OSBR.CA

The Open Source Business Resource

JANUARY
2009

JANUARY 2009

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

EDITOR:
Dru Lavigne
dru@osbr.ca

ISSN:
1913-6102

ADVISORY BOARD:
Tony Bailetti
James Bowen
Kevin Goheen
Leslie Hawthorn
Chris Hobbs
Thomas Kunz
Steven Muegge
Donald Smith
Michael Weiss

© 2008 Talent First Network

In "The Role of Participation Architecture in Growing Sponsored Open Source Communities" (http://www.joelwest.org/Papers/WestOMahony2008-WP.pdf), Joel West and Siobhán O'Mahony argue that "to some extent, firms and technical communities have always collaborated to create standards, shared infrastructure, and innovation outcomes that are bigger than any one firm can achieve." and that "there is increasing evidence that path breaking innovations cannot occur without a community to interpret, support, extend and diffuse them". When considered in this light, it should not be surprising that more enterprises, both large and small, are increasing their participation in open source communities to drive innovation.

The theme for this month's issue of the OSBR is enterprise participation and the authors provide practical advice for effective enterprise/community collaboration. Their experiences provide perspectives on: i) the Eclipse Foundation, which maintains an ecosystem of over 150 enterprises that participate in Eclipse open source projects; ii) an independent software vendor that sells closed source solutions constructed on top of an open source platform to large enterprise customers; iii) the impact of major players collaborating on a common open source platform for the mobile industry; iv) the role users can play in the very large (over 14 million) GNOME community; and v) the lessons a scientist from the National Research Council of Canada learned when he released software and started a small open source community.

As always, we encourage readers to share articles of interest with their colleagues, and to provide their comments either online or directly to the authors. We hope you enjoy this issue of the OSBR.

The editorial theme for the upcoming February issue of the OSBR is commercialisation and the guest editor will be Robert Withrow from Nortel.

Dru Lavigne

Editor-in-Chief

dru@osbr.ca

Dru Lavigne is a technical writer and IT consultant who has been active with open source communities since the mid-1990s. She writes regularly for O'Reilly andDNSStuff.com and is the author of the books BSD Hacks and The Best of FreeBSD Basics.

An Independent Software Vendor (ISV) would be foolhardy if it did not consider open source as part of an overall business strategy. Even if not using or participating in open source projects, ISV's using a purely commercial license approach still need to keep a keen eye on their market and be aware of open alternatives. The same is becoming increasingly true for non-ISV's, those organizations who make strategic use of software but are not ultimately known by their customers as software companies. Aerospace, automotive, retail, mobile and embedded devices, shipping and logistics, banking and healthcare are some examples of industry verticals where organizations are greatly increasing their participation in open source. This growing wave of enterprise participation in open source is promising to radically change the relationship between consumers and suppliers of software. This issue of the OSBR explores these changes and will help software vendors and consumers alike prepare to take advantage of emerging opportunities.

Ian Skerrett from the Eclipse Foundation outlines a model that has proven successful at establishing innovation networks. These networks encompass all roles within a software supply chain -- be they between suppliers, consumers or some combination thereof. Skerrett notes that an independent governance model and rules for sharing intellectual property are keys to this model's success.

Kingston Duffie, CTO of the Fanfare Group, provides an outline of how a small ISV can work with large enterprise consumers in an open collaborative environment to the benefit of all. Duffie presents the importance of thinking about the platform and how an open collaboration can lead to better initial requirements.

Stephen Walli, a consultant on open source, standards and software, provides a view of the impact open source has on the mobile Internet and offers models for looking at the future. Walli describes the various key players in the mobile space and how they've reacted to, and participated in, various open initiatives such as Mobile Linux and Android. He also overviews Symbian's recent move to become an independant foundation and its adoption of the Eclipse Public License.

Stormy Peters, Executive Director of the GNOME Foundation, discusses the importance of consumers within an open source project. Peters notes the importance of enterprises in testing, marketing and financial support and the various ways the GNOME foundation encourages consumers to become more active.

Alain Désilets, a Research Officer at the National Research Council of Canada, provides the perspective of an enterprise looking to release an asset as open source, and what to consider ahead of time. Désilets notes the importance of lowering the barriers to entry and collaboration in order to build a strong community.

A common theme amongst all of these articles is the need for open collaboration and a clear governance model. With these two crtieria in place, a level playing field can be established where consumers and suppliers alike can continue to drive innovation in software.

Donald Smith is Director of Ecosystem Development for the Eclipse Foundation, an independent not-for-profit foundation supporting the Eclipse open source community. He brings over a decade of worldwide industry experience, from small "dot-com" through Fortune 50 companies. Donald speaks regularly at both technical and business oriented conferences.

"Innovation distinguishes between a leader and a follower." Steve Jobs

Open source has been many things to many people. In most cases, open source software (OSS) has focused on the tools and the infrastructure software used to build and deploy applications. Relative to infrastructure, little use or investment has occurred in the development of industry specific or vertical oriented open source solutions. This is not only a missed opportunity for organizations, but it is also possibly the next wave of open source collaborations.

An important lesson of OSS is a development process that requires collaboration between individuals and organizations that isn't necessarily driven by a traditional hierarchy of command and control. Information technology (IT) departments are driven to be more efficient while simultaneously creating innovative new solutions to meet their business needs. More and more, companies are turning to external sources for ideas that drive innovation. A series of books by Henry Chesbrough (http://en.wikipedia.org/wiki/Henry_Chesbrough) has coined the term "innovation networks" to discuss research and development (R&D) departments that treat their R&D systems as an open system. They describe how to include partners, customers and even competitors as part of an extended R&D team.

However, that series does not answer the question "how does OSS contribute to innovation networks for IT departments?". Further, what steps are required to establish a successful software innovation network and what are the resulting benefits for organizations? For the most part, this article will draw upon the experiences of Eclipse open source projects. These projects often include competing independent software vendors (ISVs) that collaborate on building a common platform for developer tools integration.

The Drive for Collaboration

Determining the scope of collaboration is often the most challenging aspect of starting an open source project. The key challenge is to understand which areas of technology are core and which are non-core to the business value of that organization. Based on previous experience in the software industry, we believe that OSS tends to lead to two logical strategies for collaboration:

- collaborating on the implementation of industry standards or protocols
- establishing an industry platform to grow a market

Collaborating on Open Standards

Globalization and government regulation have increased the importance of industry standards and protocols. There are many examples of consortiums that define standards and protocols for specific technologies or specific industries. However, the implementation of these standards is often left to ISVs or individual IT organizations.

For technology standards such as HTTP, XML, and Java, software vendors are expected to implement these standards in their products. However, there is little incentive to do so as the implementations provide very little differentiating features and customer value add. For this reason, OSS provides an effective mechanism for creating a common implementation that drives the adoption of these standards. The Apache web server is a good example of an open source application which drove the HTTP standard.

A similar case can be made for IT organizations that need to implement specific industry standards and protocols.

The actual implementation of these standards provides very little benefit to the core business of an organization. IT organizations typically rely upon ISVs or internal development groups to implement these standards and thus incur the costs of sourcing the implementation.

The drive for collaboration is propelled by the need for IT organizations to quickly and efficiently implement new regulations or standards for their business. Organizations within the same industry can join together as a software innovation network to create a shared implementation of a standard. A common implementation means that the development cost is shared and provides the further benefit that the common deployments result in greater interoperability.

Collaborating on a Common Platform

Creating a common industry platform can address the IT challenge of integrating solutions from different vendors and help accelerate the growth of a fragmented market.

A consistent requirement of IT organizations is the need to integrate solutions from different vendors. For instance, customer relationship management (CRM) systems often need to be integrated with e-mail systems; financial institutions need to integrate data feeds from many providers; or large scale manufacturers, such as automotive or aerospace, have extensive supply chains that need to integrate across the product lifecycle. Typically, the integration is a cost of doing business, not a core value, so creating a common platform that is adopted by a number of industry players effectively streamlines the integration requirements.

Establishing a common platform in a fragmented market of providers can help grow the entire industry. In fragmented markets, significant investment is often duplicated across solution providers but provides no real customer value. In addition, a valuable market ecosystem cannot develop because the market share of each provider is not big enough to sustain investment on one particular platform. Therefore, if multiple players agree to collaborate on a common platform, it can reduce the barriers for increasing the size of the overall market.

Establishing a Software Innovation Network

OSS development provides a proven model for creating shared implementations. However, the ultimate goal of a software innovation network is to create business value. We need to consider several aspects of OSS that allow for value creation and value capture when establishing a formal collaboration amongst equal partners. We discuss the factors required to create this type of collaboration environment, something we call a "Software Innovation Network".

1. **Open Development Process:** the success of OSS in facilitating collaborative development is in an open development process. Most major open source communities, like Apache, Eclipse and Linux, operate using the following three principles:

a) **Openness:** being open to participation by any individual or organization, including competing organizations.

b) **Meritocracy:** openness does not mean democracy and successful open source projects instead work on the principle of meritocracy (http://en.wikipedia.org/wiki/Meritocracy). In this model, newcomers are invited to participate based on their proven merit and ability.

c) **Transparency:** having important project discussions, plans, and meeting minutes available in a transparent manner so anyone can view them.

2. **Enabling a Governance Model for Collaboration:** all successful long-term organizations require a set of rules that establish a governance model for setting policies and strategies. Governance becomes even more important if the organization is a collaboration amongst competitors. It is, therefore, critical that the governance model does not allow a single player to control or influence the organization. The perception or reality that a single participant controls the overall community can inhibit the participation of others. Organizations like the Eclipse Foundation, Apache Foundation, and Linux Foundation have been established as not-for-profit organizations with the specific purpose of being vendor-neutral entities that act as the steward of their respective open source communities.

3. **Intellectual Property Management:** intellectual property (IP) management is a critical consideration when creating a shared technology base. Effective IP management includes the selection of an appropriate software license, legal agreements for participants that cover the contribution of IP, and scanning of source code to ensure pedigree and license compatibility.

For instance, the Eclipse Foundation has a well established IP management system. All participants in the Eclipse community sign the same agreement and follow the same IP processes. All Eclipse open source project committers sign a committer agreement that specifies that their contributions are licensed under the Eclipse Public License (EPL, http://opensource.org/licenses/eclipse-1.0.php).

All source code that is contributed to any Eclipse project is automatically scanned to ensure all of the code is licensed under the EPL or a compatible open source license. The result is that the technology created within the open source projects has clear software license and IP pedigree.

4. **Creating a Community:** Tim O'Reilly coined the term "architecture of participation" to describe how open source projects are able to build and engage a community (http://www.oreillynet.com/pub/a/oreilly/tim/articles/architecture_of_participation.html). The idea is that an open source community forms around the ability of any individual, regardless of their affiliation, to participate. An architecture of participation is enabled by: i) making it easy to extend the technology; and ii) having an open development process that is transparent to all. Participation then occurs by those individuals contributing directly back to the project or building new technology on-top of the base technology. The end result is an ecosystem that adds the needed components for quick adoption of new technology.

The network effect of smaller communities within larger communities has also proven beneficial for starting new projects. A significant challenge for any new community is generating awareness and participation. Well established organizations like Apache and Eclipse allow new projects to leverage the larger community to raise their profile with potential community members.

5. **Establishing the IT Infrastructure:** the IT infrastructure required to host a community software innovation network is non-trivial. Typically, open source collaborations require a web site, source code repository, bug tracking database, wikis, mailing lists and newsgroups. Consideration needs to be given to the ongoing administration and management of the associated infrastructure.

6. **Open Business Models:** a goal of a Software Innovation Network is to create an ecosystem of organizations, commercial and not-for-profit, that benefit from a common platform. These organizations will employ a variety of business models and strategies. Therefore, it is important to ensure that the choice of license and governance model allows for maximum flexibility.

Conclusion

Most IT organizations have reduced software licensing costs by being users of OSS. The next step to additional IT efficiencies will be their participation in OSS projects. Open source communities have demonstrated a model for collaborative software development that can be the basis for Software Innovation Networks. Visionary IT departments have already begun to leverage this model to collaborate on the development of technology specific to their domain. Over the next few years, open Software Innovation Networks could very well become the future of software development.

This article has been updated and refocused by the author from its original publication on CIO.com (http://www.cio.com/article/367213/Using_Open_Source_Innovation_Networks_to_Drive_Collaborative_Software_Development?page=1).

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

Recommended Resources

Building Technical Communities
http://www.osbr.ca/ojs/index.php/osbr/article/view/612

Innovation through Global Collaboration: A New Source of Competitive Advantage
http://hbswk.hbs.edu/item/5760.html

ISVS, ENTERPRISE CONSUMERS, AND OPEN SOURCE

"Together they would travel on a boat with billowed sail,
Jackie kept a lookout perched on puffs gigantic tail,
Noble kings and princes would bow whene'er they came,
Pirate ships would lower their flag when puff roared out his name."
Puff the Magic Dragon

Selling software to consumers is tough as they want perfection and expect to pay nothing. Consumers are fickle and the competition can be fierce. By comparison, large enterprises have big problems and buckets of money to spend on solutions, allowing you to focus your marketing on a small target. While large enterprises sound like ideal customers, the small independent software vendor (ISV, http://en.wikipedia.org/wiki/Independent_software_vendor) selling to a large enterprise faces long sales cycles, extreme quality expectations, challenges integrating into a complex jungle of preexisting systems, and the need to partner with others to create a complete solution.

At Fanfare (http://fanfaresoftware.com), we believe that there are a variety of successful software business models that employ open source strategies. One such strategy is to sell closed source solutions constructed on top of an open source platform. In fact, the Eclipse Foundation (http://www.eclipse.org) considers enabling this kind of business model as one of its mandates. We sell a complete system test automation solution that is built on top of Eclipse and sold to customers as a "turnkey" solution using Eclipse's Rich Client Platform (RCP, http://wiki.eclipse.org/index.php/Rich_Client_Platform) technology.

In this article, we examine six benefits of selling closed source solutions built on top of an open source platform. Some of these benefits apply to any ISV, but many apply to ISVs selling to large enterprises.

Benefit 1: Platforms Instead of SDKs

The days of selling a completely standalone software product are long gone. Enterprise customers need software customized to integrate into their systems. However, customizing software is a distraction for the ISV. Customers demand integration between products from different vendors. Yet, bilateral integration between vendors is time-consuming. Worse, when a small vendor requires integration with a larger vendor, it may be difficult to attract the attention of the larger vendor.

In short, the integration answer is no longer the software development kit (SDK, http://en.wikipedia.org/wiki/Sdk). A small vendor can create an SDK, but it is going to be difficult to get others to use it. Enterprise customers seeking customization are likely to just ask the vendor to do the work themselves.

Today, the answer is the platform itself. With the right extensibility model, a small ISV can succeed by translating its domain expertise into an open source set of domain-specific APIs (http://en.wikipedia.org/wiki/Api) and extension points that are available to anyone to use and build on. For example, we recently proposed a new Test Automation Platform (TAP, http://www.eclipse.org/proposals/tap/) project to the Eclipse Foundation. What started as an SDK for Fanfare's iTest product may become a new set of APIs that Fanfare's partners, customers and competitors can use. Because it will be part of Eclipse, the API will have a much better chance of forming a vibrant community that extends beyond Fanfare. It also means that Fanfare's enterprise customers can develop internal proprietary components on top of this platform with some comfort that they are not specific only to Fanfare's products.

Several of the companies that have expressed interest in this project are the large enterprise customers themselves such as British Telecom, Cisco, and HP ProCurve.

How does an SDK differ from an API? The significant difference is that an SDK is specific to one vendor and is "one-sided", meaning that the vendor is exposing an interface that others can use to implement some extension that will only work on top of that one vendor's solution. By defining an API in an open platform, it becomes a "two-sided" interface, meaning that any implementation on top of the API will work with any implementation providing that API. The implementation might be provided as open source within something like Eclipse, or it could be implemented through a services extensibility model, allowing multiple vendors to contribute competing solutions.

Why would an ISV contribute an API and potentially expose itself to alternative solutions from their competitors? In short, small ISVs thrive when communities develop. Trying to build a community around a closed SDK is very difficult. But by opening the APIs to potential competition, they also create a climate in which a community is more likely to flourish. Moreover, the ISV's customers are more likely to encourage other suppliers to support these APIs because they are not vendor-specific.

Benefit 2: Collaboration

Arguably, one of the most difficult challenges for ISVs selling into large enterprises is in defining the requirements. It is not uncommon to deliver a product to a large customer only to hear about the features that they need that were not included.

Even with a thorough product management strategy, minor miscommunications can result in major delays as these missing requirements are addressed. And that usually means a delay in revenue.

Does open source change this? Not always. But many large enterprises have their own internal development teams which are tasked with integrating your solution with their proprietary systems, and this is often the source of the most complex requirements. Rather than waiting until you are ready to deliver your final, tested software to your customers, open source allows collaboration with your customers and partners. There is much less room for ambiguity when you are exchanging requirements using actual software. Even better, your customer can be developing their integrations at the same time that you are developing your implementations. By the time you are ready to deliver, they are ready to deploy.

This open source benefit can expand far beyond a particular deployment. Fanfare is working with one of its biggest customers on a project based on the same code that we expect to contribute to the upcoming TAP project described earlier. They can comment on the interfaces they need while they build the integrations into their existing proprietary tools. We gain confidence that what we are building is actually going to be used. Partners can see that this has momentum and will build complementary components that interoperate at the same interfaces. Win-win-win.

Benefit 3: Thinking Ahead

Small ISVs have a speed advantage when compared with large software suppliers. An ISV can compete by innovating faster and bringing new ideas to market sooner than their larger competitors.

What often happens, though, is that this can quickly produce an intractable code base. Success in the early years of an ISV can be followed by a delay from the necessary regrouping as more sustainable development processes are put into place. While few ISVs will admit that they have this problem, we all do and it can happen at the worst time when you are selling to enterprise customers. You finally have adoption starting to happen after a long sales cycle. Now you're being asked for a whole set of new features. This is exactly the time when you don't want to experience a long delay as you re-architect your software.

How does open source help to address this problem? First, the more transparent the software, the better its quality. If software is going to be scrutinized by your peers, you are going to do more work to ensure that it is not only bug-free, but that it is also well-organized and highly extensible.

Second, one can build on a mature and successful model of modularization. By following Eclipse's natural extensibility model, our software designers are more inclined to build reusable components rather than a big monolithic codebase – even for the closed-source components in the final solution.

Third, one is faced with the decision "will we open-source this?" at the beginning of every project. If not, will our big customers or partners expect to be able to extend this functionality? If so, we need to define a part that is open and extensible.

Benefit 4: Cross-Platform

Fanfare's first generation product was built on top of Microsoft's .NET framework (http://en.wikipedia.org/wiki/.NET_Framework).

At the time, the Microsoft development tools seemed like a good solution for getting a rich client application to market quickly. Customers quickly indicated that broad adoption would require product availability on all leading operating systems. Instead of moving to a closed-source Java solution, we recognized that being truly cross-platform means a lot more than just Java. Again, attempting a solution that works on multiple operating systems without an open source platform would be folly as the vision of "write once, run anywhere" is a pipe dream. Supporting a product on multiple platforms is a lot of extra work, and an open source platform like Eclipse is a huge advantage in this regard.

Benefit 5: More Credibility with Customers

We sell our products to some of the largest enterprise customers in the world who use these products for mission-critical purposes. Many of these customers are not active users of Eclipse. At the same time, virtually every customer is delighted to hear that Fanfare's solution is built on top of a solid open source foundation. This gives them confidence that the platform is solid and that they will be able to take advantage of other innovation coming from the Eclipse community.

Benefit 6: Less Code to Develop

Regardless of its target market, the last thing that an ISV can afford to do is to reinvent the wheel. Almost every software product requires a platform that provides modularity and common user interface components. One doesn't want to spend one's valuable development resources on a platform.

Buying one might be possible. But it is nearly essential to have access to the source code for the platform because, as a commercial vendor, your corporate customers will expect you to troubleshoot and resolve all problems in your solution – including those in the platform. If you don't have the source, you are in a tough position.

One way in which this benefit is specific to ISVs selling to large enterprises is because of "solution selling". Enterprises typically don't want to assemble components themselves. They prefer to have a vendor provide them with a turnkey solution. For example, our customers use a variety of version control systems. Because our product is built on Eclipse, it can bundle a variety of source control client components that are available for Eclipse. Moreover, our customers can seek out their own preferred components and integrate these themselves using Eclipse's software management capabilities.

Conclusion

Selling software to large enterprise customers is difficult. These customers require high complexity, demand superlative quality, expect infinite extensibility, and don't want to be held hostage by a single vendor. At the same time, these customers are willing to pay for good solutions and will invest their own technical resources to ensure a great product that they can integrate into their workflows.

We believe that an open source platform strategy is the best way for ISVs to attach to these markets. Development can be focused solely on the pieces that are unique to the ISV's expertise. Enterprise customers now appreciate products built on open source platforms like Eclipse, especially when they can extend that platform themselves when needed. ISVs benefit from an open source development model, even for the portion of their software that remains closed source. At the same time, it takes more time to deliver open source software and ISVs need to be prepared for that.

Open source is a two-edged sword for small software vendors trying to sell into large enterprises. But that might be just the weapon Jack and Puff need to ward off those pirate ships.

Kingston Duffie is the founder and CTO of The Fanfare Group, a market leader in system test automation tools. Previously, Kingston founded two other successful data communication equipment companies, Whitetree Networks and Turnstone Systems. Kingston holds a bachelor's degree in engineering physics from Queen's University in Kingston, Ontario, and a master's of science degree in electrical engineering from McGill University in Montreal, Quebec.

"Sometime around the middle of this decade -- no one is sure exactly when -- executives on the go will begin carrying pocket-sized digital communicating devices. And although nobody is exactly sure what features these personal information gizmos will have, what they will cost, what they will look like or what they will be called, hundreds of computer industry officials and investors at the Mobile '92 conference here last week agreed that the devices could become the foundation of the next great fortunes to be made in the personal computer business."

Peter Lewis
http://tinyurl.com/8xbm25

The promise of the mobile Internet has been long in coming. In 1992, then Apple CEO John Sculley was promising this "pocket-sized digital communicating devices" market would be "the mother of all markets", while Intel CEO Andrew Grove called it "a pipe dream driven by greed." Since then, the mobile phone business has exploded, and personal digital assistants (PDAs) like the Palm Pilot (1997) have burst onto the scene. The launch of the RIM Blackberry (1999) brought a real email interface to the PDA world (2002). The World Wide Web itself continues to grow enormously, with Netcraft's December 2008 survey (http://news.netcraft.com/archives/2008/12/24/december_2008_web_server_survey.html) receiving responses from 186,727,854 websites.

We are just now arriving at a convergence in the market that is 16 years in the making. Handset and PDA manufacturers, mobile network operators, chip manufacturers, and computer platform hardware and software vendors all collide with the economics of the Web, collaborative development, and open source software (OSS).

Indeed, we are seeing a point in history in which the mobile handset manufacturers and their partners are using OSS and collaborative development to ensure they do not get trapped in the narrow margin price war that caught the personal computer (PC) original equipment manufacturers (OEMs) in the previous technology wave.

A Little History

The historic business of the mobile network operators is based on phone calling time (minutes) and data transfer (bytes). Browsing the web has remained uninteresting for some time, due to low bandwidth and poor interfaces. A few square centimetres makes a poor window onto the Web, especially as web sites have become more content rich and application complex over time.

A significant revenue stream also developed in the mobile communications space around text messaging. The global short message service (SMS) revenue for 2007 was US$100 Billion. This is greater than the combined revenues of the movie box office, DVD sales and rentals, the music industry, and video games (http://www.slideshare.net/grigs/going-fast-on-the-mobile-web). The SMS revenue stream also has an amazing margin, as most of the traffic happens with small limited messages in the control channel reserved for network management (http://www.nytimes.com/2008/12/28/business/28digi.html?th&emc=th). The phone has become a texting device for a whole new generation of users.

In July 2007, the iPhone launched with the wifi-enabled iPod Touch coming shortly on its heels. Within a short few months, these devices dominated traffic on the mobile Internet, claiming 65% of the traffic by January 2008 (http://www.netapplications.com).

The communications device world finally had the innovation example it needed for a mobile Internet. Bandwidth was also catching up as 3G (http://en.wikipedia.org/wiki/3g) networks came on stream. In the past 18 months, all the main device manufacturers have been rushing products to market that present significantly better viewing surface area for media and the Web.

Apple's innovation example went beyond the device. The Apple iTunes Store enables a network ready service from either Wifi or the mobile carrier network. Customers demonstrated they were more than happy to pay for the convenience and immediacy of instant music downloads. This was rapidly extended to the App[lications] Store, also hosted through iTunes. A class of iPhone/iTouch mobile applications appeared on the App Store as Apple encouraged a developer community to form around their mobile platform.

Nokia entered the services market very shortly thereafter with their US$8.1 billion acquisition of digital map maker Navteq in October 2007 (http://www.informationweek.com/blog/main/archives/2007/10/nokias_acquisit.html). Navteq maps are used by almost all major Internet mapping tools, including MSN, Yahoo, and Google. Garmin GPS devices are a primary customer of Navteq. As the world's largest handset maker, Nokia appears to be staking enormous bets that location dependent services will be a booming business in the mobile device world.

Google has long dominated the Internet advertising business. But with approximately 3.3 billion mobile subscriptions in the world versus 900 million personal computers (http://communities-dominate.blogs.com/brands/2008/01/when-there-is-a.html), one can see that Google would want to extend its reach into the mobile Internet space quickly. In November 2007, Google led the formation of the Open Handset Alliance (http://www.openhandsetalliance.com/) to collaborate on developing a Linux-based royalty-free development platform (operating system and middleware) for mobile handsets.

As Google was bringing Android to market, Nokia announced in June 2008 its intent to acquire Symbian Ltd., the makers of the Symbian handset operating system, for US$410 million and its intentions to form the Symbian Foundation (http://www.symbianfoundation.org/) in early 2009, making the operating system available royalty-free and as open source through the foundation through 2010.

Christensen Economics and Innovation

The trend in mobile operating systems to open source licensing and collaborative development can best be described in terms of Clayton Christensen's (http://en.wikipedia.org/wiki/Clayton_M._Christensen) observations about economics and innovation. Christensen's model provides a great lens through which to view the disparate mobile efforts from the iPhone to the various Linux-centric efforts, and why they differ so widely. Three ideas captured in the model are relevant to our discussion:

1. In an early market, the product that "wins" is tightly integrated because the manufacturer has complete control of all aspects of the device and can deliver best on the customer experience. Over time, as the technology space matures, new players can enter the market at the bottom with "good enough" products that satisfy the low end of the market.

2. As a technology market matures, standards emerge, enabling more players to enter a market where each attempts to differentiate their offerings.

3. Products live in a network of connected technologies, services, and experience. As standards cause value to fall in one of the "nodes" in the network, the value moves to adjacent complementary technology spaces in the network. For example, as inexpensive standards-conforming PCs eroded the value of minicomputers, the profits moved into operating system revenues.

The iPhone and the Blackberry

We will now discuss how open source impacts on the various players within the mobile market. While the iPhone runs an operating system that is related to the Apple-led Darwin OSS project, Darwin has never been a major project and isn't a primary engagement mechanism for Apple with its developer community. The iPhone, true to Apple's history of profitability over market share and its cult of design, is first and foremost an amazing consumer experience. The complete Apple experience depends upon controlling the entire technology stack (hardware, software, application development layer, and service provision) in a tightly integrated fashion.

In the Christensen economic model, the iPhone is a new class of product. It will deliver more value to its target customer through tight control and integration than can be delivered through standardized interfaces and components, and Apple will reap the margin benefits. That focus on integrated design means that unlike Symbian, Apple will never own 65% of the global market. Apple won't be interested in producing "cheap" iPhones for developing economies as Apple is about "life style" computing.

The iPhone has demonstrated to the mobile industry what the mobile web experience can be from both a device and a services perspective.

What works so well from a tightly integrated design perspective turns out less well when we look at other single-source devices like RIM's Blackberry. Initially, RIM had their own new class of device. What started as two-way paging in 1999, rapidly became the niche push email service that was key to business users. The Blackberry Enterprise Server connects to the corporate email system, pushing email in real-time down to Blackberry devices. By tightly controlling the entire stack from the hardware and operating system through to the email service provision, RIM was able to present the best business consumer email experience, as well as the associated profit margins.

However, technology has converged, the iPhone has licensed the Microsoft Exchange ActiveSync technology for the push email used with Microsoft Exchange servers, and RIM is left trying to evolve the Blackberry against a device with two years of advanced features and services well beyond its own push email niche. While Apple continues to expand the connections within its technology network to best effect, RIM is left somewhat in a one-trick pony position.

Linux Variations on a Theme

Symbian and Microsoft were able to deliver powerful smartphone operating systems that ran on a wide variety of handsets across all the mobile networks within their regulatory environments. They were rewarded through their per handset royalty. Handset companies, however, are exploring Linux as a mobile operating system platform.

Symbian and Microsoft looked to China, India, and other developing economies and assumed a proportionate claim of market share as those markets evolved. The problem from the handset manufacturer perspective changes when you start considering billions of dollars in royalty payments from those potential markets. At that level of cost, Linux experimentation and investment makes good business sense.

Motorola delivered the Ming phone into China using a Linux base. This was almost a year before the iPhone was announced and, for the Chinese market, was arguably a more useful phone. The Ming supported full stylus input for Chinese characters, an enormous consideration for texting. The Ming also had the first 2 megapixel camera, a media player, and came in a sleek package.

At the same time, Nokia was delivering the N770 and N800 Series Internet tablets, devices not much bigger than the first iPhones, with stylus input and a viewing area similar to the iPhone. These were Linux-based and Nokia was working directly in the open source community on several fronts to maintain concurrency with the Debian Linux distribution rather than living on a fork (http://static.maemo.org/static/a/a54f861c268711dc931aad2aaa49dce9dce9_developing_770_with_linux_and_oss.pdf). From these efforts, Nokia developed the Maemo open source developer community (http://maemo.org/).

This was a significant step. As the handset industry changed and the operating system royalty became a problem, each handset manufacturer wrestled with Linux. Each wanted a royalty free operating system, but trying to integrate into the Linux community has been a source of frustration.

Things that are critically important to handset manufacturers, such as battery life, aren't necessarily interesting to the mainstream Linux community. Each handset manufacturer was forced to fork with the attendant costs. Nokia, working within an existing community, was a huge step forward.

Out of the desire to avoid forking Linux came LiMo (http://limofoundation.org/). LiMo is a not-for-profit organization formed by the key mobile players in January 2007 to collaborate on a Linux kernel for mobile. This level of collaboration is not foreign as the mobile handset manufacturers shared technology through Symbian Inc. for ten years. Symbian Ltd. was a for-profit corporation, but its primary shareholders and board members were the handset manufacturers. The handset manufacturers do not want to fall into the trap that the PC manufacturers fell into with Microsoft, where the manufacturers took the biggest hit in the narrow margin price war, while Microsoft made money on the operating system. So, for the handset manufacturers to work collectively to manage mobile operating system costs while continuing to deliver their own differentiation on the handset makes good economic sense.

Before Google announced Android and before Nokia announced that Symbian would become open source, LiMo was likely the best opportunity for a shared royalty-free Linux mobile platform. LiMo's downfall will probably be Nokia's proven ability to work directly with the open source community. With the formation of the Symbian Foundation and the Open Handset Alliance, it remains to be seen if LiMo will deliver the Linux distribution for mobile, or whether the handset manufacturers will choose to instead focus investment in the other two efforts.

There are other Linux-based efforts underway in the mobile Internet space. Intel created Moblin (http://moblin.org/) to work on a similar mobile focused Linux distribution for the growing "netbook" and in-vehicle devices. Canonical Ltd., the primary developer of the Ubuntu Linux distribution, is working on the Ubuntu Mobile Internet Device Edition (MID, http://ubuntu.com/products/mobile) which is focused on the growing netbook market. Each of these are carefully not targeting the mobile phone space. It will be interesting to see how this space evolves as the mobile phone and laptop/netbook spaces converge.

Android

Google led the formation of the Open Handset Alliance to collaborate on developing a Linux-based royalty-free development platform (operating system and middleware) for mobile handsets. This culminated in the release of the Android mobile software platform as OSS in October 2008, and was shortly followed by the Android Market as a distribution mechanism for Android-based applications. Like Apple, Google is forming a developer community around Android and wants to set up the Android Market as the marketplace for developers and consumers to meet.

Google wants to drive Android application development that uses Google services in order to grow their ad revenue as the mobile web comes into its own. They are providing the necessary focus around which the handset manufacturers, mobile network operators, and related OEMs can collaborate. Based on their own open source experiences, they are also providing the necessary architecture for participation required for a successful open source community to develop.

Android is likely to face two key challenges. First, Google is not an operating system company. They don't necessarily have the experience and resources to meet the calendar demands the handset manufacturers have for regular releases, something that Symbian has traditionally done well. Second, while it makes sense from Google's perspective to drive mobile application development around their services, the rest of the world may not feel that way and have their own, possibly competing, perspectives.

This should be an opportunity for the newly forming Symbian Foundation. Google application services running on a Symbian base would seem to be a win for Google, application developers trying to settle on a model, and for Symbian doing what it does best around the mobile operating system. Since the not-for-profit Symbian Foundation will not be under market competitive revenue pressure, profit-centric competitive decisions are removed that might have historically put cooperating with Google at risk. This would seem to be a good time for Symbian and Google to explore their complementary spaces in the technology network.

The Symbian Foundation

In December 2008, regulatory approval was given and Nokia completed the acquisition of Symbian Ltd.. Nokia will launch the Symbian Foundation early in 2009. Fujitsu, Motorola, Nokia, NTT DOCOMO, and Sony Ericsson will contribute related software and documentation assets to the newly formed foundation. All the assets will be available to foundation members under a royalty-free license.

A new platform will be developed from SymbianOS and other donated components.

The platform will offer the means to build a complete mobile device while providing the tools to differentiate devices through tailoring of the user experience, applications and services. Platform assets will be made available as open source gradually over the next 2 years. The intent is to use the Eclipse Public License (EPL, http://opensource.org/licenses/eclipse-1.0.php), making the platform code available to all for free.

The Symbian Foundation's use of the EPL will likely make handset manufacturers more comfortable than the GPLv2 under which all Linux work must be licensed. The IBM-lineage of the license ensures that a hardware patent clause is still intact to protect unambiguously all hardware patents.

There will certainly be challenges. Nokia will need to manage the challenges that come with any acquisition. Organization and governance of the new foundation will be key, but the Eclipse and Mozilla Foundations provide excellent blueprints for the Symbian Foundation. Quickly putting into place an architecture of participation for everyone to understand how to contribute and participate will be essential for the Symbian Foundation to succeed.

Windows Mobile

Microsoft faces interesting challenges for the Windows Mobile platform. First, there's the royalty problem. With collaborative development, open source intellectual property (IP) management, and a lack of royalties defining the investments of the industry, it is difficult for Microsoft to justify the value of their royalty-laden offerings. Windows Mobile only had a quarter of the deployment of SymbianOS in a pre-acquisition, pre-foundation world.

Now SymbianOS will be royalty free, as is Android, anything delivered from LiMo, and the Intel and Ubuntu work. Offering a royalty free operating system, however, only makes sense for Microsoft if it drives significant revenue growth and profitability in related products and services in the adjacent nodes of their solution network.

Second, there's the open source culture versus IP protection problem, both internally and from an external partner perspective. It is unlikely that Microsoft will risk participating in any of the existing open source foundation or alliance efforts.

There is also a more subtle business problem. The handset manufacturers do not want to be caught in the same trap as the PC OEMs. Mobile industry partners have banded together through various foundations and alliances to force the economic devaluation of the operating system. This will further isolate Microsoft from the industry.

Summary

The mobile Internet has been a long time coming. Even as mobile phones and PDAs became prominent and the World Wide Web itself exploded, poor bandwidth and awkward viewing interfaces hampered the mobile Internet from taking off. We are finally at a point of industry convergence where these problems are being solved rapidly.

Christensen's economic model for innovation allows us to see why the different device manufacturers behave as they do. OSS and collaborative development are redefining how the mobile Internet will be built. The various handset manufacturers and their mobile industry partners are sharing the costs of mobile operating system development through multiple different alliances.

This allows them to drive their own differentiation on the handsets themselves to maximize margins.

At the same time, the Apple iPhone continues to provide a great innovation example for all to follow. It remains to be seen how RIM and Microsoft will respond, if the situation allows them to respond at all.

Stephen R. Walli has been in the software industry since 1980 as both customer and vendor. He presently consults on open source, standards, and software business. His clients include Symbian, Microsoft, and the Eclipse and Linux Foundations. In 1995, he was a founder and vice-president, R&D at Softway Systems, a venture-backed startup that developed Interix to migrate UNIX applications to Windows NT based on the POSIX/UNIX standards he helped develop. Interix was Softway developed code, Microsoft licensed code, and a wealth of OSS covered by many licenses. Microsoft acquired Softway in 1999, where Stephen spent five years before joining another open source based start-up, Optaros, as vice-president, open source development strategy. He left Optaros in 2006 to pursue his own interests. Stephen organized the first Beijing Open Source Software Forum as part of the Software Innovation Summit 2007, and remains interested in OSS growth in China. He blogs at http://stephesblog.blogs.com.

Recommended Resources

The Innovator's Dilemma
http://www.businessweek.com/chapter/christensen.htm

The Innovator's Solution
http://www.theinnovatorssolution.com/book.html

"Contrary to many stereotypes, delighting users is what gets a developer up in the morning and meeting user expectations inspires them to always do better. Consumers also buy things, and that is like fuel in the tank for open source development. Aside from the practical matter of funding salaries, competition for consumers is an important driver of innovation, and fires the imagination of developers."

Rick Spencer,
Ubuntu desktop engineering manager

The software provided by the GNOME Project (http://gnome.org/) is produced by a large community comprised of several thousand developers, translators, quality assurance testers, and documentation writers. Consumers are represented in the community by technical users and organizations that distribute GNOME technologies. And while the community reaches out regularly to non-technical end users and welcomes any that approach the community, these two worlds rarely interact. This article draws upon our experience within the GNOME Project to examine the question "Why do consumers and the community rarely interact?". Our insights may prove useful for other projects and consumers wishing more interaction.

Reasons for Limited Interaction

The GNOME Project provides both a desktop environment and a development platform. The focus of the desktop is ease of use, stability, internationalization, and accessibility support. The development platform provides an extensive framework for building applications that integrate into the rest of the desktop. GNOME 2.24 represents the first release of the GNOME Mobile development platform which brings standard desktop components together to provide a core platform on which distributors and handheld manufacturers can build rich programming environments.

The GNOME community currently has over 14 million users that consume GNOME technologies and software in many ways. Some consumers use the full desktop, others use its multimedia applications for photo editing or listening to music, while others run its software on their smart phone. When doctors use a Supersonic Imagine scanner, they are using GNOME technologies to detect breast cancer. When children around the world use their One Laptop Per Child XO, they are using GNOME technologies. What is true in all cases is that the end user probably didn't directly download GNOME or its applications.

The GNOME developer community would like to hear more from end users. End users' feedback can improve the product, making it better for everyone. Also, users might be doing something the developer never thought of and the developer can make that feature better for everyone. In spite of the fact that developers would like to meet consumers, there are several reasons that non-technical users and the community do not interact. These reasons are typical of most project/consumer interactions and are not limited to the GNOME community.

The main reason consumers and the community don't interact is because most users, whether they are using a desktop, a netbook or a cell phone running GNOME Mobile technologies, get their technology from a vendor, not the GNOME community. Many consumers do not even know that they are using GNOME. They might be listening to music with Banshee, editing their photos with Gimp and managing their finances with Gnucash and have no idea that the underlying technology is GNOME.

When it comes to interacting with the GNOME community, the user first has to become aware of the community. When most users encounter a problem, they end up on a GNU/Linux distribution site. Or, if they were using a Garmin 880 GPS, they call Garmin, not GNOME. In most cases, this is a good thing. The consumer is using a product from a company that happens to include or be built upon GNOME technologies. While the GNOME community of developers would like to hear their feedback, it doesn't always make sense to introduce an additional relationship to the end user, not when they've paid for a warranty or support from a vendor.

Some users don't interact with the GNOME community because they are unfamiliar with its communication tools. Most non-technical users interact on the user forums (http://gnomesupport.org/forums/). These forums are easy to find via a Google search and it is easy for a new user to ask a question. In contrast, most GNOME developers communicate on IRC (http://live.gnome.org/GnomeIrcChannels) and mailing lists (http://mail.gnome.org/mailman/listinfo). IRC users tend to be more technical and IRC conversations are not typically picked up by search engines.

Lastly, users often look for the GNOME community only after they've encountered a problem and they discover that to report it they need to learn how to post to the bug tracking database bugzilla (http://bugzilla.gnome.org/) or subscribe to a mailing list. By the time they are reaching out to the community, they are not looking for relationships or to make suggestions. Rather, they have an immediate problem that they need help with. Instead of finding someone who can immediately answer their question (although that does happen), the process directs them to tools that they may not be familiar with.

Importance of User Contributions

If consumers don't participate in an open source project through the traditional community channels, how do they participate? How can an end user contribute and be part of the community? We often talk about users being the ultimate testers and contributing bug reports, and if they are technical, perhaps a patch, or maybe some documentation. But users can contribute in many other ways:

Users test software. Some users do test software. Hopefully end users are testing already well tested software, but occasionally they run into problems. Or, they get excited about a new project and download a pre-release version that still needs testing. At the Maemo Summit (https://wiki.maemo.org/Maemo_Summit_2008), a Nokia manager publicly thanked a group of developers for being his guinea pigs. In a sense, all open source users are testers. Some contribute by reporting any problems they run into while others continue to make sure it works. Users also contribute feedback through blog posts, talks, and articles.

Users spread the word. Users who talk about their experience often recruit more users and developers. Consider a high school student who discovers Open Office and Google docs and saves her mother the money it would have cost to buy Microsoft Office. She tells her friends and other parents, potentially creating more happy users which spread the word about the software they are using. GNOME software now includes great integration of system and applications (like plugging in your camera and having F-spot automatically import pictures) as well as very reasonably priced software that meets accessibility needs. Many people are finding that many accessibility solutions from speech synthesis to drawing tablets to eye trackers work well in GNOME and the software is free.

Free software is also a good fit for people that want to promote free trade.

Users motivate developers. When users talk about, blog or demonstrate GNOME, they are not only spreading the word, they are also giving kudos to the developers. Such recognition is incredibly motivating. Developers enjoy writing software, but they also enjoy hearing how that software is changing people's lives, from the kid using a XO in Africa to the kid in the US that can't see using a screen reader. Knowing that your software is helping someone else is very motivating.

Users lend credibility. To a project, credibility enables growth. Hearing that Firefox has millions of downloads or that GNOME has 14 million users definitely encourages new users to try that software.

Financial contributions. Directly and indirectly, users contribute to the financial well being of GNOME. Sometimes they purchase support or products from companies that sponsor the GNOME Foundation. Sometimes they contribute directly to the GNOME Foundation through Friends of GNOME (http://www.gnome.org/friends/). All of these contributions flow back to efforts that support the project from hackfests to usability studies.

Participating in user groups. Through user groups (http://live.gnome.org/UserGroups), people help support each other and spread the word about GNOME.

Increasing Interaction

While more interaction between developers and users can produce better products, many people have heard of a story of a user getting ignored or getting a negative response from the community. Unfortunately this sometimes happens as there are many more users than developers and the best channels for communicating aren't always clear.

Here are some ways for consumers and developers to work together:

Using forums. Given that there are a lot more users than developers, providing some structure to the feedback can help developers respond and react to feedback. Whenever possible, use the appropriate forum, not a direct email to a developer.

Specific issues. If there's a specific issue with a project, whether it's a bug or just a use issue, bugzilla is the best place to go. First, search to see if the issue has already been reported. If so, just add to the existing report. Users talking about very specific issues in the right locale are very useful to the project as they can help identify the problem, come up with ideas to fix it, and verify that it has been fixed.

Using intermediaries. Most people obtain GNOME through a third party and have associated channels of support and communication. While it adds a layer to the communication, it is still useful information and a valid way to communicate. The GNOME distributors work closely with the GNOME community and their work reflects the feedback they receive from consumers.

Outreach programs. There are a number of ways even novice consumers can interact with developer communities. Conferences often offer tutorials or hands-on, get started days. Users can take advantage of these avenues to get advice from more experienced users or developers and also to demonstrate what's giving them trouble. College professors are also good outreach coordinators. Several universities are teaching students computer programming skills by having them work on GNOME technologies.

Helping in non-technical areas. Developers can always use feedback in the form of usability studies or requirements gathering. While these work best when run by a member of the team, projects can always use help setting them up.

Existing discussion channels. There are a number of other places where conversation happen. All of these places are text based, online channels. Some cater towards end users like the GNOME Forums and user groups. Others tend to have more developers than consumers such as the mailing lists and IRC channels. When using these channels, be sure to be on topic. There are lots of mailing lists and IRC channels as everyone has a specific audience and topic.

Summary

Users are the target audience for GNOME. The GNOME Project's goal is universal access. Making sure technology is available to anyone, not just technical people, regardless of culture, financial well-being or physical ability is what GNOME is all about. The fact that people use it makes the project a success, the developers happy, and the whole thing going.

The GNOME developer community works really hard to understand its users and to make sure that the default options, the ones that most users will first encounter, make sense. They have also put a lot of thought and effort into making sure that GNOME is accessible to all users regardless of ability. They work hard to communicate their core values to all users: free software, internationalization and localization, usability, and accessibility, and to make sure users are welcome, especially at GUADEC (http://www.guadec.org/), our annual conference. But whether users show up in the developer community or not, they are definitely contributing members of the GNOME community.

Stormy Peters is Executive Director of the GNOME Foundation, a 501(c)3 non-profit which works to further the goals of the GNOME Project. She has established relationships with the open source community and industry sponsors. Stormy has been involved with the GNOME Foundation, having been one of the founding members of the GNOME Foundation Advisory Board in 2000. Her previous positions include that of Open Source Program Manager at Hewlett-Packard and Director of Community and partner programs at OpenLogic. Stormy graduated from Rice University with a B.A. in Computer Science.

"When you start community-building, what you need to be able to present is a plausible promise. Your program doesn't have to work particularly well. It can be crude, buggy, incomplete, and poorly documented. What it must not fail to do is convince potential co-developers that it can be evolved into something really neat in the foreseeable future".
The Cathedral and the Bazaar

In this paper, we present a series of lessons learned on how to effectively run a small open source community. These lessons are based on my own experience as the leader of VoiceCode (http://sourceforge.net/projects/voicecode/), a project that aims at developing an integrated programming-by-voice toolbox. It provides tools that allow programmers with Repetitive Strain Injury (RSI) to write computer code by talking to their machine instead of typing.

The VoiceCode project started in 1999 by the National Research Council of Canada, and was first officially released in 2003. The system is now at a point where it can be used by programmers to do real work, and there have been over 9,100 downloads so far. The project has also attracted the attention of the media (http://www.newscientist.com/article/dn9066-software-lets-programmers-code-handsfree.html).

In the process of leading this project, we have learned many important lessons, too many to discuss exhaustively here. We will however share three that seem particularly important.

The Elusive "Plausible Promise"

When I started the VoiceCode project, I naively expected that I could get away with developing only a small core part of the system. Just enough to get people excited and foster the emergence of a community of dedicated contributors that would extend it into something usable. After all, this approach worked for Linus Torvalds, and it is what Eric Raymond advocates under the term "plausible promise" (on page 47 of The Cathedral and the Bazaar, http://www.oreilly.com/catalog/9780596001087/).

However, producing a plausible promise turned out to be more challenging than we expected. To date, a full six years after the VoiceCode project was started, only two other people have contributed code to it: David Fox (then an astronomy student at Harvard who now works at Nuance) and Stuart Norton (then a student at UC Santa Cruz who now works at Borland). Our biggest mistake was that we tried to get people excited about VoiceCode's potential by focusing first on its coolest feature: the ability to translate pseudo-code utterances like "for each index up to ten do the following" into native programming code. We demonstrated this feature early on, but not in a way that could be used for real. You had to simulate the act of speaking by typing text in a DOS console window, and the result appeared in a console window instead of inside an actual editor. While this gained us much kudos from the community of programmers-by-voice, it was obviously not perceived as a believable promise, because no-one actually contributed code to the project.

In my experience, people only start believing the promise when they can use the system for something real, even if it is small. Yet, if you are to co-opt people into investing countless hours in developing the system, it better be cool too. A good balance seems to be to "Think Big and Cool, but start Small and Useful". In other words, write down a clear, big and cool vision, but start by implementing a very small end-to-end piece of it that people can use right away. With VoiceCode, we did "Think Big and Cool", but we failed to start "Small and Useful".

It's all About People and Collaboration, Not Software

It has often been noted that in software development, people and collaboration issues are more critical than technical ones. Open source development is no exception. It has developed practices that make collaboration easier. Indeed the corner stone of open source, the GPL license, was designed specifically to foster collaboration by allowing developers to build upon each other's work. Also, one of the most attractive things about open source for developers is that it allows them to engage in direct conversations with the users (usually through email), without having to go through an intermediary like a marketing department. Developers quickly become addicted to such direct feedback on their work.

While the GPL and direct contact with users help with collaboration issues, open source development presents special challenges of its own in that respect. Because team members often live miles away from each other and sometimes in different time zones, it is harder for them to coordinate. Moreover, because the project usually has no funds, it is not economically possible for team members to travel and meet face to face. Finally, the fact that lines of authority in the project are usually inexistent means that people have to work harder to agree on things and work towards a common goal.

With VoiceCode, we did a number of things to facilitate collaboration. First, even before we started the project there existed a mailing list called VoiceCoder (http://tech.groups.yahoo.com/group/VoiceCoder/), founded by Brad Litterel in 1999, where programmers-by-voice discussed particular issues they faced and exchanged best practices. This forum proved invaluable for connecting with our constituency of future users, and to define a vision for the project. However, we found email contact to be somewhat limiting. Therefore in March 2000, Eric Johansson and I organized a one day face-to-face workshop in Boston (a hotbed for speech recognition) where 21 people interested in programming-by-voice met to discuss and exchange ideas. This workshop was instrumental in shaping the VoiceCode White Paper (http://iit-iti.nrc-cnrc.gc.ca/publications/nrc-48547_e.html), a document that laid down the vision which the team pursued throughout the project.

On a more day to day basis, we used frequent emails and a wiki site to coordinate. Combined with a clear common vision, this turned out to be fairly efficient, but not as efficient as face-to-face contact. At one point, David Fox and I kept engaging in heated technical debates that took several days and sometimes close to a hundred emails to resolve. Sometimes the tone of the messages bordered on flaming, which kept both of us awake at night. Eventually, we decided that the spirit of our collaboration would be greatly improved by having a weekly one hour meeting on the phone. This turned out to be instrumental in allowing us to keep a good working relationship and to discuss difficult issues more efficiently and in a spirit of mutual respect.

In summary:

- make sure you have a good way to communicate with your constituency of future users very early on, ideally even before you have written a single line of code

- arrange for key players to sporadically meet face to face

- keep the team working towards a same goal by defining a clear vision that everyone can buy into right from the outset

- use lots of asynchronous communication (email, wiki) as the basic mechanism for communication within the team, but make sure you meet regularly in a more synchronous medium to foster good working relations

Take a Leaf from the Extreme Programming Book

I knew from day one that an open source project would not be amenable to top-down waterfallish management. After all, how much control can you, as a leader, exert on a team of volunteer developers? But at the same time, I knew that I would have to put some sort of lightweight process in place to control (and hopefully leverage) the somewhat chaotic nature of an open source project. I just didn't know what.

Then in 2002, I attended a one week training workshop on Extreme Programming (XP, http://www.extremeprogramming.org/rules.html). I could immediately see how the practices of XP were immediately applicable to an open source project. In particular, practices like close collaboration with customers/users, small iterations, release planning, simplicity, and no functionality added early provided a clear and operational way of rapidly delivering a "believable promise". I could also see how test driven development, collective code ownership, coding standards and system metaphor would help newcomers in understanding the code and making them feel comfortable with it (something which is important for recruiting new contributors).

I shared what I had learned with my co-developer David Fox, and we immediately started applying some of the practices in VoiceCode. Altogether, using XP practices made such a difference that I found myself wishing I had learned about them before I had started the project. I am convinced that it would have focused us on delivering something smaller and useful end-to-end early on. Unfortunately, by the time I learned about XP, the system had already grown big and ambitious, and we felt compelled to complete what we had started. Before you start an open source project, learn as much as you can about Extreme Programming and other Agile Development (http://www.agilealliance.org/) methods. This may save you a lot of headaches.

Conclusion

Dumping code on the SourceForge.net site does not an open source project make. Running a successful open source project is hard work. It requires that you pay a lot attention to delivering end-to-end value early on, ensuring collaboration amongst developers and with stakeholders, and developing light handed processes for managing and leveraging the somewhat chaotic nature of open source development.

This article is based upon NRC publication 48551 (http://iit-iti.nrc-cnrc.gc.ca/iit-publications-iti/docs/NRC-48551.pdf) which is Copyright 2006 by the National Research Council of Canada. It is republished with permission of the NRC.

Alain Désilets is a Research Officer at the National Research Council of Canada. For the past 15 years, he has been involved in the rapid development of prototype software that uses bleeding edge Human Language Technology (ex: Speech Recognition, Machine Translation, Natural Language Understanding). For the past five years, his work has focused on Computer Assisted Translation tools.

FLOSS-Like Education Transfer Report

Copyright: Andreas Meizsner et al

From the Preposition:

Free/Libre and Open Source Software (FLOSS) communities are not only an exemplar for successful software development, but also for well working learning environments. Yet little is known about how learning occurs in the FLOSS communities and what the underlying success factors are. FLOSS communities might be seen as an example of 'Best Practice' in how ICT can help to improve education in terms of learning processes, up to date content and open inclusive education where no learner is excluded from participation. The FLOSSCOM project was undertaken in order to evaluate how learning in FLOSS is organised and if, to which degree, and how FLOSS learning principles can be transferred to and used for the improvement of ICT supported formal education.

http://flosscom.net/index.php?option=com_docman&task=doc_view&gid=201&Itemid=116

Slouching Toward Open Innovation: Free and Open Source Software (FOSS) for Electronic Health Information

Copyright: Greg R. Vetter

From the Introduction:

Gartner, one of the most respected market research firms for information technology, recently called open source software the "biggest disruptor the software industry has ever seen and postulated it will eventually result in cheaper software and new business models." The degree to which this prediction materializes depends on many influences, one of which is the subject of this Article. I argue that some software markets are more favorable for open source approaches than others. Using a case study of one particular software market, this Article develops a tentative framework of factors characterizing markets likely to disfavor contemporary approaches in free and open source software (FOSS). A software market is intimately intertwined with the licensing techniques employed in the market. This suggests that demand side responses may change based on new licensing techniques - an effect that is already a feature of the FOSS movement. If identifiable characteristics describe FOSS-disfavoring markets, this perspective may lead to the development of new FOSS techniques to enable open innovation in those markets. The last part of this Article outlines directions to facilitate this process.

http://opensource.mit.edu/papers/VetterSlouchingTowardOpenInnovation-FOSS
forEHI_6.29.2008.pdf

January 14

Startup Club Kickoff

Ottawa, ON

Startup Club is a group of startup enthusiasts that come together and volunteer to help each other directly. Space for the kickoff is limited to 30 people and lunch will be provided. Pre-registration is required.

http://startupclub.eventbrite.com/

February 11

CloudCamp

Toronto, ON

CloudCamp was formed to provide a common ground for the introduction and advancement of cloud computing. Attendees can exchange ideas, knowledge and information in a creative and supporting environment, advancing the current state of cloud computing and related technologies. As an informal, member-supported gathering, we rely entirely on volunteers to help with meeting content, speakers, meeting locations, equipment and membership recruitment. We also have corporate sponsors that provide financial assistance with venues, software, books, discounts, and other valuable donations. Anyone may attend a meeting, there are no fees or dues.

http://cloudcamp.com/?page_id=219

February 20-21

Northern Voice

Vancouver, BC

The Norther Voice conference is a yearly blogging and social media conference currently in its 5th year. The focus of the conference is a coming together of all things blogging and social media to increase our knowledge base.

http://2009.northernvoice.ca/

February 21-22

PodCamp Toronto

Toronto, ON

Podcamp Toronto is a FREE "unconference" for all those interested in all things podcasting, blogging and new media. Amateurs, pros, newbies and veterans are all welcome.

http://podcamptoronto.pbwiki.com/

February 28 - March 3

TikiFest

Montreal, QC

A tradition in the TikiWiki community, a TikiFest is when there is a meeting between at least 2 TikiWiki contributors that don't usually meet. It is a great opportunity for Tiki users and Tiki power users to meet some of the developers and learn more stuff.

http://tikiwiki.org/TikiFestMontreal2009

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

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

1. Does my research or experience provide any new insights or perspectives?

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

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

4. Am I constantly correcting misconceptions regarding this topic?

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

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

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

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

2. Know your central theme and stick to it.

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

4. Write in third-person formal style.

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

Upcoming Editorial Themes

February 2009:	Commercialisation Guest Editor: Robert Withrow, Nortel
March 2009:	Geospatial Guest Editor: Dave McIlhagga, DM Solutions
April 2009:	Open APIs Guest Editor: Michael Weiss, Carleton University
May 2009:	Open Source in Government Guest Editor: James Bowen, University of Ottawa
June 2009:	Women in Open Source Guest Editor: Rikki Kite LinuxPro Magazine

Formatting Guidelines:

All contributions are to be submitted in .txt or .rtf format.

Indicate if your submission has been previously published elsewhere.

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

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

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

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

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

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

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

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

Copyright:

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

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

For pricing details, contact the Editor dru@osbr.ca).

eclipseCON™ 2009

Sixth Annual Gathering of the Eclipse Community

Attend Eclipsecon 2009

EclipseCon is the premier technical and user conference focusing on the power of the Eclipse platform. From implementers to users, and everyone in between, if you are using, building, or considering Eclipse, EclipseCon is the conference you need to attend.

Over 240 Sessions and Tutorials Including:

- Java Development
- Mobile and Embedded
- Modeling
- OSGi
- Reporting
- Rich Client Platform
- SOA Development
- Test and Performance
- Technology and Scripting
- Tools
- Web Development
- Business
- C/C++ Development
- Database Development
- Industry Vertical
- Project Mashups

This is your opportunity to get in-depth technical information from the Eclipse experts, learn the latest tips and techniques for using the tools, network with fellow enthusiasts and experience the breadth and depth of the Eclipse community. Attending EclipseCon will expand your knowledge and make Eclipse work better for you.

March 23rd - 26th
Santa Clara, California

Register at:
www.eclipsecon.org

Ontario

The Talent First Network program is funded in part by the Government of Ontario.

TIM Carleton UNIVERSITY

The Technology Innovation Management (TIM) program is a master's program for experienced engineers. It is offered by Carleton University's Department of Systems and Computer Engineering. The TIM program offers both a thesis based degree (M.A.Sc.) and a project based degree (M.Eng.). The M.Eng is offered real-time worldwide. To apply, please go to: http://www.carleton.ca/tim/sub/apply.html.