TIM Technology Innovation Management Review

January 2012

Image licensed under CC BY by John Martinez Pavliga

Open Source Business

Welcome to the January 2012 issue of the *Technology Innovation Management Review.* The editorial theme of this issue is Open Source Business. We invite your comments on the articles in this issue as well as suggestions for future article topics and issue themes.

Carleton UNIVERSITY

www.timreview.ca

Publisher
The *Technology Innovation Management Review* is a monthly publication of the Talent First Network.

ISSN
1927-0321

Editor-in-Chief
Chris McPhee

The PDF version is created with Scribus, an open source desktop publishing program.

Overview

The *Technology Innovation Management Review* (TIM Review) provides insights about the issues and emerging trends relevant to launching and growing technology businesses. The TIM Review focuses on the theories, strategies, and tools that help early-stage technology companies succeed.

Our readers are looking for practical ideas they can apply within their own organizations. The TIM Review brings together diverse viewpoints – from academics, entrepreneurs, companies of all sizes, the public sector, the third sector, and others – to bridge the gap between theory and practice. In particular, we focus on the topics of managing innovation, technology entrepreneurship, economic development, and open source business.

Upcoming Issues

- *February:* Entrepreneurship I
  Guest Editor: Tony Bailetti
- *March:* Entrepreneurship II
  Guest Editor: Tony Bailetti
- *April:* Social Innovation
  Guest Editor: Stephen Huddart
- *May:* Global Business Creation
  Guest Editors: Marko Seppä and Stoyan Tanev

We welcome input from readers into upcoming themes. Please visit timreview.ca to suggest themes and nominate authors and guest editors.

Contribute

Contribute to the TIM Review in the following ways:

- Read and comment on past articles and blog posts.
- Review the upcoming themes and tell us what topics you would like to see covered.
- Consider writing an article for a future issue; see the author guidelines and editorial process for details.
- Recommend colleagues as authors or guest editors.
- Give feedback on the website or any other aspect of this publication.
- Sponsor or advertise in the TIM Review.
- Tell a friend or colleague about the TIM Review.

Please contact the Editor if you have any questions or comments: timreview.ca/contact

Editorial: Open Source Business

Chris McPhee, Editor-in-Chief

Leslie Hawthorn, Guest Editor

From the Editor-in-Chief

It is my pleasure to introduce Leslie Hawthorn, Community Manager for AppFog, as our guest editor for the January 2012 issue of the *Technology Innovation Management Review* (TIM Review). The editorial theme of this issue is Open Source Business and this issue features authors from Australia, Finland, Italy, and the United States.

The theme for the February and March issues will be Entrepreneurship, presented by guest editor Tony Bailetti. We encourage you to suggest themes you would like to see covered in future issues.

We hope you enjoy this issue of the TIM Review and will share your comments on articles online. Please also feel free to contact us (timreview.ca/contact) directly with feedback or article submissions.

Chris McPhee
Editor-in-Chief

From the Guest Editor

While technologists have long been used to the terms "free software" and "open source software," a testament to the success of these development methodologies lies in the emerging application of the term "open source" to a wide variety of sectors. Groups of doctors in disparate locations who are collaborating to discover cures for malaria describe their approach as "open source" (http://tinyurl.com/7d7oza9). Marcin Jakubkowski's TED Talk on "Open Source Ecology" – the publication of plans and schematics to enable anyone to construct the industrial machines required to power a "sustainable civilization with modern comforts" (http://opensourceecology.org/about.php) – has enjoyed more than 636,000 views (http://tinyurl.com/6dc4bpb). Danielle Gould eloquently argues that eliminating inefficiencies in our systems of food production and distribution "requires creating incentives to move from closed, proprietary approaches to open ones" (http://tinyurl.com/76haums). With the application of its high-level concepts to such a disparate set of challenges, it is tempting to argue that free/libre open source software (F/LOSS) methodologies have been validated as the superior choice overall, at least in the court of public opinion.

However, the ultimate testament to the power of the F/LOSS model remains its application in the business world. F/LOSS continues its surge to prominence as a means for businesses to cut costs, promote innovation, and further engage their customers. In 2011 alone, F/LOSS based businesses saw venture capital investment of more than $466M USD, a 24% increase over such investments in 2009 (http://tinyurl.com/6rqnphs). In one recent industry survey, 25% of respondents reported that 75% of all software used by their organization is F/LOSS (http://tinyurl.com/3sn647f). The F/LOSS software stack acts as the key component in the emerging, though increasingly prominent, world of cloud computing, allowing Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) vendors to provide services that increase their customers' revenue by decreasing time to deployment, hardware costs, hosting costs, etc.

With all the activity – and, in some cases, pure hype – surrounding F/LOSS and the application of its principles, it can be easy to assume that open source and

Editorial: Open Source Business

Chris McPhee and Leslie Hawthorn

open innovation models "just work" and carry on from there. As with all things, the narrative is much more nuanced. Open business models have proved incredibly effectively, but the overall landscape is much more complex than simply reducing the problem space to "open = good."

In this issue of the TIM Review, we present a variety of examinations of the open business model, from legal and governance issues to the long-term implications of the F/LOSS model for businesses, both those that are a pure F/LOSS play to those businesses who choose to use F/LOSS where they had been traditionally averse to it.

Juho Lindman, Assistant Professor in the Hanken School of Economics, and **Risto Rajala**, Director of Research at Aalto University, examine the impact that F/LOSS models have had on entrepreneurship and the software industry as a whole. Their research dives into the critical issues facing F/LOSS businesses and the corresponding benefits achieved by those players who effectively approach these challenges.

Deborah Nicholson, Community Outreach Director for the Open Innovation Network, presents the barriers to open innovation raised by software patents and their use by non-practicing entities – less charitably known as "patent trolls" – to produce profits purely through litigation. She provides an overview of the important legal cases informing the current debate and details how the Open Innovation Network's model provides a useful and feasible option for those open source projects and companies looking to license their intellectual property in a way that protects their business interests while helping technological innovation to more effectively progress. Deborah's article bridges the current issue with that of last month's issue on Intellectual Property Rights (http://timreview.ca/issue/2011/december).

Liz Laffan, Research Partner at Vision Mobile, discusses the Open Governance Index, a new framework of 13 proposed metrics to measure the openness of open source projects. Laffan's work focuses on eight open source projects in the mobile space, examining the successes and failures of each project in light of their openness in key areas, from acceptance of outside contributions to lowering the barrier to entry for community participants who wish to influence the outcome of product decisions. Of particular interest, Laffan examines the corresponding market success of each project when compared to its openness score, with some perhaps surprising results.

Ruth Suehle, writer and editor for opensource.com and former editor of Red Hat Magazine, examines Red Hat's formula for success as a pure-play open source business on track to earn $1B in annual revenue. She gives us a glimpse into the profound impact F/LOSS methodologies have on each aspect of Red Hat's offerings, from its software to its community outreach initiatives and its promulgation of "the open source way" as a means to promote more widespread innovation across all sectors.

Carlo Daffara, Head of Research at Conecta, discusses the long-term effects of F/LOSS on the software industry, arguing that the interconnectedness of software projects produced by the massive increased use of F/LOSS code propels innovation, collaboration, and profit in profound ways that we are only now beginning to fully understand.

Leslie Hawthorn
Guest Editor

About the Authors

Chris McPhee is Editor-in-Chief of the *Technology Innovation Management Review* and 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.

Leslie Hawthorn is an internationally known community manager, speaker, and author with over 10 years' experience in high-tech project management, marketing, and public relations. In January 2012, she joined AppFog as their Community Manager, where she is responsible for developer engagement. Prior to AppFog, she served as Outreach Manager at Oregon State University's Open Source Lab and as a Program Manager for Google's Open Source Team, where she managed the Google Summer of Code Program, created the contest now known as Google Code In, and launched the company's Open Source Developer Blog. She is also a member of the Advisory Board of the *Technology Innovation Management Review.*

Citation: McPhee, C. and L. Hawthorn. 2012. Editorial: Open Source Business. *Technology Innovation Management Review.* January 2012:3-4.

How Open Source Has Changed the Software Industry: Perspectives from Open Source Entrepreneurs

Juho Lindman and Risto Rajala

> *In addition to the business rationale behind our decisions, there are of course the ideological reasons... we wouldn't be doing this – we wouldn't have started our own company – if we didn't believe in F/LOSS.*
>
> CEO, Tripod (a case company in this study)

The emergence of F/LOSS (free/libre open source software) has triggered several changes in the software industry. F/LOSS has been cited as an archetypal form of open innovation; it consists of the convergence and collaboration of like-minded parties. An increasing number of software firms have taken upon this approach to link outsiders into their service development and product design. Also, software firms have been increasingly grounded their business models on user-centric and service-oriented operations.

This article describes a study that investigates these changes from the perspective of F/LOSS entrepreneurs. The findings are summarized into four issues that are critical in managing an F/LOSS business: i) dealing with organizational changes in the innovation process; ii) mastering user involvement; iii) successfully using resources; and iv) designing revenue models.

Introduction

The socio-technological changes pertaining to the free/libre open source software (F/LOSS) phenomenon have received increasing attention among information systems (IS) researchers. The focus of research has ranged from individual-level motivations and behavior to organizational adaptation and consequences. F/LOSS has been seen as an important driver of the information economy and an essential consideration for all software companies (Fitzgerald 2006; http://tinyurl.com7zwsn8k). In the research literature, the F/LOSS itself has been described through two distinct features. First, it is connected with licenses that provide existing and future users with the right to use, inspect, modify, and distribute modified and unmodified software to others (Von Krogh and Von Hippel, 2006; http://tinyurl.com/7747c6t). Second, it has created new communities of practice for collaborative innovation in numerous F/LOSS communities (Kogut and Metiu, 2001: http://tinyurl.com/6pexrwx; Dahlander and Magnusson, 2008: http://tinyurl.com/6w6k95q). Years of development in such communities has resulted in high-quality mainstream applications.

Recent research on open innovation has emphasized the importance to understand the impact of open innovation on firms' strategies (see Harison and Koski, 2010: http://tinyurl.com/7rxbd4a; Hauge et al., 2010: http://tinyurl.com/7sdhvjl). However, prior empirical studies have not paid sufficient attention to the effects of F/LOSS on the business of small and medium-sized software enterprises from the entrepreneurial viewpoint. Despite that the ideological roots of the F/LOSS phenomenon are rather well documented, the realities of the business environment deserve more attention. This study aims to fill this gap by improving the understanding of the critical issues in F/LOSS businesses and related entrepreneurial perceptions through a qualitative empirical inquiry among software entrepreneurs.

In this article, we focus on software companies that actively take part in open source software development. Bearing this in mind, we posed the following research question: What are the critical issues in managing an F/LOSS business? We addressed this question empirically through an inductive qualitative inquiry. Here, we describe narratives emerging from interviews of software entrepreneurs and senior managers in software firms. In our analysis of these narratives, we found four categories of issues that describe the challenges rooted in the realities of managing an F/LOSS business successfully. These lessons from practitioners add an important perspective to the theoretical knowledge of the changes the F/LOSS phenomenon has brought to the software industry.

Emergence of Open Innovation in Software Companies

The benefits of open innovation are widely accepted in open source software development communities (e.g., Von Hippel and Von Krogh, 2003; http://tinyurl.com/7w46h24). In its broadest sense, software innovation refers to research and development (R&D) activities that involve intellectual capital, physical products, and processes in software production (Vujovic and Ulhoi, 2008; http://tinyurl.com/83za6ut). Chesbrough (2003; http://tinyurl.com/cgu9u7w) observed that strategic innovations have typically been regarded as company's most valuable competitive assets, which also serve as barriers to entry by competitors. This kind of proprietary development and competition is characteristic of closed innovation processes, where technological progress has generally been kept secret to capture the potential for extraordinary returns (Meyer, 2003; http://tinyurl.com/czo6ob9). Conversely, in an open environment, a company's ability to remain competitive increasingly relies on utilizing accessible resources in the continuous development of new and superior products and services. In business environments characterized by growing instability, this approach enables them to remain competitive (Vujovic and Ulhoi, 2008; http://tinyurl.com/83za6ut).

The "F/LOSS phenomenon" describes a new paradigm for the management of software-intensive innovation as well as the development and delivery of software. Using this approach, firms work with external partners and users to develop their internal innovations and obtain resources related to external innovations that complement their offerings. Currently, F/LOSS has established positions in several market segments, ranging from operating systems, middleware, and end-user products, such as media players, office suites, and games (Von Krogh and Spaeth, 2007; http://tinyurl.com/c82ernz).

Most research on the domain has focused on successful F/LOSS projects (Radtke et al., 2009; http://tinyurl.com/d8mezja) and open source development approaches, rather than responding to the growing interest in F/LOSS among companies (Osterlie and Jaccheri, 2007; http://tinyurl.com/7hum94f). Prior research on making commercial use of F/LOSS has primarily focused on guidance for managers when considering whether or not to adopt F/LOSS (see Ven et al., 2008: http://tinyurl.com/6tgwkhg; Fitzgerald, 2009: http://tinyurl.com/8y3qqcs). Adoption for the wrong reasons can harm the organization; whereas not adopting F/LOSS might be a missed opportunity (Ven et al., 2008).

The emergence of F/LOSS facilitates access to resources and the use of capabilities as the source of competitive advantage among software vendors. Service-dominant logic (Vargo and Lush, 2004; http://tinyurl.com/4zt926w) describes a significant transition in business in terms of the use of resources. It considers resources in the development and delivery of offerings as operand resources (i.e., those in which an operation, or act, is performed) and operant resources (i.e., those that act on other resources). F/LOSS development depends, to a great extent, on resources that are external to a firm. The operand resources include, for example, the physical resources required to make services available to customers, while the operant resources, such as the requisite skills, knowledge, and capabilities, represent the intangible resources of the parties engaged in the collaboration. In F/LOSS businesses, resources are accessed through collaborative relationships between two or more parties, or, as suggested by Dahlander and Magnusson (2005; http://tinyurl.com/88djuec), in a company-community relationship.

There is an ongoing discussion regarding the principles that software companies should adopt when leveraging F/LOSS (Dahlander and Magnusson, 2005; http://tinyurl.com/88djuec). Ljungberg (2000; http://tinyurl.com/6ocuucm) argues that collaborative relationships include reciprocal obligations, which enact social relationships between the actors. Dahlander and Magnusson (2005) add that understanding the nature of the relationship is crucial to conducting business in a way that engages commercial and non-commercial actors.

Methodology

In this study, our goal was to elucidate the perceptions and practices of entrepreneurs that are engaged in the development of open source software. For this reason, we selected as the key informants in our study software entrepreneurs or senior managers of firms that have engaged with in F/LOSS activity and are responsible for the company's strategic decisions. Arguably, they represent the most influential individuals in their firms' business model decisions although we acknowledge that the responses reflect only the "opinions of the few." However, we want to stress that CEOs and founders have a particular point of view that reflect the managerial viewpoint, which can be considered salient regarding the changes in their business.

We selected five F/LOSS companies to determine how their managers perceived and described the ongoing changes in their environment. The respondents and their firms are anonymized in this paper. The method used for data collection included semi-structured, in-person interviews and an extensive set of secondary data on the case companies. We conducted interviews with each of the respondents from the selected firms over a five-year period (2004 to 2008). To gain a rich understanding about the organizations in their contexts, we interviewed the entire staff then employed by Tripod (3 persons), Yoga (1 person), and Tulip (5 persons). For larger the companies – OurDB and Nemesis – we limited our discussions to the CEOs and CTOs.

This study uses a narrative approach to interpret the stories of F/LOSS entrepreneurs and senior managers. Ramiller and Pentland (2009; http://tinyurl.com/8xuk6x8) stated that: "A story involves actors undertaking *actions* intended to accomplish certain *goals* by certain *means*, within specific settings, leading to particular *outcomes*." In our analysis, we adopt Ramiller and Pentland's approach and structure our findings by focusing on the actions, goals, means, and outcomes (pertaining to F/LOSS activities) emerging from the narratives of our interview subjects. There is an assumption that, by analyzing how stories are told and what their tellers say and do not say, we can discover the hidden meaning behind the world they describe (Burr, 1995; http://tinyurl.com/6pbzvdw).

Findings: Critical Issues in Managing F/LOSS Businesses

Interviews with the F/LOSS entrepreneurs provided a view of the realities of managing F/LOSS businesses. The narratives gathered from these interviews included a mixture of the entrepreneurs' personal experiences and their tried-and-true organization-level business practices. Moreover, they illustrate how the respondents view their business environment and the factors upon which they base their decision-making.

F/LOSS induces user involvement in software development

Integrating user feedback and requirements to the software was seen as one of the main opportunities and challenges with F/LOSS activity. The entrepreneur who founded Yoga underscored that *"the main idea of F/LOSS is working together to create tools everyone wants to use individually."* The interviewees emphasized that F/LOSS development was organized differently from proprietary development and thus is able to respond more promptly to customer needs.

> *"We would never have gained 5 million users to our database product without acting according to the principles of the open source software community. Since we first released our software under an open license, we have gathered feedback – development ideas, problem descriptions and solutions – and responded to all possible initiatives from the user community to develop the product with the skillful individuals using the product."* (CEO, OurDB)

F/LOSS enables and invites user participation in software development. The narratives reveal differences in the numbers of possible contributions among the different software product categories, but are harmonious in their descriptions of the clear role users play as contributors to the software project, product, and service. In addition, the narrative of Yoga's manager shows an example of a "classic open source development". He described that his own contributions to several F/LOSS projects have been guided by his own personal needs and preferences. His contributions were motivated by the reason that the available software did not meet his personal needs. Moreover, the CEO of Nemesis narrated that:

> *"Our solutions are made for the customers, not for ourselves. We want to build a working solution, but we want the customers to sit down with us, so we can do it on the users' terms. We believe that it is not enough for us to provide open source software. In our opinion, customers should also have open access to the actual work process – not only through external communication, but also in internal collaboration. We want to get the customers' messages heard."*

Our respondents assumed that customers had the necessary capability and willingness for the elicitation of their requirements and that this information should guide the development of software products. In this vein, customer expectations determined which solutions would become commercially viable. The significant role played by such expectations was taken for granted; the respondents assured us that customer expectations drive actual customer behavior and their identification is thus salient to software vendors.

Open source development relies on external resources

One of the advantages of F/LOSS is tapping into external contribution in the innovation activity. That is, F/LOSS activity emphasizes access to external capabilities rather than internal resource ownership. Harnessing this innovation potential would allow the production of software and services that would be more tailored to users' needs. The extant research shows that, due to the search and use of external resources and capabilities, the borders between networked companies and their environments are becoming blurred (Von Hippel and Von Krogh, 2003: http://tinyurl.com/7w46h24; Vujovic and Ulhøi, 2008: http://tinyurl.com/83za6ut; Elpern, 2009; http://tinyurl.com/76hj35f).

The CEO of OurDB clarifies that external resources are of key importance to the company's success:

> *"The vast community of [our OurDB product] users and developers is what drives our business – we have five million server installations in use worldwide. Around them there are small "software ecosystems." There are books and articles written, lectures held, courses taught, and applications developed around our products. This community of volunteers is our most important asset. Yet, it is difficult to define."*

Moreover, a respondent from Tripod described the context in which resources are publicly available, but then stated that the capability to make use of these resources to capture value potential is essential:

> *"With a large user group, you can attain a higher product quality, as a larger number of people use the software in different situations and provide feedback. It also limits your development costs, as you will receive some of the software from others. Sometimes, the greatest ideas come from outside – ideas that you never had thought of. Users often widen our view."* (Project manager, Tulip)

In a related exchange, the original developers share the code in the F/LOSS community, and the clients gladly rely on the developers' knowledge in applying the code in the development of applications, consultancy, training, and maintenance of the software.

According to our narratives, the ability to utilize external resources and capabilities is recognized as one of the key factors in remaining competitive in the software industry. As public goods, F/LOSS-based platforms, components, and applications shift the focus from the development of proprietary innovations to the use of the goods and knowledge that are publicly available.

Open source development encourages open innovation

The openness of innovation activity is a key theme in commercial F/LOSS development (Chesbrough, 2003: http://tinyurl.com/cgu9u7w; Watson et al., 2008: http://tinyurl.com/coemo53). The manager of Tripod argues that this kind of a joint project will succeed *"only if you let people see that their response has some effect on the software."* The responses depict a fundamental difference between open and closed innovation paradigms. *"There was a lot to do with our software before it was ready, but we opened in a very early stage. We were able to give plausible promise and thus received a lot of valuable feedback. This resulted in a quite different end product."* (Manager, Tripod)

Hence, the quality of innovation outputs is an essential reason to engage in open innovation. The open innovation approach embodies working together with numerous partners and various members of the F/LOSS community. In such an innovation model, the feedback loops are short and the software benefits from continuous improvement.

Our narratives underscore that, through F/LOSS activity, firms open their innovation processes to benefit from the knowledge and the innovation capacity of diverse open source communities. In this way, firms aim to benefit from the innovation capacity of both developers and users, resulting in shorter lead-times, shorter times to market, and ultimately, better product quality.

F/LOSS-based public goods change the revenue models

The CEO of OurDB claims that the entire business "will face a fierce price war, where profits disappear". A vital consideration in F/LOSS activity is how it changes the means of value capture in software businesses. During the interviews, our respondents tended to discuss services, rather than products. The respondents agree that proprietary software cannot compete successfully for long in the same market as a complementary F/LOSS product.

"Basically, our revenue streams are very broad and far-fetched. So, any reduction of problems in our service reduces cost and is actually an increase in our profit" (Project manager, Tulip). Our narrators discussed their revenue strategies at length, including the following statement from the CEO of OurDB: *"Enterprise software buyers are tired of complex pricing models (per core, per CPU, per power unit, per user, per whatever the vendor feels like that day) – models that are still in use by the incumbents. [With OurDb], you pay by service level and the number of servers. No nonsense, no special math."*

The case firms in this study have experimented both with revenue models that are directly product or service-related and those that indirectly benefit from the large user base:

> *"In the beginning we did not focus on profits at all. Instead, we focused on boosting the use of the software. The vast community of users and developers is what drives our business. Then we sell our offerings to firms – those who need to scale and cannot afford to fail. The enterprise offering consists of certified binaries, updates and upgrades, automated DBA services, 7x24 error resolution, etc."* (CEO, OurDB)

The narratives underscore that the software delivery price as such is not a key purchasing criteria, but the total cost of ownership, including all transaction costs and lock-in costs, is of greater concern. The manager of Yoga claims that the strength of F/LOSS is that, in many cases, it is the most cost-efficient solution:

> *"Some proprietary software companies communicate – and in some cases quite correctly – that the total cost of ownership of their products is lower the total cost incurred from an open source software installation. In some cases it may be true, but in others, the cost comparison clearly favors the open source software."*

However, our interviewees share the view that, when software is distributed freely, traditional revenue sources wane and firms are compelled to develop novel revenue models that may be based on services and may be only indirectly bound to the distribution of software licenses.

Conclusion

The lessons learned from the experiences of the entrepreneurs in this study indicate that F/LOSS-based software development urges software innovators to open up their innovation processes in a way that calls increasing attention to user involvement throughout the software development and delivery processes. That is, the F/LOSS activity emphasizes a need to maintain access to external capabilities, rather than leaning on internal resource ownership. In addition, entrepreneurs and managers should consider the objectives of the open innovation activity, because F/LOSS-based public goods change the focus of competition in the software business from product-centric to service-centric operations. These findings are summarized in Table 1 and are described in greater detail below.

The amount and nature of user involvement in software development and delivery, which is characteristic in the F/LOSS activity, would take many long-haul software entrepreneurs by surprise. In fact, our narratives indicate that rich social interactions with users may bring challenges even for experienced F/LOSS entrepreneurs. However, our informants are harmonious in describing the benefits of rich user interactions for gaining user insight and increased understanding of users' needs and wants, and they agree that these benefits clearly exceed the burden of managing extensive user interactions. These findings are in accordance with the arguments presented by Fitzgerald (2006; http://tinyurl.com/7zwsn8k). Moreover, user involvement may foster user commitment to the company's offerings.

In line with the finding of Vujovic and Ulhøi (2008; http://tinyurl.com/83za6ut), we found that the F/LOSS activity emphasizes access to external capabilities, rather than internal resource ownership. The key resources include developer capabilities and skills as well as readily available F/LOSS components. On the whole, maintaining access to relevant capabilities seems to be far more rewarding than efforts to assimilate new resources. This was grounded mainly on the reasoning that, if some companies provide their software as "public goods", the managers of all firms providing competing offerings must rethink the rational for maintaining in-house resources. The narratives express that increased utilization of external resources has long-term effects on the whole software industry. That is, companies that do not endorse F/LOSS will also face changes in regard to resource availability and costs.

Our informants described significant productivity enhancements related to collaborative innovation pro-

Table 1. Summary of findings including implications for managers of firms engaged in F/LOSS development

Essential themes	Key considerations	Managerial actions
Actions: User involvement	F/LOSS activity emphasizes user involvement in software development and delivery.[*]	Orient the firm toward richer social interactions with the users.
Means: Utilizing external resources	F/LOSS activity emphasizes access to external capabilities, rather than internal resource ownership.[†]	Maintain access to relevant capabilities rather than assimilate new resources.
Goals: Managing the open innovation processes	F/LOSS-based software development urges software innovators to open up their innovation processes.[‡]	Consider the purpose of external contribution in the innovation process.
Outcomes: Revenue models	F/LOSS-based public goods change the revenue models of firms taking part in open source development.	Grasp the logic of generating revenues from increasingly service-oriented offerings.

[*] Fitzgerald, 2006: http://tinyurl.com/7zwsn8k
[†] Vujovic and Ulhøi, 2008: http://tinyurl.com/83za6ut; Rajala, 2009: http://tinyurl.com/d6bkev9
[‡] Watson et al., 2008: http://tinyurl.com/coemo53

cesses. Our findings highlight that in order to capture the full potential from external contribution in open innovation, managers should try to engage the users or other external contributors already at the early phases of their innovation processes. However, the interviewees emphasized the importance of understanding the purpose such openness and to consider the aims for external contribution in the innovation process. If the goal for collaboration is clear, it would be easier to reveal even the once confidential information, as the innovators are surged forward with enthusiasm, bearing barely in mind that what parts of the innovation should be kept secret to ensure the future presence in the business. These findings are congruent with the views of Watson et al. (2008; http://tinyurl.com/coemo53).

F/LOSS-based public goods change the revenue models of firms taking part in open source development. In our data, it is evident that F/LOSS has an impact on the software industry as a whole, as it degrades traditional sources of revenue and compels firms to develop new revenue models primarily based on services. The interviewed executives recounted the need to grasp the logic of generating revenues from increasingly service-oriented offerings. This is congruent with prior studies on open source software business. Our findings are also consistent with existing studies on F/LOSS business models, and underscore that new revenue models are increasingly grounded on services. These F/LOSS-induced changes can affect all firms in the software industry and are not limited to companies engaged in F/LOSS activity.

Perspectives from Open Source Entrepreneurs

Juho Lindman and Risto Rajala

About the Authors

Juho Lindman is an Assistant Professor in the Hanken School of Economics in Helsinki, Finland. Juho defended his doctoral dissertation focusing on open source software development organization in the Aalto University School of Economics in Helsinki. In the field of information systems science, his current research is focused in the areas of open source software development, open innovation, open data and organizational change.

Risto Rajala is Director of Research in a multi-school collaboration platform in service research and education at Aalto University in Helsinki, Finland. His recent research has been centered on user-centric and open forms of managing software-intensive innovations, the transformation of software vendors' business models, IT services and the socio-technical aspects of service systems. Risto holds a PhD in Information Systems Science from the Aalto University, School of Economics. His work has been published widely in refereed scientific journals, international conference proceedings, academic books, and other scientific volumes. He is also a member of the Review Board of the *Technology Innovation Management Review.*

Citation: Lindman J. and R. Rajala. 2012. How Open Source Has Changed the Software Industry: Perspectives from Open Source Entrepreneurs. *Technology Innovation Management Review.* January 2012: 5-11.

Open Invention Network: A Defensive Patent Pool for Open Source Projects and Businesses

Deborah Nicholson

> *I think it is important to realize that technology defined as practice shows us the deep cultural link of technology, and it saves us from thinking that technology is the icing on the cake. Technology is part of the cake itself.*
>
> Ursula Franklin
> *The Real World of Technology*

This article explores how patents impact innovation within free/libre open source software (F/LOSS) businesses and projects. The number of software patent suits brought each year is increasing and is diverting millions of dollars in funds from developers to lawyers. With patent suits on the rise, the US Supreme Court has left the F/LOSS community in a position where it must either wait years for legislation or address the issue of patent suits itself. However, defending the Linux kernel and related technologies is a different challenge than the one that faces proprietary software businesses. This article describes Open Invention Network, an initiative that is designed to meet the particular challenges facing the F/LOSS community and businesses by providing a defensive patent pool.

Introduction

The threat of software patent suits impacts standards, dictates what software becomes part of GNU/Linux distributions, creates extra work, and makes the end-user's experience less than ideal, as will be shown in this article. In last month's issue of the *TIM Review*, Monica Goyal (2011; http://timreview.ca/article/503) thoroughly examined some of the legislative ideas being discussed with regards to patent reform. However, legislative change will take years to achieve. In the meantime, more software patent suits are brought about each year. F/LOSS companies are being sued by both proprietary competitors and non-practicing entities. The F/LOSS community needs a viable defense now.

In this article, we examine the role of software patents and their impact on open source projects and businesses. First, we focus on the general challenges related to software patents. Next, we examine the particular challenges software patents pose to open source projects and businesses. Finally, we discuss Open Invention Network (OIN), a defensive patent pool established to help Linux-based projects and businesses defeat or deflect the threat of litigation.

Software Patents in the United States

In 2010, the Supreme Court ruled on *Bilski v. Kappos* (http://tinyurl.com/7mo5vvx), a case considering whether a particular business method for hedging risk ought to be patentable. No case addressing the patentability of abstract ideas had been heard in twenty years. Many hoped to see the Court use this case to generally narrow the scope of what is patentable, and sixty-eight amicus briefs were filed in this landmark case. An amicus brief allows stakeholders can choose to act as a “friend of the court” and typically offers the stakeholder’s perspective on how the court's decision on a particular case is likely to affect them. F/LOSS businesses and many others pleaded with the Court to use *Bilski v. Kappos* to restrict what is patentable to a "machine or transformation," or alternatively to hand down some new doctrine that would put software patents (and perhaps business-method patents) outside the scope of patentability.

An earlier case, often referred to as simply "State Street", had established the "useful, concrete and tangible" doctrine, which required that there be a practical application for an invention to be considered patentable (http://tinyurl.com/7a2o7br). Most business method patents and software patents were believed to be outside the scope of the older doctrine although the lower courts had not upheld that idea. See Box 1 for a brief history of US patent law. For further details, see Patent Absurdity (http://patentabsurdity.com).

Ultimately, the Court ruled that Bilski's method was not patentable. Moreover, the Court chose not to take any kind of stand on what ought to be patentable; the majority opinion states:

> "*...patent law faces a great challenge in striking the balance between protecting inventors and not granting monopolies over procedures that others would discover by independent, creative application of general principles. Nothing in this opinion should be read to take a position on where that balance ought to be struck.*"

It would be hard for the Court to more thoroughly express their desire to maintain the current scope of patentability. In the decision, Judge Stevens spoke about patents in the information age, "If a high enough bar is not set when considering patent applications of this sort, patent examiners and courts could be flooded with claims that would put a chill on creative endeavor and dynamic change." Not only will nothing be done about software patents, the Supreme Court does not believe that there is a problem. Thus, the US courts have struggled to find a way to help investors make good on their investments while still promoting competitive innovation in a way that keeps pace with evolving technologies.

Non-Practicing Entities and Other Patent Challenges

Non-practicing entities (NPEs) are businesses that do not ship software or hardware or develop any sort of technology. These companies buy patents taken out by other companies; they sometimes purchase patents for current technology and sometimes for old technology, preferably if those patents include vague wording that could apply to other contexts. Some NPEs acquire pat-

Box 1. A brief history of US patent law

1952 The Amendment to the Patent Act Legislation added the word "process" to the list of what is patentable. Previous patents had been limited to manufacture and composition of matter.

1972 *Gottschalk v. Benson* (http://tinyurl.com/7cev3cl)
In 1972, the courts felt that algorithms should not be patentable, but this idea was slowly chiselled away over the next 38 years.

1978 *Parker v. Flook* (http://tinyurl.com/7nksmvy)
Mathematical algorithms are patentable if the implementation is "novel and non-obvious". This suit was about whether or not having some kind of trigger signal when a catalytic converter is operating outside certain desirable parameters ought to be patentable. In the end, the algorithm was not deemed patentable but the *application* of it was.

1981 *Diamond v. Diehr* (http://tinyurl.com/73f2n5d)
A computer program that controls a machine is patentable. This case was about software being used to control the process of curing rubber. Again, it is the *application* of the software in a novel way that makes this innovation patentable.

1982 The Court of Appeals for the Federal Circuit was set up to hear appeals based on subject matter, including patents. The upshot? Patents suits are largely presided over by patent attorneys and the case law that was created here gradually paved the way for the unfettered patentability of everything, including software.

1994 *In re Alappat* (http://tinyurl.com/75z3dvn)
Installing software on a computer makes a new machine which is patentable. This is often derided as the Piano roll blues, from Judge Rich's observations that a player piano is the same device no matter what roll of music making paper is loaded on it.

1998 *State Street Bank* (http://tinyurl.com/7a2o7br)
The useful, concrete, and tangible doctrine came from this case. This was an attempt to exclude business method patents from the realm of what is patentable. Both concrete and tangible had potential to also knock software out of the pool of what is eligible for patentability as well, but this case was not upheld.

2010 *Bilski v. Kappos* (http://tinyurl.com/7eep7cl)
The "machine or transformation test" is not the only valid test for patentability. The bench decided that they would not narrow the scope of patentability at this time.

ents in areas that they believe other companies may be moving towards. Seventy-five percent of the suits brought by NPEs are software suits.

Ostensibly, the NPE business model is to help individual inventors or very small firms to manage their "intellectual property." However, their main source of income is filing lawsuits. Just fourteen NPEs raked in a combined $7.6 billion from 2000 to 2010. That figure represents only 9% of what the companies who were sued actually lost; defendants in those suits lost an estimated $87.6 billion in litigation costs and lowered stock value (Bessen et al., 2011; http://tinyurl.com/7jtap6x).

In 2010, the number of companies in all realms (including software) that found themselves in litigation with an NPE increased by an average of 48% when compared to the average of the previous three years (PatentFreedom, 2011; http://tinyurl.com/8ax9n3f).

Litigation is expensive, and so many companies settle. This ensures that a poor-quality or out-of-date patent can continue to be used to sue other companies. Many times, an NPE will sue using the same patent again and again. Fighting these types of suits can help knock out bad patents, but the cost is high.

In addition to the challenges posed by NPEs, there are also suits brought by other software vendors hoping to squash, annoy, or perhaps assimilate their competition. For example, in 1997, Intel sued a microprocessor competitor called Cyrix (http://tinyurl.com/6tsv5wy). Four years of litigation later, Cyrix "won" the suit, but they missed the opportunity to make money on their innovation. Technology moves faster than lawsuits, and the time for that particular microprocessor had passed.

Patent lawsuits are costly, even for the winners. According to James Bessen and Michael Meurer in *Patent Failure* (2008; http://tinyurl.com/6m8zf7o), a lawsuit that does not go on for too long can "cost only one-half million to a million dollars" and a case that goes to trial can cost "several million dollars" while, "in extreme cases, legal costs can mount to tens of millions." Those figures are enough to start another company or sink an existing one. Companies that are being sued will often see their stock prices plummet, while also suffering indirect costs due to the distractions a lawsuit brings. Money and energy are being diverted to legal battles from software development, project management, sales, support, and community outreach. All these costs can make the difference between success and failure.

Does the money that is exchanged in lawsuits ultimately fund innovation at another company once the lawsuit is over? As found by Bessen and colleagues (2011; http://tinyurl.com/7jtap6x), the answer for lawsuits brought about by NPEs is no: "most of the private losses incurred by defendants in NPE litigation do not appear to be transfers to other parties." It is clear that patent suits are not good for the business being sued, but the more important question is whether or not they are good for the industry as whole or even more broadly for society. The evidence does not support the theory that NPE activity is good for business or for innovation. "While the lawsuits might increase incentives to acquire vague, over-reaching patents, they do not increase incentives for real innovation" (Bessen et al., 2011). Promoting innovation is the supposed goal of the US patent system.

F/LOSS Companies and Projects

Patent lawsuits are not challenges for proprietary software companies alone; F/LOSS projects and companies may also be targets for litigation. The entities that are the most tempting targets are those that generate substantial revenue, such as Red Hat (http://redhat.com) and Google (http://google.com). In many cases the success of smaller F/LOSS projects depends on upstream success; imagine the GNOME desktop environment (http://gnome.org) without a major operating system distribution, or imagine Android applications without the Android platform. Also, many large F/LOSS projects depend on a closely related volunteer community, which represents a considerable asset that cannot be converted into a legal department or liquidated to fight a lawsuit.

Smaller projects are less likely to be sued, but patent concerns are still often harmful and time-consuming. For example, the GIMP photo-editing project (http://gimp.org) no longer includes the image mosaic plug-in after its developer received a letter alleging patent infringement. Says Peter Kirchgessner who has developed a number of GIMP plug-ins: "It is not clear if the patent is applicable in this case. But I have neither the time, interest or money for legal action. So I complied with the cease and desist request." (http://tinyurl.com/2emqbz). Even without a letter, the huge legal fees associated with software patents suits creates a chilling effect in certain areas or can consume large amounts of volunteer time to avoid hot spots. In another example, the Wine project (http://winehq.org), which allows GNU/Linux users to run Windows applications, has been forced to eliminate a critical feature. “Concerns about the Borland patent have prevented developers on the project from adding structured exception handling (SEH) to the free software compiler.” (http://tinyurl.com/7nlm2m3) Removing SEH leaves developers in the situation of depending on a non-free tool, a licensing problem for free software distributions or writing a costly work-around. Even an unproven or vague suggestion of patent infringement can create a significant amount of additional work for a small project.

Even when no lawsuit is brought, the threat of a suit can cause problems for F/LOSS projects and companies. The 2009 debate over video formats for the web is a prime example of how patents can negatively impact end users. Apple worried that the Theora video compression format may be patent encumbered, which effectively stopped the adoption of Ogg Theora as the official HTML5 video codec (http://tinyurl.com/p8kfce). The firm MPEG LA has implied that all video standards are likely to infringe on existing patents (http://tinyurl.com/24yjld4). MPEG LA licenses related patents, so it is in their interest to make others wary of potential infringement and encourage them to pay licensing fees to use the technology. Would close scrutiny reveal that the Ogg Theora codec infringes on existing patents? Until a suit goes to court, there is no way to be sure. Meanwhile, the potential for patent infringement prevents content creators from using a single format that can be processed by all major browsers and developers for projects such as Fedora, Blender, and Miro spend time carefully excluding certain types of video support that would benefit their users (http://tinyurl.com/nelhsk). Also, lawsuits brought against one project can create work for other projects, result in exclusions to their final products, and ultimately impact their competitiveness in the market.

In light of the tremendous money to be made from patent suits, one might think F/LOSS projects ought to just “play the game” and start suing other companies for patent infringement. However, many free software contributors consider patent aggression morally repugnant. A company or project that relies on community support would endure a lot of backlash if it were seen to be a patent aggressor, especially if its actions negatively impacted other F/LOSS projects. F/LOSS communities differ from proprietary software businesses in several important ways, the foremost being motivation. Developers may just be “scratching their own itch” (i.e., working to solve a problem they personally experience) or they may be working to provide the wider community with a solution that may not be met by proprietary software, regardless of the community’s ability to pay. Ethical concerns over control and access to computing motivate many contributors. These various motivations lead to different project structures and business models, filling every point on the spectrum from reliance on unpaid community members to fully funded staff. Most software projects are a hybrid, with community members moving from one project to another; some community members may be paid, some may not be. Community goodwill is critical for success in the F/LOSS world and its culture makes a strategy based on patent aggression unworkable.

Furthermore, the reluctance to wield patents as a weapon is often contractual. Many free software licenses have addressed patent aggression in their terms. The latest version of the GNU General Publish License (GPLv3; http://gnu.org/licenses/gpl-3.0.html) forbids a company from making patent infringement claims related to code that it contributed to a project under that license. The GPLv3 also asserts that patent rights that are extended to one recipient of GPL code must be extended to all recipients of that code. The Apache License (http://apache.org/licenses/) and the Mozilla Public License (http://mozilla.org/MPL/) also include clauses discouraging use of code under their license being used as a basis for a patent infringement suit. Apache terminates your license when litigation is filed:

> *“If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.”*
> (http://www.apache.org/licenses/LICENSE-2.0)

Many F/LOSS projects have neither the willingness nor the legal leeway to recoup losses from patent infringement suits by bringing suits against other software projects. For the free software community, the rise of software patent suits is a nuisance, not an opportunity.

As annoying as software patent suits are for F/LOSS projects, free software does not present a higher risk for infringement compared with proprietary software. As Dan Ravicher (2004; http://tinyurl.com/87ltxfk) points out, free software is at the same risk, since patents cover the idea or function rather than copyright, which covers the actual lines of code. Proprietary software and free software perform many of the same functions so both types of projects are equally vulnerable to suits for infringement. The good news is that a significant amount of code is being used by many projects, including some with significant resources to protect. For example, the number of projects using Autoconf (http://tinyurl.com/2psee4), the X window system (http://tinyurl.com/4mt9y), or OpenGL (http://tinyurl.com/733y8mo) is vast.

When a suit is brought, the court can choose to issue what is called preliminary injunction, or a mandate to stop the activity that is objectionable to the prosecution before the case is heard in full. As Ravicher (2004) points out, a preliminary injunction against a particular program would be impossible to enforce and there is no way to obtain a meaningful estimate of how many copies of any given piece of code are out in the world. The defendant would be unable to comply with such an order. A permanent injunction can be handed down after a suit is decided. After such a decision, the community would need to code around that particular patent. It is far better for broad and vague patents to be overturned through effective defense and prior art (http://wikipedia.org/wiki/Prior_art). Prior art refers to anything that has been made available to the public regarding a particular invention, including anything that proves an invention is obvious or not novel. Prior art can keep bad patents from being issued, overturn recent wrongly issued patents, and help with a pending lawsuit.

For F/LOSS projects and companies, lawsuits consume vast amounts of money and time that could be better spent on development, promotion, documentation, or translation. Pamela Jones from Groklaw says: "Knocking a patent infringement case out depends on having the precise weapons to do so. You can't fight something with nothing. If they are going to aim patents at you, you can't just stand there and hope for the best." (http://tinyurl.com/3xj5brl).

Open Invention Network

To help F/LOSS companies and projects overcome the challenges of patent lawsuits in a way that is compatible with the culture of free software, Open Invention Network (OIN; http://www.openinventionnetwork.com) was launched in 2005. OIN is an intellectual property company that was formed to further software innovation and promote Linux by using patents to create a collaborative ecosystem. OIN established a defensive patent pool to help F/LOSS projects, particularly those associated with Linux. OIN does not seek revenue by asserting its patents, but rather its intent is to allow community members to use its patents in a defensive way against those who attack Linux. Patents owned by OIN are available royalty-free to any company, institution, or individual that agrees not to assert its patents against Linux and related technologies. This enables companies to make significant corporate and capital expenditure investments in Linux – helping to fuel economic growth. OIN is backed by investments from IBM, NEC, Novell, Philips, Red Hat, and Sony. These six companies decided it would be mutually beneficial if they agreed not to sue each other over Linux and related technologies.

An example of OIN's role comes from late February 2009, when Microsoft filed a patent infringement suit against TomTom on eight patents, including three related to File Allocation Table (FAT) technology. Microsoft simultaneously sought an US International Trade Commission injunction against TomTom shipping product into the United States. TomTom reached out to OIN, as well as Linux Foundation and Software Freedom Law Center, for assistance with the suit. On March 23, 2009, OIN publicly distributed a press release indicating that TomTom had joined the OIN community of licensees. Microsoft settled the suit against TomTom shortly thereafter. TomTom was not required to disclose the terms of its settlement with Microsoft because the terms were deemed to be "nonmaterial" based on disclosure requirements in the Netherlands. Many believe that this particular suit was brought just to scare Linux kernel users. Bruce Perens observed: "What Microsoft really wants from TomTom isn't money, it's support in building fear about Linux in other companies, especially the makers of mobile and wireless devices just like TomTom's own product." (http://tinyurl.com/cq8d7v). There is a struggle going on for what kind of software we will use in the future. Given that lawsuits are expensive, the courts represent a stacked deck for the wealthier litigant.

In another example, Red Hat and Novell were sued in 2007 by IP Innovations, an NPE that owns 536 patents (http://tinyurl.com/76svjho). OIN supported the search for prior art to help invalidate the three patents using Linux Defenders (http://linuxdefenders.org), an online clearing house for prior art. Post-issue prior art, a term referring to evidence garnered after a patent has been issued, was crowdsourced from the community. Three junk patents based on X windowing systems from 1987 were knocked out (http://tinyurl.com/2g9jumu). IP Innova-

Open Invention Network: A Patent Pool for Open Source Projects and Businesses

Deborah Nicholson

tions will not be able to sue anyone else over those specific patents, but there are still many more to be struck down. It is notable that IP Innovations is a subsidiary of Acacia Technologies; there has been some speculation about the relationship between Acacia and Microsoft (http://tinyurl.com/8x3vqxw), which could mean deep pockets in addition to many technology patents.

Given the interconnections between F/LOSS projects, OIN would like more projects to become licensees so the F/LOSS community can focus on the external threats as a united front. For F/LOSS companies and projects, this means that OIN's defensive patent pool may be licensed for free. It is in the interest of our founding companies to see suits against the F/LOSS community defended adequately. Future cases over the same patents may refer back to decisions made in previous suits. Precedents built by suits against companies unprepared to fight back hurt the whole community.

OIN is staffed by a small group of F/LOSS community members, attorneys, coders, and outreach personnel who support OIN while also participating in other segments of the community. As with many other examples of the F/LOSS community working together on shared goals, it is impossible to gauge how much mutual success each organization is responsible for. OIN's success cannot be quantified as a separate element from the overall community's continued success. Given the current environment, where patent aggression is on the rise, OIN is proud to play its role in mitigating the risk of patent aggression to the Linux system.

Conclusion

Patent aggression exacts a substantial toll. As calculated by Bessen and colleagues (2011; http://tinyurl.com/7jtap6x), defendants in lawsuits with NPEs lost an estimated $87.6 billion in litigation costs and lowered stock value. Consider the social utility of $87.6 billion worth of coders, designers, and builders. If those suits are being brought strategically to erode the resources of the F/LOSS community, then this is a fight for the viability of free software. If the courts are not motivated by this cause, then another way must be found, such as that offered by OIN: a defensive strategy for F/LOSS projects and companies.

About the Author

Deborah Nicholson works at the intersection of technology and social justice. She has been a free speech advocate, economic justice organizer, and civil liberties defender. After working in Massachusetts politics for fifteen years, she became involved in the free software movement. She is the Community Outreach Director at Open Invention Network and the Community Manager at Media Goblin. She also serves on the board at Open Hatch, a nonprofit dedicated to providing tools and education for potential new free software contributors. She lives in Cambridge, Massachusetts where she is slowly pursuing a graduate degree in Information Technology.

Citation: Nicholson, D. 2011. Open Invention Network: A Defensive Patent Pool for Open Source Projects and Businesses. *Technology Innovation Management Review*. January 2012: 12-17.

A New Way of Measuring Openness: The Open Governance Index

Liz Laffan

“ *How open is open enough?* ”

Joel West
Professor of Innovation
and Entrepreneurship

Open source software is now “business as usual” in the mobile industry. While much attention is given to the importance of open source licenses, we argue in this article that the governance model can be as necessary to a project’s success and that projects vary widely in the governance models – whether open or closed – that they employ. Open source governance models describe the control points that are used to influence open source projects with regard to access to the source code, how the source code is developed, how derivatives are created, and the community structure of the project. Governance determines who has control over the project beyond what is deemed legally necessary via the open source licenses for that project. The purpose of our research is to define and measure the governance of open source projects, in other words, the extent to which decision-making in an open source project is “open” or “closed”. We analyzed eight open source projects using 13 specific governance criteria across four areas of governance: access, development, derivatives and community.

Our findings suggest that the most open platforms will be most successful in the long term, however we acknowledge exceptions to this rule. We also identify best practices that are common across these open source projects with regard to source code access, development of source code, management of derivatives, and community structure. These best practices increase the likelihood of developer use of and involvement in open source projects.

Introduction

Much has been written and debated regarding open source licenses – from the early days of the GPL license to the modern days of the Android open source platform. Yet we believe that there is one very important aspect of open source projects that has been neglected: open source governance models. While licenses determine rights to use, copy, and modify, governance determines the rights to visibility, influence, and derivative creation (Table 1). And while licenses apply to the source code, governance applies to the project or platform. More importantly, the governance model describes the control points used in an open source project – such as Android, Qt, or WebKit – and is a key determinant in the success or failure of a platform.

The governance model used by an open source project encapsulates all the hard questions about a project. Who decides on the project roadmap? How transparent are the decision-making processes? Can anyone follow the discussions and meetings taking place in the community? Can anyone create derivates based on that project? What compliance requirements are there, and how are these enforced? Governance determines who has influence and control over the project or platform – beyond what is legally deemed in the open source license. In today’s world of commercially-led mobile open source projects, it is not enough to understand the open source license used by a project. It is the *governance model* that determines whether or not decision making within an open source project is open, accessible, and transparent to all users or whether it is concentrated amongst a specific set of users.

A New Way of Measuring Openness: The Open Governance Index

Liz Laffan

Table 1. Key differentiators of open source licenses versus governance

	License	Governance
Rights	Use, copy, modify	Visibility, influence, and creation of derivatives
Use	70% of projects use one of seven licenses	No agreed definition of governance
Examples	GPL, LGPL	No formal examples
Legal	Legally binding	Non-binding

Pisano and Verganti (2008; http://tinyurl.com/67bc3b) characterized open source projects as examples of an "open collaboration model" that is both open (membership) and flat (governance). Based on this view, there is an expectation that open source projects will also be governed openly. However, our findings suggest that some open source projects – such as Android, Qt, and Symbian – employ closed (hierarchical) governance models and that governance models can change over time. While Pisano and Verganti characterize governance models are either flat or hierarchical, we employ the term "open" in reference to the degree to which a project's decision-making processes are open to the community. For example, identifying who the decision makers are within open source projects (transparency) and accessing information around the actual decision-making process (accessibility) are governance criteria that are not readily captured in describing governance models as either flat or hierarchical.

In this article, we firstly explain the key governance criteria that we used to analyze eight different mobile open source projects and the outcome of this analysis. We then examine why Android has been so successful given that we find it is also the least open mobile open source project. Following from this, we identify best practices used by the most successful open source projects across the four governance areas of access, development, derivatives, and community. Finally, we suggest areas for future research and provide some conclusions regarding our research findings to date.

Analysis of Governance Models

We set out with an ambitious goal: to measure openness – the degree to which an open source project is "open" or "closed" – in ways that are rarely discussed publicly or covered in its license. We set out to define and measure the governance of open source projects in a transparent and comprehensive manner – much like how open source licenses are defined and classified into "copyleft", "permissive", and so on. Unlike open source licenses, the governance model is made up of less visible terms, conditions, and control points that determine access, influence, decisions, and derivatives of that project.

We researched eight mobile open source projects: Android, MeeGo, Linux, Qt, WebKit, Mozilla, Eclipse, and Symbian. We selected these projects based on breadth of coverage; we picked both successful (Android) and unsuccessful projects (Symbian); both single-sponsor (Qt) and multi-sponsor projects (Eclipse); and both projects based on meritocracy (Linux) and on membership status (Eclipse).

Our research, carried out over a six-month period, included analysis of these popular open source projects and conversations with community leaders, project representatives, academics, and open source scholars. West and O'Mahony (2008; http://tinyurl.com/66fly95) identified three dimensions of open source projects: production (of source code), governance (of the open source project), and intellectual property (of the source code produced by the project). We build upon this work by also investigating how users (developers) of the project source code can influence the direction and content of the open source project through the accessibility and transparency of the decision-making processes and governance of the open source project. For example, we show how the management of source code contributions is a critical control point for governance of an open source project. Additionally, we have focused on how derivative source code (i.e., applications that can run on the open source project platform) is controlled; this is an important governance control point that is being exploited by commercial organizations supporting open source projects. Therefore, our focus has been very much on the use of the governance models as a descriptor of open source control points.

Based on our analysis, we published a report in which we proposed the Open Governance Index (OGI), a measure of open source project "openness" (Vision Mobile, 2011; http://www.visionmobile.com/research.php#OGI). The OGI comprises 13 metrics (Box 1) across the four areas of governance:

1. **Access:** availability of latest source code, developer support mechanisms, public roadmap, and transparency of decision making

A New Way of Measuring Openness: The Open Governance Index

Liz Laffan

2. Development: the ability of developers to influence the content and direction of the project

3. Derivatives: the ability for developers to create and distribute derivatives of the source code

4. Community: a community structure that does not discriminate between developers

The OGI quantifies how open a project is in terms of transparency, decision making, reuse, and community structure. We ranked projects across each governance parameter and on a scale of one to four on each question from Box 1. The higher the score, the more open the project. Details on how the OGI is computed, including individual scores for each project against the 13 governance criteria, are available in the full report (VisionMobile, 2011; http://www.visionmobile.com/research.php#OGI). Also note that our assessment of Qt was done before the project's governance model was revised in October 2011.

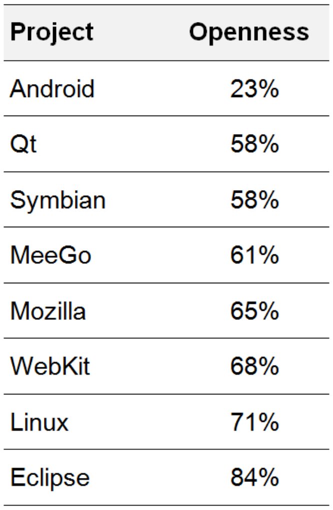
Are "Open" Projects More Successful?

A successful open source project demonstrates long-term involvement of users and developers, along with a substantial number of derivatives, and the project continually develops, matures, and evolves over time. Our research suggests that platforms that are most open will be most successful in the long-term. Eclipse, Linux, WebKit, and Mozilla each testify to this through their high OGI scores (Table 2). In terms of openness, Eclipse is by far the most open platform across access, development, derivatives, and community attributes of governance. It is closely followed by Linux and WebKit, and then Mozilla, MeeGo, Symbian, and Qt. Seven of the eight platforms reviewed fell within 30 percentage points of each other in the OGI.

Our research has identified certain attributes of successful open source projects. These attributes are: timely access to source code, strong developer tools, process transparency, accessibility to contributing code, and accessibility to becoming a committer. Equal and fair treatment of developers (i.e., "meritocracy") has become the norm and is expected by developers with regard to their involvement in open source projects.

We also note that there are common areas where most open source projects struggle to be "open". These attributes coalesce around decision making with regard to the project roadmap and committing code to the project. In particular, we find that open source projects that originate from commercial organizations struggle most with relinquishing project control, which is not surprising, considering the structured and hierarchical decision-making structure of most organizations.

Table 2. Open Governance Index results



Project	Openness
Android	23%
Qt	58%
Symbian	58%
MeeGo	61%
Mozilla	65%
WebKit	68%
Linux	71%
Eclipse	84%

The Android paradox

Android ranks as the most closed project we examined, with an OGI score of 23%. Yet, at the same time, it is one of the most successful projects in the history of open source. Is Android proof that open governance is not needed to warrant success in an open source project?

Android's success has little to do with the open source licensing of the public codebase. Android would not have risen to its current ubiquity were it not for Google's financial muscle and famed engineering team. Development of the Android platform has occurred without the need for external developers or the involvement of a commercial community.

Google has provided Android at "less than zero" cost, since its core business is not software or search, but driving ads to eyeballs. As is now well understood, Google's strategy has been to subsidize Android such that it can deliver cheap handsets and low-cost wireless Internet access in order to drive more eyeballs to Google's ad inventory.

More importantly, Android would not have risen were it not for the billions of dollars that OEMs and network

A New Way of Measuring Openness: The Open Governance Index

Liz Laffan

Box 1. OGI Governance Criteria

Access

1. Is source code freely available to all developers, at the same time?

2. Is source code available under a permissive OSI-approved license?

3. Developer support mechanisms – are project mailing lists, forums, bug-tracking databases, source code repositories, developer documentation, and developer tools available to all developers?

4. Is the project roadmap available publicly?

5. Transparency of decision mechanisms – are project meeting minutes/discussions publicly available such that it is possible to understand why and how decisions are made relating to the project?

Development

6. Transparency of contributions and acceptance process – is the code contribution and acceptance process clear, with progress updates of the contribution provided (via Bugzilla or similar)?

7. Transparency of contributions to the project – can you identify from whom source code contributions originated?

8. Accessibility to become a committer – are the requirements and process to become a committer documented, and is this an equitable process (i.e., can all developers potentially become committers?). Note that a "committer" is a developer who can commit code to the open source project. The terms "maintainer" and "reviewer" are also used as alternatives by some projects.

9. Transparency of committers – can you identify the committers to the project?

10. Does the contribution license require a copyright assignment, a copyright license, or patent grant?

Derivatives

11. Are trademarks used to control how and where the platform is used via enforcing a compliance process prior to distribution?
12. Are go-to-market channels for applications derivatives constrained by the project in terms of approval, distribution, or discovery?

Community Structure

13. Is the community structure flat or hierarchical (i.e., are there tiered rights depending on membership status?)

operators poured into Android in order to compete with Apple's iconic devices. As Stephen Elop, CEO of Nokia, said at the Open Mobile Summit in June, 2011, "Apple created the conditions necessary for Android".

However, there are some very good lessons to learn from Google's management of the Android open source project. First, Android was released as an open source project at a point in time where it was already a very advanced, complete project. OEMs, operators, and software developers could more or less immediately use it to create derivative handsets and applications. Second, Google kickstarted a developer buzz around the project with the $10 million Android Developers Challenge. Alongside financial incentives, Google sent an alluring message by opening application development within a previously inaccessible mobile industry. Finally, Google's speed of innovation (e.g., five platform versions were released in 2010) outpaces any external innovation and makes the ecosystem entirely reliant on Google.

Best Practices

Based on our research of major mobile open source projects, we have outlined the best practices for governance models. These practices are listed across the four key areas of governance: access, development, derivatives, and community.

Access
The minimum requirement for any project to be an open source project is *source-code access* such that developers can easily read, download, change, and run the code. There should be no developer discrimination; all source code should be available to all developers in a timely manner. Restrictions with regard to source code should be at a minimum, and there should be no preferential access to specific developers because this can cause friction and lead to branching of the project. All open source projects should use open source licenses that are approved by the Open Source Initiative (OSI; http://www.opensource.org).

The next most important requirement is *ease of access to developer tools, mailing lists, and forums*, such that developers can get up to speed on the specifics of the project and build and run the code with minimum effort.

Development
As much as possible, a *simple code contributions process* should operate freely and without any hindrance. While we appreciate valid intellectual property concerns, such as the risk of copyright infringement, these should not complicate the contributions process any more than necessary. We also note that none of the projects reviewed in this study mandate copyright assignment; this is a good example of why copyright assignment is largely unnecessary. A broad copyright (and ideally patent) license for use of the work should suffice, provided the project has researched and identified the appropriate open source license under which to distribute the project. Copyright assignment is only ever needed when the project decides to change the terms under which it licenses the source code of the project, and this should be largely unnecessary, provided that the correct open source license is identified in the first place.

Given that the success of open source projects is largely based on the accrual of developer interest and support, we identify the *transparency of decision-making and equitable treatment of all developers* (such that they can become project committers) as being critical to long-term success. Restriction of commit rights to specific developers or organizations is a sure way to lose developer support in the long run because developers become frustrated with the inability to commit code themselves, especially if their contributions are continually rejected or ignored.

Developers often need to know where the project is headed, how it will get there, and why it is headed in that direction. They also often want the opportunity to influence the project to meet their own needs (i.e., to "scratch their own itch"). The main means by which developers can achieve this influence is by being able to commit code to the project. Therefore, it should be possible for all developers to commit code to the project, once they have shown sufficient knowledge of the code to do so. This is where *meritocracy* comes into play: those that "do" should be rewarded accordingly. Additionally the project should provide *transparent project metrics* regarding where contributions come from and who committed them.

With regard to the actual development process itself, the project should have a *policy of contribution to up-stream projects firs*t (if the project comprises other open source projects) such that changes and benefits accrue to up-stream and down-stream projects.

Derivatives
Compliance frameworks are becoming more and more common among open source projects in order to deter fragmentation and ensure that applications are trans-

ferable across multiple platforms or operating systems. However, the best mechanism to keep compliance requirements honest is to *make the compliance process as independent and transparent as possible* such that it cannot be manipulated by any one developer or organization. For example, MeeGo has asked the Linux Foundation to manage its trademark compliance requirements so that they are independent of the project.

Community
A number of the projects we reviewed use a not-for-profit foundation structure to provide independence, such that the platform is not controlled by any one organization. Other projects have established a formal association with the Linux Foundation, and this lends strong "open source credibility" to the project.

Another aspect of open source communities is the method by which authority is exercised within the community. For example, we note that both Linux and Mozilla use the benevolent dictator model, where decisions regarding disputes are made by one person. Whilst this process may work, it is still centralization of authority and decision-making, and as such it does not easily allow for others to permeate this decision-making process.

Evolving the Open Governance Index

We aim to continue the discussion on governance, to refine our criteria even further, and to make the OGI measure as meaningful as possible for the open source community. One of the first suggestions has been with regard to having a time dimension to the criteria (i.e., does openness change over time). Mature open source projects such as Eclipse, Linux, and WebKit that have stood the test of time, score quite highly with regard to openness of governance. But this has not always been the case. For example consider the following. Apple forked KHTML to create WebKit in the early 2000s, releasing the first WebKit open source project in 2005 but with reviewer and commit rights restricted to Apple personnel only which effectively sidelined the KDE community. In 2007 however Apple reversed this decision allowing allow non-Apple developers to have full commit access to the WebKit source code version control system. This shows that openness can change over a project lifecycle.

Our vision for the Open Governance Index is to for it to be a robust, and as much as is possible, an objective measure of governance for open source projects. We believe that this is necessary such that users and contributors to open source projects, including commercial entities, understand the means by which they can, or cannot, influence the direction and content of the project.

Conclusion

Today, open source software is "business as usual" in the mobile industry. It is proven that open source platforms such as Android can be as successful as proprietary platforms in terms of platform adoption, device sales, and applications development. And while open source plays a key role in developer attraction, it does not predetermine success. The mobile open source project space is undergoing consolidation to the extent that:

1. Symbian is no longer an active project, having been closed by Nokia and brought in-house while Nokia refocuses its effort using the Windows Mobile platform.

2. Nokia sold the commercial licensing rights for Qt to Digia in March 2011 and advised in November 2011 that they would "abnegate ownership" of Qt to focus on being maintainers only.

3. MeeGo is no longer being actively supported by either Nokia or Intel as an open source project, although parts of the MeeGo project are being used in the newly launched Tizen open source platform, which was launched in September 2011.

This consolidation does not detract from the fact that the mobile open source platforms can be very successful – witness Linux, Eclipse, and Android – but it does reiterate the importance of organizational support to the success of any open source project and community. To become a successful opens source project we find that there are best practices, as we have detailed in this article, which should be used to provide the best possible likelihood for success.

"Open governance" goes hand-in-hand with "open source"; it is about ensuring that developers and users have equal freedoms not to just use, but also to modify and build on the project. In many ways, open governance is the missing piece the open source licenses do not cover. Clearly, an open source license alone does not make an open project. It takes an open governance model as well. We hope our research is a step towards a fundamental change in the common understanding of how open source projects are managed and directed, including transparency regarding how decisions are made in open source projects..

A New Way of Measuring Openness: The Open Governance Index

Liz Laffan

The research described in this article was partially funded by webinos, an EU-funded project under the EU FP7 ICT Programme (#257103). For further background information and details of the analysis of each project, see the full report on which this article is based: "Open Governance Index: Measuring the True Openess of Open Source Projects from Android to WebKit" (VisionMobile, 2011; http://www.visionmobile.com/research.php#OGI*).*

About the Author

Liz Laffan is a Research Partner at VisionMobile. Liz has been working in the telecoms and mobile industry for over 20 years, with large telecommunications organizations, startup technology ventures, software development, and licensing firms. She holds a BA (Hons) in Business Studies from Thames Valley University and an MA in International Political Economy from Warwick University. Liz's interests lie in open source software governance and licensing and in particular how commercial organizations can best interact with open source projects.

Citation: Laffan, L. 2012. A New Way of Measuring Openness: The Open Governance Index. *Technology Innovation Management Review.* January 2012: 18-24.

Q&A

Ruth Suehle

Q. *What is the secret of Red Hat's Success?*

A. In 1993, Red Hat Linux was one of the first Linux distributions, along with Debian and Slackware. That was still five years before the term "open source" was coined. Linux and open source software were not widely known for many more years, much less trusted by large companies or recognized by future competitors. As late as 1999, while promoting his book, Bill Gates remarked about Linux, “Certainly we think of it as a competitor in the student and hobbyist market. But I really don't think in the commercial market, we'll see it in any significant way." (http://tinyurl.com/7ka28ab) He criticized open source for its lack of central control. Fast-forward to 2011, when the Microsoft Openness blog quoted CEO Steve Ballmer: “Our goal is to promote greater interoperability, opportunity and choice for customers and developers throughout the industry by making our products more open and by sharing even more information about our technologies” (http://tinyurl.com/2lf9ok). The world is changing.

Today open source software is not a hobby or a threat, it is simply reality for the technology industry. It is also no longer just about developing code. The Red Hat company used open source to create a business model that brings the company nearly a billion dollars in annual revenue. CEO Jim Whitehurst has called that model "the most powerful thing about what we've accomplished." The company is listed on the New York Stock Exchange and the S&P 500 and has grown a long list of offerings beyond Linux, from middleware to virtualization, training, and consulting.

All of this has been built on the belief that open source is more than a way to develop software. It is a way to run a business and the best way to participate as a part of the global community. Red Hat has grown through the power of collaboration, not just on source code, but on everything it does.

Red Hat sees the opportunity for the principles that have made open source successful as a business model to change the world, and the company intends to help make it happen through promoting principles like transparency, collaboration, diversity, and rapid prototyping, collectively something it calls “the open source way.” These characteristics can – and will – change everything about our world in the same way the open source model has changed how software is created, based on a few key openness concepts:

1. An open exchange. A free exchange of ideas is critical to creating an environment where people are allowed to learn and use existing information toward creating new ideas.

2. The power of participation. When we are free to collaborate, we create more. We can solve problems that no one person may be able to solve on their own, and we can create solutions that will be applied in ways we did not imagine.

3. Rapid prototyping. Rapid prototypes can lead to rapid failures, but that leads to better solutions found faster. When you are free to experiment, you can look at problems in new ways and look for answers in new places. You can learn by doing.

4. Meritocracy. In a meritocracy, the best ideas win, and the best features make it into the end product. In a meritocracy, everyone has access to the same information. Successful work determines which projects rise and gather effort from the community.

5. Community. Communities are formed around a common purpose. Together, a global, open community can create beyond the capabilities of any one individual. It multiplies effort and shares the work.

But how does all that apply to Red Hat’s success? It starts with the subscription.

The Red Hat Subscription Model

Although "open source" is now clearly about more than the code, to understand how it created Red Hat’s success, we have to start where it began: with the code.

Q&A: What is the secret of Red Hat's Success?

Ruth Suehle

All software has source code. As we know, not all software creators choose to share that code. But when they do, it means freedom and choice for the user. Industries can no longer operate in silos – neither the companies within a single industry, nor industries apart from one another. The world is becoming only more connected. What one company needs today, another company needs tomorrow. And what that company needs tomorrow might change the world in an entirely unrelated field. We have seen it over and over again when one person or organization creates a piece of code for its own needs, shares it, and another organization is able to reuse it in unforeseeable ways. That is the value of open source, and it happens every day.

Red Hat believes that because of that value, open source is inevitable, because it puts the choice and control in the hands of the customer, and Red Hat accomplishes the combination of open source value with business profit through its subscription model – the enterprise complement to the rapid innovation of open source development.

Open source empowers impressive innovation and rapid change. But if you are running a production environment, innovation and rapid change are frightening words. So Red Hat takes thousands of packages, freezes the code, and creates an enterprise-ready edition of that software, working with chip designers, hardware vendors, and independent software vendors to certify and tune the hardware and software that Red Hat products will work with. Then we back it with a promise of support for seven years, bringing the strength of open source to a level of security that is right for the stability production environments need – enough stability for many of the world's stock exchanges to run on Red Hat Enterprise Linux. So while traditionally licensed software loses its value as it ages, subscription software continues to support an infrastructure with updated features, security enhancements, and increased hardware and software support, not to mention predictable costs.

The Importance of Community

But of course, all of that value that Red Hat is able to offer its customers is built on the contributions of the much larger open source community, both as a whole, and the specific communities that feed directly into Red Hat products. In fact, community is so important that it is the heart of the Red Hat mission statement: “to be the catalyst in communities of customers, contributors and partners creating better technology the open source way” (http://tinyurl.com/6rrve4n).

To nurture and fuel those communities, the company created a Community Architecture and Leadership team. Our most notable involvement is with The Fedora Project (http://fedoraproject.org), the results of which feed directly into Red Hat Enterprise Linux. The Fedora Project’s mission is wholly focused on the advancement of free and open source software around four foundations:

1. Freedom. The advancement of software and content freedom, not just in Linux, but overall

2. Friends. Including more than 24,000 Fedora Account System members

3. Features. Many features that benefit all Linux distributions start in Fedora.

4. First. The future of Linux is built into Fedora.

Fedora releases come out every six months, showing the edge of innovation and new features. Red Hat engineers participate in that process from the beginning. (However, 65–70% of Fedora’s code is maintained by volunteers.) Then, Red Hat dedicates its quality assurance resources to testing, hardening, and certifying those features to ensure that they meet the requirements for enterprise-level interoperability and performance. Code that started in the upstream community becomes the code that Red Hat customers, from DreamWorks to the NYSE Euronext, rely on to solve their daily business problems. A similar process happens in other communities that Red Hat participates in, including the JBoss Community (http://jboss.org), the OpenShift Community (https://openshift.redhat.com/app/), and the Gluster.org Community (http://gluster.org).

However, Red Hat's mission statement (which was developed collaboratively by all Red Hat employees) does not just say “contributors”. Red Hat also has customer and partner communities that are just as vital to its success. Red Hat customers have an unusually direct relationship with the company and influence what goes into releases and the direction of products, and they benefit from one another’s input, sometimes in surprising ways. What they are able to build together is greater than what any one company could build alone, and Red Hat can do a lot more when it works closely with them.

Q&A: What is the secret of Red Hat's Success?

Ruth Suehle

Community Beyond Code

Just as open source is no longer limited to code, Red Hat’s community contributions are not limited to code. As the leader in open source software and communities, Red Hat created opensource.com (http://opensource.com) in January 2010 to capture, highlight, and spread the open source culture. It tells stories of openness in six areas – Business, Education, Health, Government, Law, and Life – demonstrating the ways that openness is changing each of those things.

If you visit the website, you will notice that it is not immediately obvious that this is a Red Hat sponsored project. If you follow the posts at opensource.com, you will seldom see stories that mention Red Hat at all. The one place you will see Red Hat is on the right side of the header, where there is a small Red Hat logo and the words, “A Red Hat community service.”

Those five words were themselves a product of open debate and collaboration, going through several iterations of attempts to encapsulate the website’s mission and relationship to Red Hat. Opensource.com is not about promoting Red Hat. We do hope that it will help to show Red Hat as a leader and visionary in openness, but not by telling the stories of our own greatness. We want to be a leader by demonstrating how others are doing it right and helping those who have not yet embraced openness see the benefits.

The Key Principles: Transparency and Trust

Two key characteristics of the open source way are transparency and accountability. If you try to think of companies that truly embody those principles, you are likely to think of highly innovative, trustworthy companies you want to do business with. The tried-and-true business practices that fought against transparency are cracking. New methods of doing business are taking root. The basic tools that allowed big business to emerge – even the very management model beneath – are in desperate need of an update to support us in an increasingly complex world.

Simply put, the future of business is *open*. Many of the principles that have made open source an innovative software development model for Red Hat and others will stimulate innovation while creating an environment better suited to a 21st century world. In any part of any business anywhere.

Trust is one of the most important features of the open source way. It is the central catalyst of open source development. Many projects have failed merely for its absence. Collaboration works better when you trust the people with whom you are collaborating. Transparency is more believable when you trust those who are opening up to you. And a meritocracy can function only with a base level of trust in the community that everyone is competent and has the best interests of the project at heart.

Legal Issues

Legal questions are inevitable in a field that changes more quickly than laws can. Software often faces patent challenges, and fear of litigation was a barrier to adoption for many companies that otherwise would have had an earlier interest in open source software. To help with that fear, Red Hat created the Open Source Assurance Program (http://tinyurl.com/2jcthj) to protect customers by promising replacements of infringing software, as well as promising defense for an intellectual property lawsuit against a customer. The company also goes to bat itself on patent issues. In May 2010, a jury found in favor of Red Hat and Novell in a case on bad software patents owned by “non-practicing entities.” It was an important victory for those in open source. Rob Tiller, Red Hat vice president and assistant general counsel wrote of the case on opensource.com:

> *“We now know for certain that those in the business of bringing software patent lawsuits are not invincible, even in the supposedly patent-friendly jurisdiction of the Eastern District of Texas. We know that Texas juries are willing to reject bogus infringement claims and invalidate bad software patents. And we know that attacks on open source based on FUD* [fear, uncertainty, and doubt] *will not stand up when subjected to the light of truth.”* (http://tinyurl.com/2g9jumu)

Red Hat takes the position that software patents impede innovation and are inconsistent with free and open source software. Red Hat representatives have put their support to that position before the National Academies of Science, the U.S. Federal Trade Commission, and the U.S. Department of Justice, and the company is a signatory to a petition to the European Union encouraging it not to adopt a policy of permitting software patents. Nevertheless, software patents still exist, and Red Hat does maintain a portfolio of software patents for defensive purposes. That portfolio comes with a the Red Hat Patent Promise, created in 2001 through collaboration between developer Alan Cox and (then) chief counsel Mark Webbink to balance the need to protect open source with the need for defense. It ensures that Red Hat can help protect innovation by making sure that our patents are still available to the open source community in accordance with a set of approved licenses.

Conclusion

Open source is about more than the code, for Red Hat, and for the future of business, innovation, and culture. Innovation thrives on openness – and that applies to everything and everyone, anywhere. Red Hat just proves how it can also be profitable. The open source development model has transformed into a business and a cultural movement that Red Hat demonstrates, protects, and promotes through its business model and community participation.

About the Author

Ruth Suehle is a writer and editor in Brand Communications + Design at Red Hat. Previously an editor for Red Hat Magazine, Ruth helps to lead discussions about the open source way in the Life channel of opensource.com. She holds a BA in Journalism and Public Relations from the University of South Carolina-Columbia and has over 10 years' experience in content development roles, primarily in the technology sector.

Citation: Suehle, R. 2012. Q&A: What Is the Secret of Red Hat's Success? *Technology Innovation Management Review.* January 2012: 25-28.

Q&A

Carlo Daffara

Q. What are the long-term effects of open source?

A. It is no longer fashionable to talk about open source. We debate clouds, virtualized infrastructures, services, and BYOD (bring your own devices) as if those happened in a vacuum. To the contrary, I would like to present some data points that indicate how economics and physical constraints are turning open source into the hidden background on which a substantial number of IT trends are being based. In this way, we can see the long-term effects of open source.

Writing good software is an art more than a science, with several trends-du-jour that appear and disappear at breakneck speed. However, it still seem to be impossible to beat Fred Brook's law: "there is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement within a decade in productivity, in reliability, in simplicity … [and] we cannot expect ever to see two-fold gains every two years" (Brooks, 1986; http://wikipedia.org/wiki/No_Silver_Bullet). This proclamation dates from 1986, and we still see a yearly compound aggregate growth rate that is between 3 and 7% (European Commission Technology Working Group, 2004; http://tinyurl.com/728xu9w). In contrast, the code complexity in systems and devices that include software is growing at a much higher rate – a 58% compound annual growth rate – and the number of devices is growing as well, at around 10% yearly. This means that there is no way for software developers to be able to create all that source code in time, independently from the use of whatever tool or methodology.

This is, basically, one of the major reasons for the adoption of free/libre open source software F/LOSS, even by companies or groups that are fundamentally averse to the idea – they had no alternative. And the amount of code reuse is growing at a substantial rate as well. Recently, Black Duck (a provider of code auditing services) found that, in an analysis of large scale code bases with an average of 700MB of source code, 22% was open source and that 80% of new development is avoided through reuse of F/LOSS code. Thus, open source compensates for the increased complexity of software projects while at the same time containing costs; it also reduces the time to market and maintenance effort.

What is the indirect effect of such a massive introduction of F/LOSS code within multiple software projects? There are several, interconnected results:

1. The reused code improves faster than the rest of the code base, indirectly increasing the economic value of the F/LOSS projects. This effect is the basis for several research results that demonstrates that large, successful open source projects tend to have a very high code quality in terms of defect per line of code, on par or better than proprietary code. (See Mohagheghi et al., 2004; http://tinyurl.com/6o95opr for details and further observations on the effect of reuse on code quality and maintainability.) This in turn increases the probability that the code will be reused in the future, and reduces the cost of integration – a positive feedback loop for adoption.

2. Even with a small number of adopters contributing back patches and effort, the increased participation and the positive feedback creates an opportunity for superlinear growth in the affected project – an effect that is not hampered by increased complexity and communication costs, further increasing the value of the reused code.

3. The implicit support of open standards by open source code facilitates the adoption of open standards in the assembled code as well – thus "osmotically" promoting openness. In fact, this is one of the reasons for the extraordinary support of recent HTML5 engines and libraries and at the same time the maturation of the web as a delivery medium for applications at the expense of non-open additions and protocols.

It is possible to continue further in this little experiment: the "good enough" status of HTML5 as an application delivery platform allows users to reduce the reliance on locally installed apps, up to the point where all apps are delivered this way (eventually with a gateway bridge between legacy apps and the web, like VMware's project AppBlast; http://tinyurl.com/bnuvfyy). At this point, if all you need is a browser, execution platforms become interchangeable – you can use an ARM processor, a MIPS one, whatever. It means that cost-effective alternatives become feasible, such as the RaspberryPi platform (http://raspberrypi.org), which at $25 can be even embedded directly in a monitor at little added cost.

A further long-term effect will be an increase in structured collaboration across industries and companies participating in F/LOSS development – something that is now restricted mainly to a few horizontal platforms such as Eclipse or embedded Linux. As the economic advantage of F/LOSS becomes more visible, a larger number of participants will start to explore collaboration in vertical frameworks, such as industry specific toolkits or individual packages that may be relevant only to a few. Examples such as Albatross (http://albatross.aero), an air traffic control workstation, that now seem peculiar will become quite the norm, as more and more developers will go from pure integration of open source pieces to reduce development cost to a more structured collaborative participation. This shift will occur especially for companies that are not primarily IT producers, but users; this will further increase the shift from packaged software to reusable components, again reinforcing the movement towards F/LOSS.

Another effect will be changes in revenue per dollar spent: because software can grow faster with more or less the same spending level, the company will grow faster. A dollar invested in F/LOSS collaboration will bring back a real value that is substantially higher, thanks to the sharing of costs across collaborators.

F/LOSS is a game changer in more ways than simple reuse. Reuse at large scales changes the economics of IT in more profound ways, allowing better software, more software, and more affordability for everyone.

About the Author

Carlo Daffara is head of research at Conecta, an open source consulting company. He is the Italian member of the European Working Group on Libre Software, chairs several other working groups, including the Open Source Middleware Group of the IEEE Technical Committee on Scalable Computing and the Internet Society Working Group on Public Software, and contributed to the article presented by ISOC to UNESCO on global trends for universal access to information resources. His current research activity is centered on the sustainability of business models for open source software.

Citation: Daffara, C. 2012. Q&A: What Are the Long-Term Effects of Open Source? *Technology Innovation Management Review*. January 2012: 29-30.

Issue Sponsor

Author Guidelines

These guidelines should assist in the process of translating your expertise into a focused article that adds to the knowledge resources available through the *Technology Innovation Management Review*. Prior to writing an article, we recommend that you contact the Editor to discuss your article topic, the author guidelines, upcoming editorial themes, and the submission process: timreview.ca/contact

Topic

Start by asking yourself:

- Does my research or experience provide any new insights or perspectives?

- Do I often find myself having to explain this topic when I meet people as they are unaware of its relevance?

- Do I believe that I could have saved myself time, money, and frustration if someone had explained to me the issues surrounding this topic?

- Am I constantly correcting misconceptions regarding this topic?

- Am I considered to be an expert in this field? For example, do I present my research or experience at conferences?

If your answer is "yes" to any of these questions, your topic is likely of interest to readers of the TIM Review.

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

- Emphasize the practical application of your insights or research.

- Thoroughly examine the topic; don't leave the reader wishing for more.

- Know your central theme and stick to it.

- Demonstrate your depth of understanding for the topic, and that you have considered its benefits, possible outcomes, and applicability.

- Write in a formal, analytical style. Third-person voice is recommended; first-person voice may also be acceptable depending on the perspective of your article.

Format

1. Use an article template: .doc .odt

2. Indicate if your submission has been previously published elsewhere. This is to ensure that we don’t infringe upon another publisher's copyright policy.

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

4. 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.

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

6. 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.

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

8. Include a 75-150 word biography.

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

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

11. Include any figures at the appropriate locations in the article, but also send separate graphic files at maximum resolution available for each figure.