private regarding the development of a good. In contrast, collective innovation involves collaboration and resource sharing between several innovators to develop a public good. Private-collective innovation is a blend of the two models; an innovator collaborates and expend private re-

sources to create a public good. However, an innovator may choose not to release all IP to the public. There are both advantages and disadvantages to the private-collective innovation model and these will be discussed further in this article.

As an alternative to private software development, a company may choose to make a project public and F/LOSS. IBM did this when it created an Eclipse consortium, which later became the

Private-Collective Innovation

Ali Kousari and Chris Henselmans

Eclipse Foundation (<http://www.eclipse.org/org/foundation/>). Instead of releasing all source code, a company could elect to only make portions of a project open source and keep the remaining software private. This is usually done to maintain control over a project or to protect IP. The knowledge dissemination in private-collective innovation includes IP and know-how in process, architecture, and software. The sharing of knowledge during collaboration can lead to higher-quality products, decreased time to market, and maximized revenues.

In today's globalized economy, competition is harsh and the average life expectancy of a company is short. In order to survive, participants have to come up with new processes and innovative ideas to create appealing products and deliver them in a timely manner. Of course, private-collective innovation does not just happen on its own. To attract individual contributors or contributing companies, a project must be deemed of value and worthy of investing time. It takes effort by the originating company (the innovator) to convince others that its intention is for all to benefit from a project. However, those who benefit are not necessarily contributors because the project is public and therefore available to anyone. Beneficiaries may be competitors, therefore the benefits and risks to private-collective innovation must be carefully weighed before making a decision to do so.

Benefits and Costs of Private-Collective Innovation

The private-collective innovation model has two components: private and collective (public). The private component is supported by private investment, which is usually protected through patents, copyrights, and trade secrets. The collective component relates to the provision of public goods that are defined by non-excludability and non-rivalry. Non-excludability means

that any person or organization that uses the public good freely cannot withhold it from others (von Hippel & von Krogh, 2003; <http://tinyurl.com/2wjeww4>). Innovations arising from the public model are supplied to the public at no charge. By combining features of public and private models, a new innovation model is created in which participants invest in knowledge and disseminate it to the public free of charge. An example is F/LOSS that is disseminated to the public by companies that invested in creating it. This type of knowledge sharing has become very common in the software world and it is notable that all the participants benefit from it.

As long as the cost of making knowledge public is less than the benefits, it is appropriate to disseminate knowledge (von Hippel and von Krogh, 2003). Normally firms protect their knowledge by creating processes and firewalls that are costly. However, even though the risk of this knowledge being exposed is reduced, it is still present. By disseminating knowledge, the cost of knowledge management is reduced significantly (Stuermer et al., 2009; <http://tinyurl.com/32v99uj>). Moreover, knowledge sharing also has benefits for the company releasing it, including the enhancement of its reputation and positioning it as a source of expertise in the domain or industry. These effects promote trust among customers or end users and provide a competitive edge against potential competitors (Stuermer et al., 2009). Another argument for sharing knowledge is cost savings. The cost of innovation is reduced since other organizations and individuals share labour costs and contribute previously developed components.

A well-researched example of private-collective innovation is the development of the Nokia Internet Tablet (http://wikipedia.org/wiki/Nokia_Internet_Tablet). After studying this project, Stuermer and colleagues (2009) identified seven benefits of private-collective innovation:

Private-Collective Innovation

Ali Kousari and Chris Henselmans

1. Enhanced company reputation: Nokia's reputation was enhanced among F/LOSS developers by its willingness and expenditure in making the Internet Tablet software F/LOSS.

2. No source protection costs: releasing software as F/LOSS eliminates the costs involved in trying to protect the source. And as mentioned earlier, source protection might be in vain since for some products source code leaks out to the public eventually.

3. Innovation without high R&D costs: F/LOSS may enable easier, faster, and lower-cost product development.

4. Leader advantage: being first to come out with a new device can be an advantage if public involvement goes viral. The product technology can become the standard.

5. Opportunity to learn from others and by doing: collaboration in software development provides many opportunities for learning. For example, developers can learn from experts in other domains. Also, junior developers have an excellent opportunity to learn from more experienced developers.

6. Reduced development cost: greater productivity can be achieved despite reductions in paid labour. Also, code written for one F/LOSS project often can be integrated into other F/LOSS projects, which further reduces development costs.

7. Improved quality and maintenance: with greater numbers of individuals involved with F/LOSS code and a greater number of individuals contributing to testing efforts, software quality is likely to be better than an in-house alternative. Software defects are identified and fixed sooner in F/LOSS.

Although the research was specific to the Internet Tablet project, many of these research findings have also been corroborated by authors studying other private-collective innovation projects.

Stuermer and colleagues (2009) identified five hidden costs of private-collective innovation projects. These costs are:

1. Enabling others to contribute: an innovator needs to make the tools, training, and infrastructure available to allow easy entry for new contributors.

2. Releasing control: F/LOSS projects are controlled by a community. An innovator's control is limited after releasing software as F/LOSS.

3. Lack of product differentiation: competitors can use the publicly available software to create products similar to the innovator's product. This reduces the ability of an innovator to create a unique product.

4. Protecting business secrets: since a development community has to have some idea of a product's direction to enable them to contribute to it, a business may have to carefully divulge sufficient information to empower the community, without tipping off a competitor.

5. Organizational inertia: an innovator has to check any third-party software for hidden IP. This takes time and resources and has an effect on F/LOSS project progress.

How Firms Manage Private-Collective Innovation

Most of the research into open source innovation has focused on the participation and contributions of individuals in F/LOSS. Now, firms

Private-Collective Innovation

Ali Kousari and Chris Henselmans

choose to disseminate knowledge by releasing open source products and then find alternative ways to increase their revenues, either through hardware or complementary software. It is useful to explore the characteristics of those firms that give out knowledge free of charge and examine how they can be profitable at the same time.

According to a study conducted by Fosfuri and colleagues (2008; <http://tinyurl.com/29wzhyo>), the way companies disseminate knowledge is not by charity. They have full control over the process to avoid product deviation. Interestingly, those who have numerous software patents or copyrights are more likely to disseminate knowledge and F/LOSS products. A key reason is that because they have complementary products, they still have other sources of revenue. Good examples are services or products sold on top of F/LOSS products. Another reason is that companies that disseminate knowledge manage to conduct the evolution of products either by having full development and feature control or by enforcing rights and patents to prevent contributors from complementing existing knowledge. For instance, IBM has shown strong support for F/LOSS by granting licenses for more than five hundred of its patents to any open source initiative in the hope that other patent holders join the effort to create a "patent commons."

Firms that have an array of patents have stronger bargaining power and are in a better position to deal with infringed patents held by other entities. Conversely, firms that have numerous trademarks have less incentive to release knowledge and innovation (Fosfuri et al., 2008). The reason is that considerable investment has been made toward branding and image creation. It is part of the intangible assets of a firm. Therefore switching business models can be costly and can confuse existing customers who have adhered to a certain brand (Fosfuri et al., 2008). Lastly, those who have trademarks on

hardware are more likely to disseminate knowledge since they have the incentives to do so. For example, firms that have built a reputation in hardware are more likely to disseminate knowledge on their software product since it complements their hardware; it is cheaper to assemble and bundle existing F/LOSS products. It also reduces the bargaining power of specialized suppliers of software by providing a more customizable alternative (Fosfuri et al., 2008).

How Knowledge is Created and Disseminated Among Teams

It is instructive to draw a parallel between private-collective innovation and collective knowledge sharing. Collective knowledge sharing is the process of transferring innovation from one innovator to another, and it is a common attribute of open innovation teams. Inter-organization collaboration can bring considerable value to products and can decrease the time needed to bring innovations to market. A good example of this type of collaboration is a strategic alliance, such as the SCOPE alliance (<http://scope-alliance.org>) which regroups major software and hardware vendors and service providers. By elaborating upon open specifications, all products offered by the alliance provide added value to customers by ensuring stability, interoperability, and flexibility. When people from different companies have to work together and share knowledge, the process becomes complicated. The diversity of the teams can be a source of creativity but can also lead to social and communicative dilemmas (Chatenier et al., 2009; <http://tinyurl.com/28ewfc>). Therefore, it is important to know how this collective knowledge is created and disseminated.

Chatenier and colleagues (2009) reviewed the literature on the process of collective knowledge creation. They split the affecting factors into three main categories: team emergent states, team composition inputs, and team-level inputs.

Private-Collective Innovation

Ali Kousari and Chris Henselmans

Team emergent states refer to cognitive, motivational, and affective states that occur when team members work together. For instance, group efficiency is a factor that measures the team capability and reciprocal commitment. In essence, those firms that participate in open innovation must find a way to be good partners and openly share information and prevent strategic spillover of knowledge to competitors.

The learning climate is another important factor, which includes team culture, atmosphere, and the qualities of environment that facilitate learning and collaborations. There must be a balance of trust between different teams. Too much trust brings high levels of information sharing but can also diminish innovation if it causes teams to neglect checking each other's work. Insufficient trust can bring other forces such as the loss of knowledge transfer, suspicion, and skill depreciation. Similarly, power distribution represents another dilemma and teams must find a good balance of power; its absence could result in loss of ownership and impact the knowledge-sharing process.

The above factors can affect the outcome of an innovation and they are directly related to knowledge sharing. It is important to consider both the practical (strategic) and the cognitive factors that lead to knowledge dissemination. When firms decide to disseminate information and

knowledge in a private-collective model, they must ensure that their strategies are clear on what they intend to perform. They must also consider the human factors at an early stage.

Conclusion

Private-collective innovation has been a success for some companies. Companies like Nokia and IBM put in a considerable effort to maintain private-collective innovation. Their involvement to commit physical and financial resources into collaboration does not go unnoticed and helps build a good company reputation. All companies that opt for private-collective innovation learn from their experiences and will be better prepared for any future collaboration.

Ali Kousari is a graduate student in the Technology Innovation Management program at Carleton University. He is currently CTO at Systema Technologies in Geneva, Switzerland. He holds a BScH in Software Engineering from Carleton University.

Chris Henselmans is a graduate student in the Technology Innovation Management program at Carleton University. He has over 25 years experience in embedded software development. He holds a BScH from the University of Waterloo and a BGS from Athabasca University.

Control in Open Source Software Development

Robert Poole

"I've always wanted to own and control the primary technology in everything we do."

Steve Jobs

In this article, we examine typical fears associated with a perceived loss of control in an open source software development project. We describe various development models, including hybrid models that provide companies with control over key aspects of product development. Finally, a description of control within open source projects illustrates that self-regulating control mechanisms that exist in this model. A better understanding of control as a factor will help companies achieve their for-profit objectives using open source software.

Introduction

Open source software has become a mainstream tool that all companies consider as part of their product development strategy. Open source provides entrepreneurs with a way to increase their chances of earning revenue quickly, often with little or no start-up costs. Senior managers of existing businesses use open source to innovate faster, compete more effectively, and grow revenue.

Fears of Losing Control

Companies ignore the benefits of open source at their peril. The legitimacy of open source as a credible business strategy is illustrated by a 2009 survey of 54 private investment firms by the 451 Group (<http://tinyurl.com/yfhw65c>). The consulting firm conducted the study to gauge the commercial adoption of open source. They found that almost three times as many investors indicated that they would invest in an imaginary start-up that used a mix of open and proprietary licensed software over the same business that used only a proprietary licensing model. It is clear that companies should not assume a proprietary model by default.

When considering relying on open source to generate revenue, companies may fear losing control over the execution of their product development strategy, including:

- development direction, priorities, progress, and product quality
- ability to compete effectively
- protecting intellectual property

Models of Software Development

Watson and colleagues (2008; <http://tinyurl.com/3yhzqu8>) describe five models of software production or distribution. The first is the closed, proprietary model, and it has dominated the software industry for most of the past 40 years. The second is the open communities model, which has been around for the same length of time, but it is only in the past dozen or so years that various forms of development based on open communities have grown in popularity and support. The remaining three are open source models: i) corporate distribution of open source software; ii) sponsored open source, and iii) second-generation open source. Each of

Control in Open Source Software Development

Robert Poole

these three open source models are used by companies to generate revenue from open source software in various ways and each of them provides at least a partial solution to the four fears described above.

The corporate distribution model provides a customer with a packaged installation of the open source software combined with other complementary services, which are typically installation, training, support, and custom development. The customer receives the software in the same way that they would receive proprietary software, thus maintaining a level playing field with proprietary software vendors and not diminishing their ability to compete.

The sponsored open source model involves the application of corporate resources to provide paid developers to work on an open source software project. This approach, which can be used in combination with the corporate distribution model, provides the vendor with influence or even control over some elements of the product development, including development direction, priorities, and progress. Ideally, the paid development work undertaken would be of sufficient interest to the volunteer community so that it can be leveraged and supported, thus realizing some of the significant advantages of the open source approach, such as speed of development. This model also reduces the risk of poor product quality since testing can be one of the areas funded by the corporate sponsor.

The third approach, that of the second-generation open source (also known as OSSg2 or professional open source), represents a hybrid of the corporate distribution and the sponsored open source software models. OSSg2 companies maximize their influence over the product by funding of much of its development. Packaged software installations are available, however, unlike the corporate distribution model where packaged installations are sold, the OSSg2 mod-

el makes them available for free. The OSSg2 model involves maximizing control over the code so that the OSSg2 company can provide higher-value services based on the company's superior knowledge of the code. One of the ways that control is created is by not releasing all of the code as open source. Because these firms control parts of the code, they can use a dual-licensing strategy to sell a traditional software license in addition to the free open source license. Examples of OSSg2 companies include Trolltech (acquired by Nokia in 2008) and MySQL (acquired by SUN for \$1billion in 2008). The OSSg2 model effectively addresses all three fears described above. Not only does this model maximize control over product development and provide an effective means of competing with both proprietary and pure open source vendors, some elements of intellectual property protection are maintained and exploited.

Further Hybrid Strategies

West (2003; <http://tinyurl.com/29u8yot>) describes two hybrid strategies that combine elements of both proprietary and open source. The goal of this hybrid strategy is to maximize control over product development in ways that maximize the advantages of both strategies while minimizing their disadvantages.

The first hybrid strategy is to open only those parts of the product that do not provide a basis for competitive differentiation. The commodity layers, once opened up to external innovation, can be used to both create communities of active contributors and drive new product innovation. This strategy can also serve both to undermine the competitive strength of competing firms and to drive broader adoption of the now-open parts of the company's product. In the extreme version of this strategy, the core product is fully open and the company derives revenue through the creation and sale of proprietary extensions.

Control in Open Source Software Development

Robert Poole

The decision whether or not to choose the "open parts" strategy comes down to evaluating where the points of differentiation lie. If the value provided to the company and its customers is derived from the entire core of the product, then the company should retain proprietary control over the core. If the value lies in discrete, identifiable parts of the code, then the company should protect and control those elements and release the remainder as open source. If the value to the company lies in either proprietary extensions to the core or in the provision of complementary services, then the company should release the entire core as open source.

Even if a significant proportion or even all of the product is subject to an open source license, modularization can be used as a strategy to control or at least influence development. The idea behind modularization is to create separate development initiatives for different parts of the product to encourage contributions and ensure that development of the entire product cannot be as easily subverted or taken off-track (Baldwin & Clark, 2006; <http://tinyurl.com/2d7j3rw>). In this way, control can be retained using an open source approach. Modularization makes it easier for others to contribute, but also allows the company to focus on controlling the key aspects that are most important to their business objectives.

The second hybrid strategy described by West is to impose disclosure or licensing restrictions that prevent the code from being shared or used in undesirable ways. The biggest challenge with this approach is building a healthy, self-sustaining development community around the open source components when restrictions govern how the software can be used.

Control Within Open Source Projects

It is important to examine the level and types of control available within open source software projects. At least in the area of control over de-

velopment direction and quality, Gallivan (2001; <http://tinyurl.com/36zonl8>) identified strong explicit and implicit forms of control in open source development practices. Explicit control refers to rules and norms provided in the documentation and agreements; implicit control refers to the emphasis on individual reputation, which is an important currency in open communities, especially when non-monetary motivations are prevalent.

Similarly, Markus and colleagues (2000; <http://tinyurl.com/2uvwbef>) noted the importance of both self-control and social control in virtual organizations generally and open source development communities specifically. In this context, the desire of developers to preserve and enhance their own reputation provides self-control mechanisms; in contrast, social control mechanisms ensure that developers are monitored by their peers, who provide openly positive and negative feedback, potentially including sanctions as a further extension of explicit control.

Companies can also exert control over development by offering incentives to developers to work on particular features or tasks (Dahlander, 2008; <http://tinyurl.com/3yvk853>). These incentives may include competitions or even financial compensation. Similarly, more and more open source developers are being paid by their employers to contribute to open source projects, which also provides a direct form of control from sponsor companies over development efforts.

The process of applying open source principles to a product opens the innovation process to individuals outside of the company. This process also requires a change to the company's business model and drives the need for entrepreneurs and senior management to make decisions around who will control a product's development and how this control will be exerted, both explicitly and implicitly.

Control in Open Source Software Development

Robert Poole

Customer Control

Finally, companies should also view the control issue from the perspective of their customers. Although the company may be giving up a degree of control, one of the key benefits of open source software, as expressed by customers, is the increased control over their own business processes. While the provider company may not wish to reduce the switching costs of its customers through their support of open source solutions, this effect may be counterbalanced by an increase in customers who are attracted by the increased control offered to them.

Conclusion

Despite the success of many open source strategies, proprietary-minded companies may still fear the loss of control over product development, and the resulting impacts on progress, quality, competitive advantage, and the protection of intellectual property. Understanding the mechanisms of control inherent in open source projects and the benefits of hybrid approaches helps companies articulate these fears and make appropriate strategic decisions to match their business objectives.

Robert Poole is a Chartered Accountant with 15 years of experience building and deploying business intelligence and social analytic solutions to global enterprises. As a consultant, Robert has provided his expertise to private and public-sector clients including Federal and Regional governments. As an entrepreneur, Robert has created several technology-related companies and has appeared on CNBC's Power Lunch. Robert is also a Master's student in the Technology Innovation Management program at Carleton University.

Nokia's Hybrid Business Model for Qt

John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim

"Business opportunities are there to be exploited if organizations opt for an effective business model. ... It is important to realize that there is not just one effective business model for the Internet era."

Wendy Jansen, Wilchard Steenbakkers, and Hans Jagers

In today's challenging economy, startup companies are finding it more and more difficult to gain a foothold and traction in the market. Free/libre open source software (F/LOSS) allows a company to gain exposure to their products. However, few firms offer F/LOSS solutions alone. The vast majority combine proprietary and open source products while receiving revenues from both traditional license fees and open source offerings (Bonaccorsi and Giannangeli, 2006; <http://tinyurl.com/22vo453>). This dual practice of offering F/LOSS as well as a commercial license is a hybrid business model.

In this article, we focus on the hybrid business model for Nokia's Qt product: how it is implemented, why it was implemented, and the extent to which the model has been effective. The Qt story illustrates how F/LOSS business models were developed during a period when participants were just beginning to understand how to make money with open source.

An Introduction to Qt

Qt (<http://qt.nokia.com>) is a cross-platform application and graphical user interface (GUI; http://wikipedia.org/wiki/Graphical_user_interface) framework that enables web-enabled applications to be run on all of the major operating systems. The key benefit is that applications can be deployed across desktop, mobile, and embedded systems without requiring modifications to the source code to support each device or operating system. Qt was first developed in the mid-1990s by a company called Trolltech in Oslo, Norway. At that time, Java was in its infancy and its user interface capabilities lacked the performance offered by platform-specific toolkits such as MFC and XWindows. The Qt framework allowed an application to be developed only once and then easily recompiled for different environments. The application design remains the same regardless of whether the application runs on

Windows, Macintosh, or UNIX. Also around that time, a new operating system called Linux began attracting serious attention from desktop and server communities alike. Linux provided a compelling desktop environment called KDE in which Qt was the underlying graphics engine. Qt provided software developers with a comprehensive library of classes ranging from GUI widgets, database access, and multithreading support (among others) that could be easily ported to all the major operating systems, thereby freeing development teams from maintaining a separate source code configuration management tree for each operating system. Over the years, the Qt framework has grown from a GUI toolkit to a fully featured application framework.

Qt and Trolltech were acquired by Nokia in 2008 and recently, the Qt framework has propagated into the world of embedded applications, specifically on Nokia's Symbian as well as embed-

Nokia's Hybrid Business Model for Qt

John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim

ded Linux platforms. Before Nokia's acquisition, Qt was offered under a hybrid business model. A stipulation to this offering was that open source users could only build Qt using open source compilers such as MinGW (<http://mingw.org>) and GCC (<http://gcc.gnu.org>), whereas commercial users could build Qt against compilers offered by Windows. This stipulation was removed when Nokia acquired Trolltech, allowing open source developers to build Qt with compilers from Microsoft. This suggests that Trolltech's hybrid strategy fits with Nokia's future plans for the Qt product.

Nokia's Business Model Strategy for Qt

From the beginning, Qt was offered as a dual-licensed product. Users could access Qt and its source code free of charge and use it as they wish within the bounds of the General Public License (GPL). For users with needs beyond what is offered by the GPL, the option of purchasing a commercial license is available. This creates a competitive advantage for Qt over other products that provide only limited access via trial versions of their software. This dual licensing forms the basis of Qt's hybrid business model. Watson and colleagues (2008; <http://tinyurl.com/2u35etk>) describe this as an example of second-generation open source (OSSg2), which is also known as professional open source. They describe OSSg2 firms as a hybrid between corporate distribution and sponsored F/LOSS. They identify the attractive feature of this business model: "customers may use a product without paying a license fee; however, if they augment the original source code and do not wish to release the modifications under an OSS license, they must buy a commercial license." This allows prospective customers ample time to use and experiment with the product before being required to purchase it, thereby increasing customer satisfaction. By offering Qt in a form that is superior to the limited trial offers (either through time or functionality) of products by proprietary vendors, Nokia has created a com-

petitive advantage over its competitors and has delivered new value to its customers.

Nokia's dual-licensing strategy for Qt actually includes three distinct licensing options. The first option is commercial; the user pays a licensing fee and receives a high degree of freedom. The next two options align with the GNU Lesser General Public License version 2.1 (LGPL; <http://gnu.org/licenses/lgpl-2.1.html>) and the GNU General Public License version 3.0 (GPLv3; <http://gnu.org/licenses/gpl.html>). Table 1 summarizes the implications of these three licensing options.

Hybrid business models, such as Nokia's business model for Qt (<http://tinyurl.com/28wz533>), work on the concept of *quid pro quo*, which is Latin for "this for that." Commercial users pay for a license and, in exchange, can use the framework without the need to share their results. Users from the open source community benefit from the full functionality of the Qt framework and, in turn, they contribute back to the open source community. Nokia also benefits from this open source relationship in two ways. First, it can use the open source community to test and validate its products. Second, it can draw upon the open source community to fill its employment needs. Trolltech's CEO once remarked in 2008 that its approximately 230 employees were recruited almost exclusively from the open source community (Watson et al., 2008). One could argue that these benefits do not make up for the risks involved with making a company's source code open for competitors and possible plagiarists to see and use. However, the risks of patent infringements and copyright problems are lower for an OSS company using hybrid business model, in part because of the visibility of the code (Watson et al., 2008).

The risks of intellectual property theft are shadowed by the benefits, especially in light of recent emerging relationships from ecosystems within the open source community. Dynamic open

Nokia's Hybrid Business Model for Qt

John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim

Table 1. Qt Licensing Options

	Commercial	LGPL	GPL
License cost	License fee charged	No license fee	No license fee
Must provide source code changes to Qt	No, modifications can be closed	Yes, source code must be provided	Yes, source code must be provided
Can create proprietary applications	Yes, without disclosing source code	Yes, in accordance with the terms of the LGPL v.2.1	No, applications are subject to the GPL and source code must be made available
Updates provided	Yes, immediate notice sent to those with a valid support and update agreement	Yes, made available	Yes, made available
Support	Yes, to those with a valid support and update agreement	Not included but available separately for purchase	Not included but available separately for purchase
Charge for runtimes	Yes, for some embedded uses	No	No

source groups such as the ZEA Group (<http://zeapartners.org>) operate towards the following goal: “we are going to group together all the people who need a whole product made but can’t invest the resources to do it, and then take that whole product and make it offerable by anyone in the network. It has so many benefits on profitability” (as quoted in Feller et al., 2006; <http://tinyurl.com/34eppr5>). A for-profit-company like Nokia benefits from the synergy open source generates, specifically in projects like KDE (<http://kde.org>). KDE is a popular desktop environment for the Linux platform and it uses Qt as its underlying graphics library. By participating in successful open source projects like KDE, interest and confidence in Qt grow. This contributes to its adoption in other high-profile applications, such as Google Earth (<http://tinyurl.com/2wnplo8>) which, like KDE, uses Qt as its underlying graphics library.

A commitment to an open source community allows organizations to align their objectives with the needs of the customer. By providing a commercial licensing scheme on top of the open source solution, Nokia has provided a solution to “the strategic problem of a firm whose customer platform and product portfolio is based on proprietary software” (Bonaccorsi, Giannangeli, and Rossi, 2006). On one hand, by offering the product under a dual-licensing scheme, a company receives the benefits of open source software, including its track record of reliability and security, and the support of a distributed network of developers who contribute to it. On the other hand, licensed software has gained an expectation for a certain standard of documentation, maintenance, product updating, and bug fixing, as well legally binding requirement expectations (Bonaccorsi, Giannangeli, and Rossi, 2006). By offering the quality expected in propri-

Nokia's Hybrid Business Model for Qt

John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim

etary software in an open source product, a company not only encourages a loyal following of developers willing to help test and improve it, but it also inspires confidence from commercial customers eager to take advantage of the high quality it offers.

The Impact of Nokia's Acquisition of Trolltech

After acquiring Qt, Nokia took steps to make the product more open. Nokia now allows open source users to build Qt applications using Microsoft's build tools, which previously required a commercial license. By removing restrictions on using Qt within the open source community, Nokia has gained more support and traction with its customers and the community as a whole. Nokia further consolidated this support and traction to promote its own open source F/LOSS platform, Symbian (<http://symbian.org>).

An operating system is nothing without the key applications that compel the market to use it. In order to create these applications, developers must have the development tools necessary to design, build, and test them. Other embedded operating systems competing with Qt all offer freely available tools to the developer; in Apple's case, the development tools are freely available after the purchase of a Mac OS X computer. Prior to acquiring Trolltech, Nokia realized it needed to follow suit in order to compete effectively.

Trolltech not only provided the Qt library, but a number of tools that allow features to be integrated into integrated development environments, such as Eclipse, Visual Studio, or Trolltech's own Qt Creator application. Therefore, the purchase of Trolltech provided Nokia with the instant toolset it needed to make application development on the Symbian platform attractive to developers. What also made Trolltech attractive to Nokia was the fact that the Qt platform had undergone years of testing and refinement in the open source community on most

platforms. By acquiring Qt, Nokia made a strategic step towards fending off the threat of Apple and Google with their own user experience solutions. Further, Nokia acquired a user interface technology on par with or exceeding the technologies offered by competitors.

Another compelling reason for Nokia's purchase of Trolltech was the opportunity to undermine the position of an old competitor. For years, Nokia's primary competitor was Motorola and both strove to get the upper hand on the other in the burgeoning mobile phone market. In order to drive their newest smartphone offerings, Motorola began using Qtopia, a product from Trolltech, to implement their user interface designs. By acquiring Trolltech, Nokia made Motorola dependent on them for their toolsets. This placed a lot of pressure on Motorola, especially after the Qtopia product line was discontinued in favour of their Nokia Qt SDK in 2009. Without a solid toolset to rely on, Motorola was hard pressed to develop products that could compete in the ever more hostile smart phone market. This ultimately led to Nokia's decision to purchase Motorola in July of 2010.

The Impact of Changes to the Open Source Community

The emergence of free/libre open source operating systems into the cellphone market changed the landscape for Qt. The Symbian operating system was originally a hidden platform within Nokia and its partner companies, including Texas Instruments. However, Nokia made the decision in 2009 to make the Symbian operating system open source and controlled by a non-profit organization, the Symbian Foundation. This move was undoubtedly in response to Google's introduction of the Android open source operating system. Qt is widely considered one the best open source GUI toolkits on the market and has "cut its teeth" by serving as the backbone for KDE, the Linux-based desktop environment.

Nokia's Hybrid Business Model for Qt

John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim

As open source projects move further into the embedded arena, Qt serves as the open source solution for realizing the front-end layers for these new projects. Nokia has caught on to this trend; effort has been made to decrease the memory footprint of Qt or maintain its performance on embedded systems. As more developers begin building new applications for these open source platforms, using a quality tool such as Qt provides Nokia with brand exposure, while increasing the breadth of applications offered on the Symbian platform. Another interesting development in the open source community is the decision by Microsoft to offer a free IDE for building Windows applications. With an IDE being freely available, this opens the door for Qt to integrate with these free tools. It will be very interesting to see what new directions the Qt product will take as the result of the recent news that the Symbian Foundation will be closing (<http://tinyurl.com/28fb2ef>).

The Impact of Market Changes

Many changes have occurred in the computing market leading up to Nokia purchasing Trolltech in 2008. To begin with, by 2008 manufacturers had perfected the mass production of sub-50nm devices (<http://tinyurl.com/2bq24qx>). This resulted in much smaller, more energy-efficient devices that included large amounts of static on-board memory. As devices became smaller, more powerful, and more energy efficient, smartphones became more accessible. Also, a stable, flexible operating system with a viable suite of applications was needed to draw consumers to the smartphone market. At this time, three major competitors existed: Nokia's Symbian platform, Google's new Android operating system, and Apple's iOS for the popular iPhone platform. Google offered both the Android operating system and associated tools for free while Apple also provided iPhone development tools for free; Nokia needed to provide a comparable development environment. As a result, Symbian decided to provide its Symbian operating system as open

source along with a number of applications, including a mobile web server, a SIP application stack, and an S60 DSS browser.

The missing link in Nokia's solution however was a set of development tools for building applications. This market push for development tools provided strong motivation for Nokia to acquire an open source user interface design toolkit like Qt.

Conclusion

Nokia's actions have proven that the hybrid business model is "a very promising business model that could emerge as a dominant model for OSS development in the coming years." (Watson et al., 2008). The hybrid business model that Nokia has adopted for its Qt product has been very successful for the firm as well as the open source community as a whole. Nokia can be commended for not only keeping its hybrid model but adding features and opening it up even more to the open source community. This lends credence to the hypothesis that "hybrid business models are not a transient stage but rather a permanent feature of the new industry" (Bonaccorsi and Giannangeli, 2006).

By acquiring Qt from Trolltech, not only did a major corporation build goodwill with community members, it also placed significant pressure on its competitor Motorola, who also recognized the value of Qt and made it a cornerstone in its smartphone solution. Once Qt was in its control, Nokia allowed it to be more integrated with other proprietary applications like Visual Studio and thereby making the framework more attractive to developers. At the same time, Nokia removed support for its Qtopia product line, sending Motorola's development cycle into a tailspin. Nokia has proven that the hybrid business model and open source in general can be used as both a sword and a shield in an ever-changing marketplace.

Nokia's Hybrid Business Model for Qt

John Schreuders, Arthur Low, Kenneth Esprit, and Nerva Joachim

John Schreuders is a graduate student in the Technology Innovation Management program at Carleton University in Ottawa. Prior to his work at Carleton, John received his BEng in Computer Engineering at the Royal Military College of Canada in Kingston, Ontario. John has 15 years of experience in designing real-time software systems in many different fields, including defense, aerospace, finance and telecommunications.

Arthur Low is a graduate student in the Technology Innovation Management program at Carleton University in Ottawa. He has over 18 years of experience in Integrated Circuit design. Art is an Electrical Engineer who uses open source IC design simulators and software development tools for his cryptographic Silicon IP business, Crack Semiconductor.

Kenneth Esprit received his BSc degree from the University of Pinar del Rio, Cuba in Telecommunication and Electronics Engineering, in 2004. He is currently a graduate student in the Technology Innovation Management program at Carleton University in Ottawa. He has over the 6 years of experience in mobile communication and has used open source software as an optimization tool for radio frequency planning and BTS maintenance.

Nerva Joachim is an Electrical Engineer and has over ten years of experience in electronic control systems design. He has worked in Montreal, Toronto, and the Ottawa capital region. He is currently a graduate student in the Technology Innovation Management program at Carleton University in Ottawa, where he is involved in a collaborative project with Ottawa University, the Natural Sciences and Engineering Research Council of Canada (NSERC), the Ottawa Centre for Research and Innovation (OCRI), and Kylowave Inc., a company that is a member of the Lead to Win ecosystem.

IT Professional: Open Source Software

The November/December issue of IT Professional covers open source in law enforcement and education, open source in geographic information systems, and open source in web services. The article on free/open services was co-authored by Michael Weiss, guest editor of this issue of the OSBR and faculty member in the Technology Innovation Management program.

<http://www.computer.org/portal/web/csdl/abs/mags/it/2010/06/mit06toc.htm>

Computing Now: The State of Open Source

From the Introduction: "Open source has become a serious and substantial component of applications, embedded systems, operating systems, and common devices that consumers use daily. It has altered software development, licensing, hardware, methodologies, and a myriad of devices used by individuals, businesses, and governments, while simultaneously impacting the lives of people across the socioeconomic scale around the world. Additionally, the evolution of open source as a business model hasn't followed the path predicted by anyone "in the know" in the late '90s, or in the early 2000s, for that matter. And as open source has meandered down its evolutionary path, technologists, businesses, and governments have faced new and interesting challenges—and opportunities. This month's Computing Now theme compiles a variety of articles that show the many current faces of open source."

<http://www.computer.org/portal/web/computingnow/archive/december2010>

CENATIC: Report on the International Status of Open Source Software 2010

From the Introduction: "The report we present here analyses the International Status of Open Source Software, enabling us to put the current situation in Spain in context based on the knowledge of technology trends around the world, the promotion and use of open technologies in the Spanish Private and Public sectors, and the contribution of Spanish Communities of Developers and Universities to important initiatives on an international scale. It is, in conclusion, a thorough overview of the international context of open source software, creating a starting point for the identification of new business opportunities for Spanish companies, and new fields of study for CENATIC to continue promoting the use and development of open source software in Spain."

<http://tinyurl.com/2d6za4z>

Upcoming Events

January 25

Innovation Night

Burlington, ON

From successful entrepreneurs, to knowledgeable professors, to experienced investors, and anyone else passionate about innovation – Innovation Night provides access to the region's thought leaders in start-up strategy, providing invaluable input and support on those critical first steps in transitioning your idea into a business. Innovation Night provides an opportunity to showcase your idea and share your passion with a captive audience.

<http://www.innovationnight.ca/>

January 26

NetGain 5.0

Toronto, ON

Is social media a game changer? Social media needs to be monitored, measured and then analysed in order to be actionable business intelligence. To ensure competitive advantage, you need to stay ahead of rapidly evolving trends in research technologies, best practices and business strategies in this growing area. Net Gain 5.0 addresses this need.

<http://www.mria-arim.ca/NetGain5/NEWS/default.asp>

February 16 - 18

Privacy and Security Conference and Exposition

Victoria, BC

The Annual Privacy and Security Conference and Exposition, hosted by the Province of British Columbia, has become a leading event in North America for those working in the information privacy and security fields. Held in beautiful Victoria, British Columbia, Canada, the two-day conference draws an international audience of over 1000 delegates with an interest in cutting edge policy, programs, research and technologies aimed at the protection of privacy and security.

<http://www.rebootconference.com/privacy2011>

Technology Innovation Management (TIM)

Unique Master's program for innovative engineers
Apply at www.carleton.ca/tim



TIM is a unique Master's program for innovative engineers that focuses on creating wealth at the early stages of company or opportunity life cycles. It is offered by Carleton University's Department of Systems and Computer Engineering. The program provides benefits to aspiring entrepreneurs, engineers seeking more senior leadership roles in their companies, and engineers building credentials and expertise for their next career move.



Carleton
UNIVERSITY

The goal of the Open Source Business Resource is to provide quality and insightful content regarding the issues relevant to the development and commercialization of open source assets. We believe the best way to achieve this goal is through the contributions and feedback from experts within the business and open source communities.

OSBR readers are looking for practical ideas they can apply within their own organizations. They also appreciate a thorough exploration of the issues and emerging trends surrounding the business of open source. If you are considering contributing an article, start by asking yourself:

1. Does my research or experience provide any new insights or perspectives?
2. Do I often find myself having to explain this topic when I meet people as they are unaware of its relevance?
3. Do I believe that I could have saved myself time, money, and frustration if someone had explained to me the issues surrounding this topic?
4. Am I constantly correcting misconceptions regarding this topic?
5. Am I considered to be an expert in this field? For example, do I present my research or experience at conferences?

If your answer to any of these questions is "yes," then your topic is probably of interest to OSBR readers.

When writing your article, keep the following points in mind:

1. Thoroughly examine the topic; don't leave the reader wishing for more.
2. Know your central theme and stick to it.
3. Demonstrate your depth of understanding for the topic, and that you have considered its benefits, possible outcomes, and applicability.
4. Write in third-person formal style. Formal first-person style (we only) may also be acceptable.

These guidelines should assist in the process of translating your expertise into a focused article which adds to the knowledgeable resources available through the OSBR.

Upcoming Editorial Themes

February 2011: Recent Research

March 2011: Co-creation

Guest Eds.: Stoyan Tanev,
Univ. of Southern Denmark;
Marko Seppä, Univ. of
Jyväskylä

Formatting Guidelines:

Indicate if your submission has been previously published elsewhere.

Do not send articles shorter than 1500 words or longer than 3000 words.

Begin with a thought-provoking quotation that matches the spirit of the article. Research the source of your quotation in order to provide proper attribution.

Include a 2-3 paragraph abstract that provides the key messages you will be presenting in the article.

Any quotations or references within the article text need attribution. The URL to an online reference is preferred; where no online reference exists, include the name of the person and the full title of the article or book containing the referenced text. If the reference is from a personal communication, ensure that you have permission to use the quote and include a comment to that effect.

Provide a 2-3 paragraph conclusion that summarizes the article's main points and leaves the reader with the most important messages.

If this is your first article, include a 75-150 word biography.

If there are any additional texts that would be of interest to readers, include their full title and location URL.

Include 5 keywords for the article's metadata to assist search engines in finding your article.

Copyright:

You retain copyright to your work and grant the Talent First Network permission to publish your submission under a Creative Commons license. The Talent First Network owns the copyright to the collection of works comprising each edition of the OSBR. All content on the OSBR and Talent First Network websites is under the Creative Commons attribution (<http://creativecommons.org/licenses/by/3.0/>) license which allows for commercial and non-commercial redistribution as well as modifications of the work as long as the copyright holder is attributed.

The OSBR is searching for the right sponsors. We offer a targeted readership and hard-to-get content that is relevant to companies, open source foundations and educational institutions. You can become a gold sponsor (one year support) or a theme sponsor (one issue support). You can also place 1/4, 1/2 or full page ads.

For pricing details, contact the Editor chris.mcphee@osbr.ca.