OSBR.CA

The Open Source Business Resource

OCTOBER
2008

OCTOBER 2008

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

Having been involved with open source projects since before the term "open source" was coined, I'm often asked to define "open source". My usual one word answer is not "code" or "license", but "community". For the essence and differentiator of any open source project is defined by the individuals it attracts and how their interactions provide the momentum needed to sustain the project's long term goals.

This issue of the OSBR is about "Building Community". Developers and project leaders provide insight into how communities attract and maintain new members and the stages a project goes through as it matures. This issue also includes summaries of research into how the interactions between open source developers and commercial interests are mediated through Foundations and how licensing can control who chooses to contribute to an open source community.

As always, the authors and other readers appreciate your comments and references to additonal resources. You can send these to the Editor or leave them on the OSBR website or blog.

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 and DNSStuff.com and is the author of the books BSD Hacks and The Best of FreeBSD Basics.

To succeed, an open source project needs a thriving community. This issue focuses on building these communities. It includes community development strategies employed by the Eclipse and NetBSD open source communities, the role licenses and governance play in shaping communities, as well as lessons learned from examining communities outside the open source domain.

Alessandro Rossi and Marco Zamarian from the University of Trento compare the development of proprietary and open source games in terms of the artifacts and actors that are involved in the development process.

Stefano De Paoli, Maurizio Teli and Vincenzo D'Andrea argue that open source licenses set the boundaries around who will participate within a project.

Zhensheng Xie, a developer from EA Mobile, examines the role of open source software foundations (OSSF). He identifies three types of OSSF governance structures and develops a set of propositions based on his observations of six OSSF.

David Maxwell, Coverity's Open Source Strategist, and Lubomir Sedlacik, software engineer at Sun Microsystems, are long-standing members of the NetBSD Project, one of the oldest modern open source software projects. Based on their experience with NetBSD, the authors provide advice on how to build a strong, vibrant community.

Thomas Prowse, a Partner with Gowlings, advances the thesis of commons sourcing which is increasingly being found at the core of new commercial initiatives. Abundance rather than scarcity is at the center of the new model, but some scarcity needs to be preserved to provide commercial differentiators for companies.

Monica Mora is a member of the technical committee of CIDETYS in Panama. She discusses the main activities that CIDETYS will focus on in the near future: creating a technology literacy program, setting up a technology incubator, and transferring knowledge on grid computing.

Cliff Schmidt, the Executive Director of Literacy Bridge, a non-profit start-up that uses open source software, open hardware, and open content to solve some of the world's most challenging problems: global poverty and disease. This article describes the Talking Book Project and describes how principles of successful open source projects are being applied to improve global literacy and access to information. This project demonstrates the power of combining community and appropriate technology to change the world.

We anticipate that you will enjoy the different perspectives on community building collected in this issue and that you will find the authors' messages relevant to your own efforts, whether you want to better understand how open source communities work, or whether you are involved in setting up your own open source community.

Yours,

Michael Weiss

Michael Weiss holds a faculty appointment in the Department of Systems and Computer Engineering at Carleton University, Ottawa, Canada, and is a member of the Technology Innovation Management Program. His research interests include open source ecosystems, services, business process models, social network analysis, and product architecture and design. Michael has published on the evolution of open source communities and licensing of open services.

"...looking at open-source practices in the computer game community [...] what we see is something different than what's advocated in the principles of software engineering."

Walt Scacchi
http://tinyurl.com/4rkbvo

Free/Libre Open Source Software (F/LOSS) development practices are gaining momentum in the computer game industry. This traditionally proprietary industry is becoming more interested in the F/LOSS paradigm for developing complex software projects. Managers and developers need to understand the potential that incorporating F/LOSS practices into their proprietary production cycle offers.

Typically, proprietary and F/LOSS software development processes are compared in terms of property rights, revenue distribution and power within a network of actors. Coordination and control practices, mediating artifacts and development tools, and the interactions between the different actors involved in the development are often neglected. Proprietary and F/LOSS development differ in terms of the knowledge exchanges between the relevant actors and the different strategies employed to overcome information asymmetries. Recognizing this difference is an essential step for evaluating how proprietary, closed–source software houses can benefit by integrating various F/LOSS practices into their development agenda.

Computer Gaming and the F/LOSS Community

The interaction between F/LOSS and computer games development goes back at least ten years and can be associated with two distinct, albeit intertwined trends: i) the efforts undertaken by some large industry players to leverage user–base potential into the development of proprietary computer games; and ii) the emergence of user/developer communities pursuing independent gaming projects.

These trends originate with end users modifying the code to enhance and customize their gaming experience. Long before incorporating end users into the production cycle became a common practice, many games, such as Wolfenstein 3D, gained popularity after their formats had been reverse engineered. This unlocked the possibility for the masses to edit default specifications (such as scenarios, maps, characters, and rules of interactions), thus "modding" the game environment according to the user's preference. Some software game companies, such as id Software and Epic Games, became proactive and started to explicitly co–opt their user–bases into the production cycle by eliciting the development and sharing of user–generated content. In most cases, they deliberately opened the peripheral specifications of their gaming platforms or shared part of the code according to liberal licensing terms. Some provided specific tools such as map editors, aimed at easing the participation of technically unskilled users.

The advent of online gaming communities reinforced this tendency. Online gaming allowed individual gamers to share their interests within a community. It is through this superficial involvement that most gamers became first "modders", and, over time, outright developers. Early communities revolved around projects building upon large proprietary software, which was later released as open source. This allowed the modification and development of F/LOSS variants of proprietary versions of games. The emergence of independent open source communities developing original games has since become more frequent.

Computer gaming seems to be very popular among F/LOSS communities: as of October, 2008 the SourceForge repository (http://sourceforge.net/softwaremap/) shows 29,831 game projects. Software developed in these communities includes role–playing games, simulation games, Multi User Dungeons, first person shooters, arcades, and board/card/strategy games. SourceForge's activity rank shows that among the 100 most active projects, 7 belong to the category "Games/Entertainment". Almost 60% of the computer game projects have end users as their intended audience. The remaining ones are software projects such as toolkits, modeling, rendering, and animation engines--frameworks that target developers.

Nature of F/LOSS Communities

It is difficult to sketch the development cycle for the typical F/LOSS computer game project due to the highly heterogeneous nature of F/LOSS communities. While early studies have convincingly contrasted F/LOSS development with closed source models, they subsumed the existence of "a unified community with shared values, motives, and development approaches" (http://firstmonday.org/issues/issue9_6/tuomi/index.html). Overall, these studies have failed to understand the nature of the processes of F/LOSS deployment, development, and implementation, as well as the contexts in which these processes occur. Recently it has been recognized that it is more fruitful to examine F/LOSS development in terms of attributes of a complex socio–technical system rather than technical enhancements. (http://www.firstmonday.org/issues/special10_10/lin/index.html). This new approach focuses on the interplay between the different actors and the various artifacts for interpreting the complex nature of the socio–technical processes that occur within F/LOSS communities.

There are several artifacts in F/LOSS communities that have the "ability of connecting different social worlds and sustaining socio–technical interactions" (See http://www.maurizioteli.eu/publications/DePaoliTeliDandreaInProgress.pdf by the authors of the next article in this issue). This can be seen in licensing schemes, which act as mechanisms involving interested participants such as hackers from the F/LOSS community and software corporations in the OSS industry. Different kinds of licenses provide different incentives for participation in the innovation process, according to the specific business model employed by the firm. For instance, GPL-like licenses, unlike BSD-like licenses, restrict the incorporation of incremental and cumulative innovations into a proprietary product. This deters participation from firms that place a strong emphasis on direct revenues schemes, while attracting firms from complementary industries or with complementary business models.

Different licenses represent artifacts that structure activities and practices through the imposition of differing political and technical boundaries, affecting the participation of actors and the implementation of innovations. Accordingly, the free/open identity is the result of negotiation between participants in everyday interactions and practices. For instance, the decision to release code under a particular license is interpreted as the result of the participants' negotiation of the direction in which one wants the technology to evolve, the boundaries of the community to be set, and so forth.

Despite the difficulties in deriving a unified model of the development process, it is still possible to generally describe which processes and practices are popular in F/LOSS computer game projects. In doing so, we will try to highlight the major differences with the proprietary model.

A typical project revolves around a community of stakeholders in which end users and developers have overlapping roles and identities. The social structure of the project is onion–like, making it possible to distinguish between different levels of involvement in the project (http://www.firstmonday.org/issues/issue10_2/crowston). Central to the project are the core developers that contribute the largest part of the code and who may take part in fundamental decision making for the project, such as the design of the architecture. External layers represent a more partial involvement in the development activities and include co–developers that provide: i) patches and bug fixes; ii) localization activities; or iii) more episodic contributions of a larger piece of code. A more external layer is represented by active users that are involved in various feedback activities, such as testing and bug reporting. Usually, a fundamental role is played by the project leader--often the initiator of the project-- who is strongly involved in the development of the early releases of the software. Over time, the project leader undertakes more general activities of project coordination.

Typically, contribution efforts in a project are skewed and centralized into a very small subset of participants (http://www.firstmonday.org/Issues/issue7_6/krishnamurthy). This has proved particularly true for fundamental activities such as coding, suggesting the existing tension regarding the preservation of conceptual integrity and the development direction of the project. This phenomenon is less evident for less critical activities such as bug tracking and fixing (http://conway.isri.cmu.edu/%7ejdh/collaboratory/research_papers/TOSEM-draft.pdf), in which more man–power is always welcomed.

The overall picture is rather distant from Raymond's idea of a "great babbling bazaar of differing agendas and approaches [...] out of which a coherent and stable system could seemingly emerge only by a succession of miracles" (http://www.firstmonday.org/issues/issue3_3/raymond). Most of the time, contributions to the project are highly monitored and peer–reviewed by various forms of hierarchical control in order to preserve, albeit within a highly decentralized development environment, the conceptual integrity of the project.

Developing Game Software in a Fully Proprietary Environment

In order to understand the contrast between F/LOSS and proprietary game development, we describe the typical development process occurring in the game console software industry, which is normally fully proprietary from top to bottom.

Essentially, the console producer has two main sources of games for its product. It can use internal resources to both design the game and publish it. Or, approved external developers can submit project ideas to internal or independent publishers. Publishers, in turn, decide on the marketability of the proposal and whether or not to fund the project. If approved, the developer produces the game code and releases it to the publisher who sends the code to the console manufacturer. The manufacturer then gives final approval of the software project as a whole after one or more rounds of code testing--in the console market, the game needs to be released with zero–defect quality as a goal. The practice, common in the PC games market, of a rushed release that is patched with remote upgrades by the end user, has not been feasible up to the current generation of consoles.

Once the release is accepted, the console manufacturer takes care of the production of the physical copies of the game. In turn, the manufacturer releases the copies of the game back to the publisher who manages the downstream relationships with distributors and retailers.

The complex interplay between these actors is usually described in terms of two kinds of exchanges. The first, direct monitoring of the output of the development and publishing cycles, is linked to the negotiating power of partners and finds its natural expression in contracts and legal agreements. The second exchange consists in the financial flows between the involved parties.

With this perspective in mind, it appears obvious that developers are marginal actors in the network of interdependencies that ultimately generate the end product (http://joeg.oxfordjournals.org/cgi/content/abstract/6/2/151), as they are completely dependent on both the console manufacturers and publishers. However, this view does not capture one of the essential ingredients of developing a successful piece of gaming software: the ability to tap strongly asymmetric sources of ideas, creativity, and technical coding expertise. With this perspective, we discover a few layers of complexity that can hardly be considered secondary to the whole process. We believe that the social role of development tools and middleware as both control tools and boundary objects between idiosyncratic domains of knowledge and ideas is key to understanding this development model.

Most projects involve the conception and development of several functional code modules that are combined in the end product. We distinguish two categories of modules: i) key modules that set the product apart from competing products and are where the developers concentrate most of their expertise; and ii) peripheral modules which complete and integrate key components and are less critical for the success of the game. Peripheral modules often rely heavily on third party development tools to port the process from the high level code produced by the creative artists within the development team into the specific, compiled code that can be run on any hardware combination. This allows developers to concentrate on the internal mechanics of the software they are producing without investing resources in researching the specific hardware architectures of each platform.

The importance of these tools in the development process cannot be overstated. According to developers, trying to interact directly with the development libraries provided by the console manufacturer has several shortcomings. First and foremost, there are problems related to the different knowledge domains that the parties control. The distance between competences is particularly relevant in the case of graphic artists, which are usually part of the peripheral development team, and the manufacturer's developers that produce the low–level libraries provided with the development kit.

This knowledge gap cannot be typically filled by most developers--hence the importance of the middleware producers. Technically savvy developers might, in principle, be able to bridge this gap. However, internal development of tools is usually economically non–viable. Thus the tendency to buy the tools on the market. The downside derives from relinquishing control to third parties of the implementation of high level ideas into working code. We need to underscore that, at least in principle, the more complex the hardware and the faster the

cycles of development for these platforms, the less a developer will likely adopt a make strategy for its middleware. We believe there is a progressive loss of competence on the translation mechanisms from the high level, descriptive languages typically used by creative artists into the lower level code required by the different platforms.

A Comparison and Some Critical Remarks

Comparing proprietary and F/LOSS in terms of property rights on the software code produced can be misleading. A first useful dimension along which to compare these processes can be defined by the actors involved. In the F/LOSS development process, there is typically a good level of technical knowledge homogeneity between the core developers, content providers, or more peripheral users. User/developers are a characteristic feature of F/LOSS development, and there is no equivalent in the proprietary domain. However, these actors are severely limited from a coordination point of view, being seldom co–located and often not sharing a clearly defined production plan. In the proprietary case, we find the opposite. The actors involved in the development process do not typically suffer from coordination problems. Within the company, coordination is solved by hierarchy while inter–firm coordination is managed through a double mechanism of contracts and quality control interactions. However, knowledge quality and heterogeneity constitute the main problems. In console development, users are not considered a meaningful source of feedback and even the actors directly involved possess highly differentiated competencies, strictly associated with the phase of the development process they specialize in.

We observe the emergence of two differentiated sets of tools used by developers. In the F/LOSS world, coordination tools such as mailing lists and CVS repositories are prevalent, but software tools used to directly produce other pieces of code are seldom employed. In the proprietary console software development market, tools bridge high–level, creative software pieces, usually produced by artists or storytellers with a very shallow programming experience, and lower–level code that can be compiled on a given platform. The presence of asymmetries in knowledge domains between the different actors creates room for third parties to introduce tools that reinforce these asymmetries. If a manufacturer stops making lower level software, it will eventually lose that ability either directly, because of a lack of practice, or indirectly, by losing tech savvy personnel or failing to attract competent programmers.

In contrast with what typically happens in F/LOSS communities, proprietary developers have a limited say in the way development tools are reshaped and modified by either the console manufacturer for the inner layers of the software kit, or by the middleware producers for the outer layers that interface with the high level language. In this regard, we might add that middleware mediates between the console hardware producer and the end product software developer not merely from a technical standpoint. In fact, the ability to filter, select and scrutinize the middleware producers' work is one of the main tools that the console producer has to influence the developer's work. By contrast, F/LOSS computer gaming communities can be thought of as an archetypal instance of a user–driven or community based innovation model in which end users are viewed as a fundamental driver in the innovation generation process.

It is apparent that the interplay between property rights on the final code, coordination needs, relative heterogeneity of actors' knowledge domains and the tools they adopt is very complex. However, the analysis of this interplay is central when trying to understand new trends emerging in the proprietary software development world, which seems to be more and more bent towards capturing features typical of F/LOSS development into their production mode.

This article was adapted from Designing, Producing and Using Artifacts in the Structuration of Firm Knowledge: Evidence from Proprietary and Open Processes of Software Development. Università degli studi di Trento. DISA Technical Report, 116, 2006.

Recommended Resources

Understanding the Requirements for Developing Open Source Software Systems
http://tinyurl.com/4py5e4

Free and Open Source Development Practices in the Game Community
http://tinyurl.com/4azc9x

Alessandro Rossi is Assistant Professor of Management at the Faculties of Economics and Engineering, University of Trento. His research interests are related to managerial cognition and to the economics and management of innovation and new technologies. He is currently investigating how organizations design and produce complex artifacts, with particular reference to knowledge intensive industries and to the open source/open content paradigm of production.

Marco Zamarian is Associate Professor of Organization Theory and Behavior and Human Resource Management at the Faculty of Economics, University of Trento. His current research interests include organizational learning, knowledge creation and replication in geographically distributed contexts, the impact of IT artifacts on organizational knowledge, industrial clusters, and the evaluation of the effects of public subsidies to the private sector, in particular for technology acquisition and R&D activities.

"The creation and management of boundary objects is key in developing and maintaining coherence across intersecting social worlds."

Star and Griesemer
http://www.citeulike.org/user/jpassoth/article/1281399

The objective of this article is to examine how software licenses build and shape political and technological boundaries.

Licenses specify the permissions granted by the copyright owner to users. According to Lanzara and Morner in "Artifacts Rule! How Organizing Happens in Open Software Projects", these permissions are inherent in a set of practices which are strongly inscribed in the license. But, according to Lin (http://opensource.mit.edu/papers/lin2.pdf) that can mean different things to different community participants. The free/open character of F/LOSS is, therefore, the result of an intricate web of negotiations around the meanings of material artefacts.

We examine the cases of the Geographic Resources Analysis Support System (GRASS, http://grass.itc.it) and the OpenSolaris (http://opensolaris.org/os) operating system. The first project is GPL licensed software developed by a worldwide community of voluntary programmers; the second project is sponsored by a company and released under the Common Development and Distribution License (CDDL, http://www.opensource.org/licenses/cddl1.php) license.

Licenses and Boundary Objects

We will conceptualize licenses as boundary objects comprised of textual artifacts. A boundary object inhabits "several intersecting social worlds and satisf(ies) the informal requirements of each of them. Boundary objects are objects which are both plastic enough to adapt to local needs and the constraints of the several parties employing them, yet robust enough to maintain a common identity across sites" (http://www.citeulike.org/user/jpassoth/article/1281399).

By considering licenses as textual artifacts, it is possible to analyze what practices and political visions are inscribed in a license (robustness) before describing what happens when a license enters a specific community (plasticity). We observe that licenses not only determine the boundaries of the permissions granted by the copyright owner to the users, but they also set the boundaries around the possible human and non-human participants to a project.

In their plasticity, licenses allow communications among different political, technical and organizational positions, shaping the meanings of participation, alliances and coordination. Nonetheless, licenses acquire their form in the interrelation between human and non-human entities in development projects. The robustness of a license takes a plastic and contingent form when the license itself is shared among different social worlds.

GNU GPL v. 2.0

In 1983, MIT programmer Richard M. Stallman launched a project for developing an entire UNIX-compatible operating system known as GNU (GNU's Not Unix). According to Stallman's plans, with the GNU General Public License (GPL) “users will no longer be at the mercy of one programmer or company which owns the sources and is in sole position to make changes” (http://www.gnu.org/gnu/manifesto.html). Stallman wanted, in fact, to redefine users, sources, programmers and companies.

In order to impose his own version of reality on others, Stallman needed a powerful intermediary: "we needed to use distribution terms that would prevent GNU software from being turned into proprietary software. The method we use is called "copyleft"" (http://www.gnu.org/gnu/thegnuproject.html). The GNU GPL was born as the intermediary or "the method" for ensuring the project's goals. The second section of the GPL, known as the Preamble, indicates the most important problem that this license wants to address: "The licenses for most software are designed to take away your freedom to share and change it."

The problem is that the majority of software licenses act against Stallman's plans. They allow programmers and companies to "own the sources". Stallman conceived copyleft which uses copyright law against its usual interpretation. With copyleft, authors allow everyone to run, copy, and modify their program and to distribute modifications. Simultaneously, copyleft imposes some restrictions on the use of the software. The intent of the GPL is to prevent GNU software from ever being turned into proprietary software.

CDDL

The CDDL (http://opensolaris.org/os/licensing/opensolaris_license), was created by Sun to foster the construction of a common environment for developers and distributors of OpenSolaris. According to the CDDL FAQ (http://opensolaris.org/os/about/faq/licensing_faq), the need for the CDDL license emerges from the convergence of two different phenomena: the need to use file-based licenses and license proliferation.

The CDDL license specifies the actors whom the license tries to interest and mobilize: developers and distributors. The CDDL is a file–based license; it only protects the individual files belonging to OpenSolaris, not the program as a whole. One of the aims of this license is to enable the "creation of larger works for commercial purposes". This defines an important difference in intent between the CDDL and GPL and shapes the interests of potential participants.

The CDDL boundary excludes two groups. The presence of the copyleft clause (see 3.1. "Availability of Source Code", and 3.2. "Modifications" in the CDDL), chosen in order to provide "the protections and freedoms necessary for true open source" excludes the possibility of combining CDDL-covered software with code covered by the GPL. The second excluded group is those who want to distribute their modifications of the covered files in proprietary form.

The second issue to influence the shape of the license is license proliferation: the increasing number of file-based F/LOSS licenses.

The last aspect we want to highlight is the control enacted by the license by means of the license steward. The license steward is an entity (Sun, in this case) that has the right to change the license. Through this right the license steward enacts a form of partial control on the future of the project.

GRASS and GPL

GRASS was started at the beginning of the 1980s as a small development project of the United States Army Corp of Engineering Research Laboratory (USACerl) and was distributed as public domain software.

In 1996, USACerl stopped GRASS development and asked users to migrate towards proprietary GIS. In 1998, a new GRASS development team (GDT) was formed to re-launch a voluntary GRASS development community. The passage from USACerl GRASS Public Domain (version 4.1) to the new GDT GRASS (versions 4.2, 4.2.1, 5.0) marked a phase of uncertainty for the software copyright ownership. In 1999, after some negotiations, the GDT adopted the GPL v. 2.0 as the copyright license for GRASS software. The GRASS community agreed to use a well-known F/LOSS license.

The GPL copyleft represents a form of controversy and a source of discussions within GRASS mailing lists. Controversies around the GDT agreement on the GPL emerged many times as its copyleft clause excludes the participation of developers using incompatible licenses.

“Why GPL” as a discussion thread occurred within the GRASS developers mailing list in March 2001 (http://grass.itc.it/pipermail/grass5/2001-March). Some aspects of releasing GRASS under GPL were questioned by a list newcomer. We summarize his two main points and his attempt to impose a new license on the GRASS framework:

1. It is possible for the authors of the original code to re-release their code under the LGPL (lesser GPL, http://www.opensource.org/licenses/lgpl-2.1.php) or another license.

2. People who write plugins for the GRASS framework may want to use their own license.

The LGPL is a license of the GNU project of the Free Software Foundation (FSF, http://www.fsf.org/) to cover specific GNU libraries.

Unlike the GPL, it allows integration with proprietary software. The proposed re-release of GRASS under the LGPL would change the participants’ positions in the GRASS community. The LGPL would place the copyleft method on the program itself but would not apply any restrictions to other software linking with GRASS. LGPL would only require the modifications applied to GRASS to be released under the LGPL license.

In this community discussion, the identities, roles and desires of the enlisted entities are redefined in a new network of heterogeneous associations. This redefinition can be achieved through the choice of the LGPL by the authors of the original code. With the LGPL, the identity and roles of entities are once more stabilized as: i) GRASS could be integrated with plugins written for specific applications; ii) participants in GRASS could then use their own licenses: and iii) programmer's rights would still be protected by the LGPL.

At this point, the new associations must be tested. Will the newcomer be able to make the LGPL, rather than the GPL, the new license for the GRASS community? He is forced to defend his own associations against the GDT's previous agreement. In a mailing list discussion dated 22nd March, 2001, he states: “The only difference between the GPL and LGPL is software linked to the GRASS library would not have to be GPL. It offers the same protection of the software, but doesn't scare away people who do not want to release GPL software. And that's a big difference. If people are scared off because they can't make money using the freely given contributions of other, so be it.”

And the response on March 26, 2001: “If end users get accustomed to the proprietary enhancements, the owner of the

proprietary rights (gets) power over the GRASS development. The GPL is the license which protects against this and thus most firmly ensures the long term freedom of the software." Here emerges a protection of the GDT agreement on the GPL. For these GDT members, there is no reason to doubt the previous license choice. These members assume that, through a different license (even another F/LOSS license such as the LGPL), owners of the proprietary rights over a proprietary extension of GRASS can exercise some power on the overall system development.

In this way, a newcomer to the development list is asked to not contribute to the GRASS community if he does not agree with the GPL choice. The GPL copyleft boundary is clear: owners of applications released under a license incompatible with GPL are kept separated from GRASS development and its community.

CDDL and GPL Incompatibility

The mailing list thread (http://www.opensolaris.org/jive/thread.jspa?messageID=3815) discussed in this section started when a non–developer in the OpenSolaris community asked how it is possible to include software covered by the GPL license which is considered incompatible with the CDDL by the FSF. This thread includes several debates which took place between July and September 2005. The first started with the above question and reached a conclusion within two days, with the resolution (mainly by Sun's employees) that the GPL'ed software included in the project is composed of separated programs, not linked modules, so that each license is applied to the respective programs. Participation boundaries are shaped by technical ways to connect different software: the license issue acts as a boundary object between developers and non–developers in the definition of a technical concern.

The second round, which lasted more than one month, was started by a non–Sun member, who stressed aggressively the political differences enacted by the CDDL: "So stop with the pathetic FUD and start reading your licenses before flaming about them. Sun could have included an exception for the GPL (as did the MPL 1.1, from which the CDDL is derived) but they clearly chose not to for political reasons."

The post involved both part of the license text and the use of conflictual expressions like FUD (fear, uncertainty and doubt, http://en.wikipedia.org/wiki/Fear,_uncertainty_and_doubt). Four answers followed this post, the last of which contained a link to CDDL Reflections (http://blogs.sun.com/webmink/entry/cddl_reflections) that presented Sun's position on the subject. In this case, the political boundaries between different licenses and Sun's positioning in the F/LOSS panorama were defined through the action of a recognized spokesman.

The third round was debated even more vigorously, lasted one month, and involved a much larger number of posts. It began with a post by a non–Sun member sent both to the mailing list and to Richard Stallman. Here are the beginning and conclusion of the message: "Alright, I wonder about this myself as well. [...] Sun should be nurturing a cooperative and mutually beneficial relationship with the FSF. If they have not been doing this from the beginning, Sun should be extending an olive branch."

The long post included three passages aimed at supporting the author's argument: i) the need to ask directly the FSF's opinion; ii) the need for the license compatibility in order to make the project flourish; and iii) the author's withdrawal from the project in relation to the perceived hostility towards the GPL.

The author's participation and the alliance between the project and the free software movement are connected by the licenses. The argument continues and raises other issues: i) requests for modifying the CDDL and the GPL in order to improve their compatibility; ii) the GPL perception as a constitutive element of the open source movement; and iii) the definition of the OpenSolaris community and its participants/users. Consider these messages: "the bottom line is that opensource developers and users want their software to be GPL. (I)f it is not then these people will be turned off by opensolaris." And: "No, *some* users and developers want their software to be GPL. And just as those users will be turned off by OpenSolaris because it is not, there will be many that will be turned off if it becomes GPL. [...] The majority of users don't care what license a program is under. They just like good software."

This debate involves and shapes the meanings of different entities: i) open source developers and users; ii) a group of GPL zealots; iii) the majority of users; and iv) the participants in OpenSolaris. Many messages involved the CDDL's political positioning, mainly in relation to the distinction between the free and the open software movements (http://www.oreilly.com/catalog/open-sources/book/stallman.html).

In these discussions, the license is enacting relationships among the individuals participating in the project. Some participants do not accept being aligned with the discussion and they criticize the intermediary organization itself. The coordination of the entire project is questioned and the license becomes a boundary object that separates allies from enemies.

Conclusions

We have discussed how software licenses participate in the community life of two F/LOSS projects. Instead of examining many examples, we have performed an in-depth investigation of two large F/LOSS projects. We have focused on both the process of license choice and on discussions about an existing license. The licensing controversies which emerged in our cases allow us to argue against a homogeneous view of communities. While the majority of the sociological debate has taken the character of F/LOSS projects as universal, we argue instead that this character is negotiated in everyday practices. Our analysis has elucidated the existence of conflicts about the free/open character of communities, artifacts, and software code. We further suggest that the complex role of licenses needs to be understood not only from the point of view of the participants' rhetoric but also from the perspective of F/LOSS participants' practices. This shift would allow the social researcher a better understanding of the political, technical, and organizational boundaries around and among the communities.

Stefano De Paolo is a Post-Doctoral Fellow at the National University of Ireland Maynooth in Ireland. His research interests include computer security, legal protection of intellectual creations, and information technology and division of labor.

Maurizio Teli, PhD in Sociology and Social Research, University of Trento (Italy), has a background in Political Science. He is involved in and researches about the importance of FLOSS "practices of freedom" in the processes of organizing a community and producing technology.

Vincenzo D'Andrea is an Associate Professor at the University of Trento. His research interests include service-oriented computing, free and open source licensing, and virtual communities.

"The biggest surprise to me was the level of involvement that firms engage in with community forms on software development and standard setting in general. That is, forms that are not government sponsored nor formally constituted by partnership, alliance, or consortia agreements."

Siobhán O'Mahony, Assistant Professor of Management, UCDavis

Communities that develop open source software (OSS) are virtual entities on the Internet, not legal entities. Some open source communities establish open source software foundations (OSSF) in order to protect their intellectual property and carry out contractual arrangements. As legal entities, OSSF help communities attain their long-term goals, hold community assets, provide resources to communities, and balance interests amongst different stakeholders.

When OSS started to draw more business interests, commercial companies became involved with open source communities. The emergence of OSSF provides a good platform and opportunities for companies to exert their influence in a more direct way.

This article summarizes our recent research regarding the relationships between company involvement, governance, revenue, and OSSF.

OSSF Primer

Typically when an open source community incorporates as an OSSF, they seek to gain financial advantages from donations and tax exemptions. An OSSF is a non-profit organization (NPO) with a primary objective to support or to actively engage in activities of public or private interest without any commercial or monetary profit purposes (http://tinyurl.com/4eod4q).

A NPO is a corporation that can handle business dealings, sign contracts, and own property as any other individual or for-profit corporation. NPOs differ from for-profits in terms of taxes and governance. A NPO's governance structures preclude private financial gain. OSSF in the US are registered under tax-exempt 501(c)(3) or 501(c)(6). 501(c)(6) is reserved for “Business Leagues and groups such as Chambers of Commerce” and is a form of a business network in favour of pursuing members’ own business interests whereas 501(c)(3) organizations provide public benefits. Most OSSF registered in the US are 501(c)(3) non-profit organizations. In the US, both 501(c)(3) and 501(c)(6) NPOs are exempt from most federal income taxes. OSSFs registered outside of the US usually have similar benefits. From a legal perspective, an OSSF is a legacy NPO.

An OSSF must have at least one software project in which individuals are contributing to a new software release. The Apache Software Foundation is an OSSF while the Open Source Initiative (OSI, http://www.opensource.org/) is not as it focuses on standards and does not hold any software projects.

OSSF in the Sample

Six OSSF registered in the US were selected for our research:

1. The Apache Software Foundation (ASF, http://www.apache.org/) was incorporated in Delaware in 1999 to support the Apache HTTP server project. As of 2008, the ASF hosts 65 OSS projects, with 1765 committers. The ASF has a meritocracy based membership structure supporting a large and mature open source community. Its organizational processes serve as examples for other OSSF and it has earned a reputation for incubating OSS projects.

2. IBM released Eclipse code as open source and formed the Eclipse consortium in 2001. In 2004, the Eclipse consortium was reorganized into the Eclipse Foundation (http://www.eclipse.org), a 501(c)(6) non-profit. In 2008, the Eclipse Foundation hosted 11 top-level projects with 21 strategic members, 179 organizational members, and 942 committers. Eclipse is dominant in the market for Java Integrated Development Environments (IDE). Unlike most other OSSF, the Eclipse Foundation supports its members' business interests with a large and established community sponsored and controlled by companies. It is a known example of the commercial open source phenomenon known as OSS 2.0.

3. The GNOME Foundation (http://www.gnome.org/) was founded in 1997 to support the GNOME project and related projects which provide a desktop environment and development platform. The GNOME desktop environment is one of the dominant desktop environments for Linux and the GNOME Foundation has a large and established community.

4. The Plone Foundation (http://plone.org/foundation) was founded in 2004 to support the development of Plone, an open source content management system (CMS). Plone has a rather small share of the CMS market, but it is deemed as one of the best in the market. The Plone Foundation has a meritocracy based membership structure with many small open source consultants and service vendors within the Plone community.

5. The Python Software Foundation (http://www.python.org/psf/) was founded in 2001 as an organization devoted to the Python programming language, one of the leading script programming languages.

6. Software in the Public Interest (SPI, http://www.spi-inc.org/) was formed by members of the Debian project in 1997. Its mission is to "help organizations develop and distribute open hardware and software". It hosts several open source projects, such as Debian, freedesktop.org, and openOffice.org. GNOME was a subproject hosted by SPI before the GNOME Foundation was founded. SPI seldom intervenes in the affairs of its member projects, but it does hold their common assets.

Governance

Governance refers to the overall processes and structures used to direct and manage an organization's operations and activities. Three aspects of governance were the focus of our research: i) governance structure; ii) activities of the Board of Directors (BOD); and iii) occupations of the members of the BOD.

Most of the sample OSSF are membership based NPOs with the elected BOD bearing the major responsibilities for the organization. Members of the OSSF can be either a merit member or a sponsor member. An individual becomes a merit member after being recognized as making non-trivial contributions to the foundation. A sponsor member is a company that donates resources, such as money and developers, and in return is admitted as a member of the foundation. Typically, individuals are merit members, organizations are sponsor members, and employees of organizations represent sponsor members.

Based on "the extent to which the sponsor members can participate in the decision making of OSSF", we found three types of OSSF governance structures:

1. **Merit:** all members are merit members with full voting rights.

2. **Merit dominated:** merit members are in a majority and it is difficult for sponsor members as a group to affect outcomes.

3. **Sponsor dominated:** sponsor members are in a majority and may be classified into tiers where the size of the payment determines tier membership. Merit members typically work for a company or research centre.

The ASF, SPI, and the Plone Foundation are merit type foundations. The GNOME Foundation and the Python Software Foundation are Merit dominated. The Eclipse Foundation is sponsor dominated.

Board Activities and Occupations

Twelve categories of governance activities were defined as being carried out by the BODs of OSSF. The twelve activity categories are: i) strategic planning and common vision development; ii) development of policy and guidelines; iii) project governance; iv) financial governance; v) primary resources governance; vi) human resources governance; vii) fund-raising; viii) external relation management; ix) BOD self development; x) governance structure management; xi) community development; and xii) conference governance.

All six BODs in the sample allocated more than 60% of their efforts carrying out activities that fall into the following four categories: i) project governance; ii) development of policy and guidelines; iii) external relations; and iv) governance structure management. All six BODs allocated the least amount of effort to carrying out the following three activity categories: i) strategic planning and common vision development; ii) human resources governance; and iii) BOD self development. Effort was calculated by counting BOD activities per month in that category within the research period. Percentage of activities is the ratio of activities per month in that category to total number of activities per month. That the BODs in the sample did not spend more effort raising funds was a surprise. The BOD of one foundation did not undertake any activities to raise funds; only 3% of another's activities were carried out to raise funds. One reason may be that the virtual nature of OSSF does not require large operational budgets.

Among the six OSSF in the sample, the GNOME BOD is the most active as it carries out more activities per month than the other OSSF, while the BOD for SPI is the least active.

We believe that the occupation and status of BOD members affects the BOD's ability to carry out tasks as well as its behaviour. Eclipse and Plone have a greater proportion of BOD members with a background in management, business development, and strategic management, while the proportion of engineers and scientists is small. More than 50% of the BOD of the Plone and Eclipse Foundations are part of their companies' top management teams.

SPI has the largest proportion of BOD members with an engineering or scientific background (67%) and a small proportion of BOD members with a background in management, business development, and strategic management (11%). SPI and ASF have the largest proportion of OSS developers in their BOD, while Plone and Eclipse have none on their BOD.

Company Involvement in Governance

Company involvement refers to when the company invests resources in an OSSF and influences the mission, primary organizational activities, and relationships

of the OSSF. A company may directly influence the decision-making processes of an OSSF by having its employees hold seats in the BOD and other governance groups of the OSSF. A company may also indirectly influence decisions made by the members of an OSSF by providing resources to affect a decision. All six OSSF in the sample receive monetary donations or membership fees from companies. All six BOD in the sample have employees from companies that have business interests in an OSSF's projects.

OSSF Effectiveness

OSSF effectiveness refers to the ability of a foundation to utilize resources to achieve its goals and missions. Effectiveness metrics can be categorized as: i) project effectiveness; ii) community effectiveness; iii) resource acquisition effectiveness; and iv) stakeholders' assessment. From our observations of the information on the OSSF in the sample, we provide the following propositions:

Proposition 1: the proportion of company employees in the BOD positively affects the power a committee of the BOD has over timing and the content of roadmap software releases.

Proposition 2: the proportion of company employees in the BOD is negatively related to the proportion of BOD members who have a reputation in the OSS community.

Proposition 3: the proportion of company employees in the BOD is negatively related to the proportion of engineers and scientists in the BOD.

Proposition 4: the extent to which sponsor members can participate in decision making positively affects the proportion of employees in the BOD from companies that do not depend on the OSSF's projects for significant revenue.

Proposition 5: the proportion of members of the BOD from companies that do not depend on the OSSF's projects for significant revenue positively affect the proportion of BOD effort dedicated to strategic planning and common vision development.

Proposition 6-a: the extent to which sponsor members can participate in decision making positively affects OSSF revenue.

Proposition 6-b: the proportion of employees in the BOD from companies that do not depend on OSSF's projects for significant revenue positively affects OSSF revenue.

Observations

It was discovered that company BOD involvement is comprised of two dimensions: i) company employees and ii) employees of companies that do not depend on the OSSF's projects for significant revenue.

OSSF governance is comprised of five dimensions. The first dimension focuses on how much effort the BOD spends on strategic planning and common vision development. The second and third dimensions relate to the composition of the BOD: i) members who have a reputation in the OSS community and ii) members with a background in engineering and science.

The fourth dimension deals with power, specifically the power the BOD has over timing and content of the roadmap and software releases and the power of sponsor members. Finally, the fifth dimension focuses on sponsor members' ability to participate in decision-making.

Revenue could be used to assess OSSF effectiveness. Two dimensions, one that relates to company involvement and one that relates to governance, positively affect revenue. The importance of including BOD members from companies that do not depend on the OSSF for significant revenue should be highlighted. Their presence increases the effort dedicated to strategic planning and common vision development as well as the OSSF's revenue. Governance structure, or the extent to which sponsor members can participate in decision making, affects company involvement and OSSF revenue. This discovery is important for designing an OSSF.

Revenue for a 501(c)(3) OSSF is rarely more than $150,000 a year over the long run. OSSF with sponsor dominated governance structures tend to register as 501(c)(6) non-profits, while OSSF with merit and merit dominated structures register as 501(c)(3) non-profits. Since donations to 501(C)(6) OSSF are not tax deductible, these OSSF must attract revenue from membership fees to survive. OSSF registered as 501(c)(3) can issue tax receipts, must receive at least one third of their support from the general public, and membership fees and revenues from OSS conferences are not deemed as public support. Large donations from BOD members, or repeatedly going back to the same donors for income may not count as public support. In the sampled 501(c)(3) OSSF, long-term public support is generally below US $50,000.

Recommend Reading

Comparing Motivations of Individual Programmers and Firms
http://opensource.mit.edu/papers/bnaccorsirossimotivationlong.pdf

Free/Libre and Open Source Software: Survey and Study
http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf

Both "policy governing BOD" and "working BOD" exist in the sample. A BOD may delegate most of the managerial and operational activities to a full-time Executive Director or Chief Executive Officer and specific councils. This working style is considered a policy governing BOD where a full-time management organization in the OSSF is the enabler. The BOD of the Eclipse Foundation is an example of a policy governing BOD. In a working BOD style, BOD members, volunteers and staff members carry out the work of the BOD. The other OSSF in the sample belong to this working style.

Conclusion

Key findings in the research reported in this article contribute to the existing literature on open source and non-profits. Our findings suggest practitioners of OSSF need to select a proper governance structure and corresponding strategies to achieve their goals.

OSSF are powerful institutional tools but are still not well understood by academics and practitioners. The research reported in this article is one of the few to address company involvement, governance, and effectiveness in OSSF. Future research can test the propositions proposed in this research, utilize subjective measurements to assess OSSF effectiveness, or evaluate companies' indirect influence on OSSF not addressed in the research reported.

Zhensheng Xie is a software developer at EA Mobile in Montreal. He worked in the telecommunication industry for six years in China, with experience in software development, system design, and product management. He received his master's degree in the Technology Innovation Management program of Carleton University in 2008.

"We intend to integrate free, positive changes from whatever sources will provide them, providing they are well thought-out and increase the usability of the system...Above all, we hope to create a stable and accessible system, and to be responsive to the needs and desires of NetBSD users, because it is for and because of them that NetBSD exists."

Announcement of first release of NetBSD
ftp://ftp.netbsd.org/pub/NetBSD/misc/release/NetBSD/NetBSD-0.8

The NetBSD Project (http://www.netbsd.org) is one of the oldest modern open source software projects. It provides an operating system that runs on over 50 hardware architectures (also called ports), including the IBM PC, Motorola PowerPC, and Sun UltraSPARC machines. Founded in May of 1993, the project has supported the operating system's active development and managed contributions from thousands of individuals.

Prior to the New York City BSD Users Group Conference held in October, 2008, NetBSD developers from across the globe held a face to face meeting for planning and problem solving. Four developers from Sweden, Canada, the US, and Slovakia took a few minutes to think about how the NetBSD community has evolved over the past fifteen years.

This article summarizes those perspectives and provides insight into how an open source community maintains development momentum while managing contributions from a large number of volunteers with varying skill levels from across the globe.

Anders Magnusson

Anders is the port maintainer for the port of NetBSD to DEC VAX computers. NetBSD/vax was the first free operating system that ran on the VAX series of computers, and by far runs on the largest number of models of VAX. Within the NetBSD project, there is one port maintainer or a team of two co-maintainers per type of machine that the NetBSD operating system supports. A port's maintainer is the final authority about changes being made to that port, and is responsible for attempting to fix problems with the port and for integrating improvements into the port in a timely manner.

According to Anders, the NetBSD project has always had a focus on the developer. You might say 'by developers, for developers'. Many of the people involved in the project, especially early on, were hardcore kernel hackers. That trend may be for good or for bad. It does have the advantage of attracting people who are very committed to the project.

Since other developers tend to share common interests, discussions are simplified as everyone involved is likely to use a similar set of criteria for evaluating the available options. This kind of like-minded community has the advantage of being very attractive to people with a similar mindset. The disadvantage is that the project may not attract enough people capable of diversifying the skill sets and viewpoints within the project. As an example of the lack of diversity, there have been very few members of the project

who were interested in spending time promoting NetBSD, until several years after its first release.

NetBSD has always had a focus on clean, well designed, understandable code. This focus was inherent in the codebase, which started with the public releases of the Unix operating system developed by the Computer Science Research Group (CSRG, http://en.wikipedia.org/wiki/CSRG) at the University of California at Berkeley, as well as 386BSD (http://en.wikipedia.org/wiki/386BSD) which was developed by Bill and Lynne Jolitz. The BSD codebase embodied principles from academia such as good choices of algorithms and work that had undergone significant peer review. The quality of the work came from the dedication of the CSRG team, working on computers with limited resources, finding solutions to computing problems that had not been tackled before, and building maintainable, readable source code as a team. As with the developer mindset, having a common desire to maintain and enhance attributes in the code like quality, readability, comprehensibility, portability (to different compilers and target platforms) means that you will attract people who are interested in the same things.

Although it's pretty common these days for new open source projects to incorporate a non-profit foundation with a formal Board of Directors early on, NetBSD did not create its Foundation (http://www.netbsd.org/foundation/) until around 2000. Until then, the project was driven primarily by the technical group called 'core'. The core membership changes over time, but it is generally a group of about five developers who are considered to have a broad range of knowledge and good skills in design and foresight. The core developers are still part-time volunteers, who spend at least ten hours a week on NetBSD.

Having the Board in place has allowed things to get done that couldn't have happened before, but it seems that not having one from the start wasn't detrimental to the NetBSD community.

The project had its early successes, such as having a fully 64-bit clean operating system as early as 1994. It has also learned from challenges, such as balancing the demand for new hardware support with the desire to implement code in a well designed, rather than fast and sloppy fashion. Failing to do this could have cost the project its reputation for clean code.

Having a concurrent versions system (CVS) for a project's developers to share, then later open for the public, may seem commonplace today. In 1993, having that commitment meant educating developers about version control systems and figuring out CVS's quirks. It means that NetBSD has a history of every code change since its inception, which many projects from that era do not.

NetBSD has a mix of younger and older developers, providing a broad range of experience. There is a mix of new ideas and people who have seen new ideas fail and who are able to understand what has changed as what was tried before and didn't work, may well work today. By older, I mean we have a number of developers in their 50s and 60s. Younger developers are attracted to the opportunity of learning from the experience of the older developers.

David Maxwell

David has been developing software for over twenty years and is a former security officer for the NetBSD Project. He states that:

My first exposure to NetBSD was at a new job, with a system running NetBSD 0.8 in

production use. I already had a background in Unix that went back almost a decade, so NetBSD had to live up to my expectations for reliability and consistency in a Unix system.

The atmosphere on the NetBSD mailing lists was pleasant, and the signal to noise ratio of the discussions was very high. The primary developers were clearly knowledgeable people from whom I could learn, and that was very appealing to me.

From a technical point of view, the source code for the system was simply amazing. I saw how I could learn from the examples of so many good programming techniques which included layering, reuse, clean code, up to date documentation, and version control. A lot of these methods came from the work of developers at Berkeley, pre-dating NetBSD. The codebase was well tested, time-honed code. The ideas being implemented weren't just theoretical, they had already been in production use for years.

I saw new features being added by the project, showing innovative solutions to problems that I didn't yet have the background to anticipate. Even if I managed to learn everything I could from the source code, there were still new lessons being created on an ongoing basis.

Centralized version control and development of both kernel and userland code taught me about real-world design and maintenance of interfaces between code. The construction of a source tree that builds cleanly for many different platforms also impressed me. I had seen userland code that used configure scripts to adapt to the differences between Unix systems, but here was kernel code that adapted to the differences between CPUs and memory architectures. Even today, after fifteen years of using NetBSD, there are still new things for me to learn.

As previously mentioned, the atmosphere within the project was very good right from the beginning. This was important because changing culture is always hard. Creating an environment that is right in the first place helps, but you can't always be certain what's right in the first place as the community is always a learning. NetBSD made many good choices at the beginning, but some percentage of that was simply good luck.

One thing that amuses me regularly is the number of people who have the impression that NetBSD is stagnant, or dying. Those people obviously aren't subscribed to the source-changes mailing list, or they would see new code being committed every day. NetBSD continues to attract new developers and users, and continues to innovate. The NetBSD community-building story isn't finished yet.

Lubomir Sedlacik

Lubomir has been a NetBSD developer since 2002, a former member of the pkgsrc (http://www.pkgsrc.org/) security team, and a current member of the pkgsrc release engineering team. pkgsrc is a framework for building third-party software on NetBSD and other UNIX-like systems. Lubomir speaks:

I started participating in NetBSD after being frustrated by other project communities where some users constantly ask questions but never make an effort to understand the answers.

When I first found NetBSD, I developed an appreciation for NetBSD as a bloat-free, cleanly coded, complete operating system, and started using it on my laptop. Some of the criteria Anders mentioned as goals clearly were a draw for me to use, and then participate in, the project.

NetBSD was better thought out and designed when compared to other operating systems I had tried. I saw a high concentration of smart people with lots of experience working on the system and actively participating on the mailing lists. While the rate of code contribution by an individual may vary over time, many of those same smart and experienced people still participate in the project by offering feedback and advice. The people who remain tend to be those who don't have trouble learning for themselves. I was drawn to the system because of the easy to understand, well designed code. I particularly remember the design paper for the rc.d system (which does initialization at boot time), and how well thought out it was.

Individuals need to realize that when you don't just give up on trying to understand and solve a problem, the results can be amazing. If you concentrate and try to figure things out and work through a problem, you'll find that once you develop this attitude everything is easy. The NetBSD community has done a good job of inspiring new developers to build this mindset. While it may sometimes frustrate those who don't make the leap, the project definitely benefits from having a very can-do attitude where individuals share responsibility for the quality and functionality of the code that the NetBSD project distributes.

As a result of that attitude of responsibility, NetBSD has had very little breakage due to API (application programmer interface) changes, and has learned from the breaks that did occur. Even when API changes are needed, the project provides shim layers that reimplement the old API using the new API. These layers mean that old programs continue to work without having to be changed. At the system level, administrators can leave the shims out if they are not needed, to forgo the additional code bloat that would otherwise build up with many layers of backwards compatibility. A similar compatibility mechanism allows NetBSD to run binaries that were compiled for other operating systems which run on the same CPU. For this reason, NetBSD can run Linux, FreeBSD, and Solaris binaries. Taking responsibility for API and ABI (application binary interface) continuity means binaries built for NetBSD 0.8 (the very first release) can still be run on NetBSD 4 (the most recent release). That kind of good design and superior technical leadership in planning and execution attracts technical people who have an appreciation for the aesthetics of a system.

Jeremy C. Reed

Jeremy is a NetBSD and pkgsrc developer. He writes:

I started using NetBSD around 1999 when I wanted a system for running web services. At work, I was already familiar with BSDI (http://en.wikipedia.org/wiki/BSD/OS), a commercial version of BSD developed by a company founded by former CSRG members. I liked that NetBSD was built to run on many different platforms and that the pkgsrc packaging system could also be used on different operating systems, including Linux, FreeBSD, Solaris, and AIX. This kind of layering and reusable architecture is very attractive to technical people.

I found the NetBSD community to be friendly, helpful, and knowledgeable. In many cases, other communities are either helpful but not knowledgeable, or are knowledgeable but socially abusive to newcomers. The social tone that the NetBSD project maintains on its public mailing lists and forums includes respect, open discussion, and the willingness to disagree. Not counting spam filtering, the only moderated mailing lists are the low volume, official project announcement lists.

I started by filing problem reports for issues I encountered and by submitting potential enhancements to the system; soon I was invited to become a developer. This process of participation and contributing which then leads to membership and CVS commit privileges in the project instills commitment, responsibility, and pride in one's work. For the NetBSD project, that results in high quality that attracts more developers on an ongoing basis.

I appreciated that NetBSD had different technical mailing lists where I could follow what interested me, without trying to track every proposal that was going on system-wide. To a large extent, the NetBSD Project has avoided moving to 'Enterprise 2.0' forums, preferring to document the system in version controlled text files in the source tree, or version controlled HTML documents. With a source tree as large as NetBSD, external discussions would be difficult to match up to the version it relates to and would incur additional work to prune or update periodically, whereas CVS content can only become outdated, not disconnected. The topic-specific lists are also useful for targetting questions and proposals.

Once in the project, I saw that there was plenty of internal discussion and that much occurs behind the scenes in the form of mentorship and design work. Lately, we've pushed for more of that to be done in public forums, so the learning benefit is available to a wider audience. While it's not always practical to open a discussion up to all comers, we have accomplished better transparency in our efforts. Having the problem reporting and tracking system in place from day one was very important. They provide a social history of discussions, why particular decisions were made, and why some solutions can't be implemented until a blocking item is fixed first.

Having public code, CVS, and CVSWeb (http://cvsweb.netbsd.org) have been valuable in making it easy to review the code, and to keeping low barriers for new users to see, learn, and experiment. The wiki and IRC presence of the NetBSD Project are examples of communities that were not set up by the project, but have sprung up around it, and are mutually beneficial. Interactive text chat tools allow developers and users to interact in real-time and allow faster dissemination of information and the exchange of ideas. Having a project chat-room also provides a way to introduce new developers to the social rules that are implicit in any group effort.

The NetBSD website has for a long time listed users and providers of NetBSD-based products. The 'NetBSD In Action' gallery (http://www.netbsd.org/gallery/in-Action/) lets people see how other members of the community are using NetBSD.

NetBSD welcomed education and research applications early on, which resulted in contributions of useful subsystems such as RAIDframe (http://www.netbsd.org/docs/guide/en/chap-rf.html) which reinforced the habit of code based on thorough analysis and has been used as reference material for teaching good programming habits.

Summary

Whether by design or by chance, NetBSD has many attributes that built a strong, vibrant community. We present the following as useful as tips to people starting new projects.

- know your audience, both committers and users, and cater to it
- have a central focus on something that is of high quality, and has attributes that make it unique

- have some form of leadership, but it doesn't have to be overly sophisticated
- learn from your mistakes
- be transparent
- have a broad membership that encourages a variety of perspectives that share common goals
- offer value to your audience, such as learning opportunities
- be friendly
- encourage independence, rather than dependence
- reward good work
- have good communication tools and practices which allow for the archiving of discussions

David Maxwell is Coverity's Open Source Strategist. An open source security specialist, he has over 20 years of experience as an open source user and developer, and is particularly active in the NetBSD community. He currently sits on the advisory board for the BSD Certification Group and the program committee for the annual BSDCan conference. He was NetBSD Security Officer from 2001-2005 and a contributor to the O'Reilly title "BSD Hacks." Maxwell has previously worked as a lead kernel developer for Nokia, and architected the Internet Service offering for Fundy Cable in New Brunswick.

Lubomir Sedlacik is a software engineer at Sun Microsystems by day and pkgsrc hacker by night. He helped to establish the pkgsrc security and release engineering teams and spent countless hours working on Solaris support in pkgsrc. He is also one of the organizers of the annual pkgsrc conference, pkgsrcCon.

"You can observe a lot by just watching."
Yogi Berra
http://en.wikipedia.org/wiki/Yogi_berra

This article advances the thesis that "commons sourcing" is the emerging third wave of commercial transformation. It begins with the iCommons concept and its origin in open source software (OSS) methodologies and emergence in other business models. It then defines commons sourcing and situates it with respect to the two earlier waves of commercial transformation. It concludes with some reflections by a commons sourcing lawyer.

Tragedy of Commons vs Treasury of iCommons

The "Tragedy of the Commons" (http://en.wikipedia.org/wiki/Tragedy_of_the_commons) is a leading social science paradigm largely based on a Garret Hardin's 1968 Science article by that name. Hardin postulates a commons pasture that is open to all and examines the position of a commons herdsman who is deciding whether to add an animal to his herd. Hardin applies the following utility model:

- if the herdsman adds the animal, he will gain the full benefit from the sale of this animal (+1)
- while there will be a cost to the commons due to additional overgrazing (-1), the effects of overgrazing by this additional animal are shared by all of the herdsmen (n) and this herdsman will only bear a fraction of that cost (-1 / n)
- after calculating the net utilities, the rational herdsman concludes that the only sensible course is to add the animal

As a result, the herdsman adds the animal and then continues to add animals.

In parallel, the same conclusion is reached and the same course of action is taken by every other herdsman that is sharing the commons pasture. In the words of Hardin: "Therein is the tragedy. Each man is locked into a system that compels him to increase his herd without limit – in a world that is limited. Ruin is the destination toward which all men rush, each pursuing his own best interest in a society that believes in the freedom of the commons. Freedom in a commons brings ruin to all."

This article is focused on an iCommons: a shared pool of information which, unlike a physical commons, is not subject to the physical constraints of scarcity. By removing scarcity from the Hardin paradigm, the cost and benefit equation of the resulting utility model is fundamentally altered. Since the information is not consumed as it is shared and used, the "tragedy of the commons" now becomes the "treasury of the iCommons".

Commons Sourcing

The OSS movement is the earliest and best example of an operating iCommons. While it is daunting, if not impossible, to distill the diverse range of OSS activities into a single methodology, the following principles are central to the OSS movement:

- advantages are achieved when software development is done as a collaborative activity
- optimal collaboration requires the sharing of source code across broad communities of interest
- this sharing must be done in a manner that promotes and preserves certain fundamental freedoms
- these freedoms are reflected in two major OSS software licensing models: permissive and reciprocal

We focus on the broader iCommons phenomenon; in particular, the iCommons pooling of shared information resources under OSS methodologies for use by companies for commercial purposes. This author has coined the term "commons sourcing" for this phenomenon. As set out below, the commons sourcing approach is increasingly being found at the core of new commercial initiatives.

Commons sourcing is the emerging third wave of commercial transformation. The first wave of commercial activity in the industrial revolution can be best characterized as in-sourcing since companies of necessity focused on doing everything themselves in vertically integrated business structures in a world of tariff protected nation states.

With the emergence of globalization and trade liberalization, out-sourcing emerged as the second phase of commercial transformation. In this era, companies became very focused on core competencies and looked for opportunities to out-source other business functions. Many of these opportunities were driven by the availability of skilled work forces in developing countries at extremely attractive labour rates. These labour rate variations, which were largely a legacy of the formerly tariff-protected nation states, initially drove strong business cases based on market arbitrage. However, these cases have generally become much less compelling as labour rate gaps narrow and as the total costs of out-sourcing become increasingly apparent to companies.

Companies are now looking at new business transformation opportunities and commons sourcing models are being increasingly adopted.

Unlike the 1:1 limitations inherent in outsourcing models, commons sourcing affords companies the opportunity to leverage the n:1 ratio. This n:1 ratio allows companies to achieve order of magnitude efficiencies in the cost structures of some business input without many of the attendant business costs of outsourcing. Applying the prior tragedy of the commons utility model to commons sourcing:

• if an iHerdsperson adds valuable information to the iCommons, all iHerdspeople gain benefit (+1 x n)

• since contributed information is not subject to scarcity constraints, the average cost to each user of the iCommons is represented by her fractional share of the overall contribution cost (-1 / n)

• the rational iHerdsperson concludes that contributing to the iCommmons is a sensible course of action

While this recasting of the tragedy of the commons paradigm is neither comprehensive, as it fails to take market conditions into account, nor precise, as it fails to probe relative contribution value or taking without giving, it does provide a simple illustration of the impact of removing scarcity from the equation.

Reflections

This author has the following reflections on the commons sourcing model:

1. Think abundance. Commons sourcing is about abundance, not scarcity, and provides a number of collaborative models for value creation.

2. But keep enough scarcity. Apart from some pure play opportunities, most companies need to maintain some commercial differentiators and remain separate from the iCommons pool.

3. Remember Goldilocks. The core challenge under a commons sourcing business model is getting it "just right". A company that commons sources too much will end up lacking the differentiators that are key to its commercial success. Perhaps less obviously, if a company commons sources too little, it will most likely end up facing uncompetitive cost structures for any inputs which its competition is commons sourcing.

4. Reset the risk paradigm. Commons sourcing activity requires a reset of the risk paradigm to properly weigh the risks involved in the activity against the risks of not being so active.

5. Allow for some optimism. The commons sourcing movement and its communities are based on a strong ethos and value system.

6. Embrace the Gordian knot. Since commons sourcing exists at the intersection of very complex business, technical and legal issues, the commons sourcing lawyer must be ready, willing, and able to fully commit to a technical and business deep dive while addressing legal issues.

This article is based on a February 13, 2008 presesentation to The Ottawa Network.

Thomas Prowse is a Partner with the Gowlings Kanata Technology Law Office. His practice focuses on providing legal advice in the areas of technology law and technology-related commercial matters. Before re-joining Gowlings, Thomas was Senior Counsel with Nortel, a leading Canadian technology company with global sales and operations. Thomas provided general legal support to numerous and diverse product development organizations. Thomas worked extensively on OSS matters during his tenure at Nortel and was the Global Law Department leader on the Nortel Open Source Advisory Team.

"It is important to utilize the knowledge revolution to join our people and continents with the help of the government, universities and technology centers. Our countries need technology centers with flexible structures to take advantage of the knowledge of the society using tools such as Information and Communication Technologies and Open Source Software to succeed in the information era."

Luis Millan Vásquez de Miguel,
President of FUNDECYT
http://www.fundecyt.es/

The International Center for Technological Development and Open Source Software (CIDETYS) in Panama is a non-profit organization, promoted by the Panamanian government to harness the potential of Information and Communication Technologies (ICT) and open source software (OSS) to create social benefits for its population. An important goal of the center is to collaborate with international organizations and become a leader in Central America in the development, use and implementation of ICT and OSS.

This article describes the main activities that CIDETYS will be focusing on during the first years of operations, how this initiative was born, and a story of the implementation of OSS in the region of Extremadura, Spain that inspired the creation of CIDETYS.

Benefits and Challenges

ICT and OSS have the potential to create social benefits in developing countries, where technological development is still inadequate. CIDETYS was created in Panama on March 2008 as a public interest association to promote the use and benefits of ICT and OSS in Panama and collaborate on projects with companies, universities, technological centers and governmental agencies.

The main objectives of the center are to: i) foster collaboration between universities, technology centers, companies and the government on projects related to ICT and OSS; and ii) collaborate and assess the government on investments related to ICT and OSS.

The initial activities of CIDETYS are three:

1. Create a program of technology literacy. This program is inspired on the success of a technology literacy project implemented in Extremadura, Spain in the year 2002. The goal of Extremadura was to provide a population of 200,000 grade one to twelve students with desktop computers. However, when implementing this program, the government of Extremadura realized that the software licensing costs were too high in comparison with the budget available. This is why the Linux based operating system gnuLinEx was created and installed on more than 100,000 desktop computers in public funded schools in the region.

The goal of CIDETYS is to identify potential implementations of OSS in the public education system as well as in publicly funded Technology Community Centers in the country. Technology Community Centers are facilities equipped with desktop computers and Internet access that aim to bridge the digital divide in the less developed regions of the country. There are currently more than 75 centers throughout the country that have serviced more than 158,000 community members since 2005.

2. Create a small and medium technology enterprise incubator. The technology enterprise incubator aims to help entrepreneurs create technology based companies in Panama. It offers facilities equipped with technology and experts to help entrepreneurs thrive during the initial steps of creating a company.

The technology includes computers with Internet and business software, telephone and fax. The experts include a lawyer, an economist and university professors with entrepreneurial knowledge that can offer mentoring and advice. The goal is to offer the services of mentoring and use of the facilities to any individual interested in initiating a business.

3. Organize workshops and conferences about grid computing (http://en.wikipedia.org/wiki/Grid_computing) and telecommunication networks. Panama is considered a hub in Central America because of its geographic position and connectivity. According to the portal Business Panama: "Panama has four submarine international connections via high-bandwidth, fiber-optical trunk routes. This in turn is already enabling fast and reliable connections for banking, e-commerce and other businesses as well as additional high-speed consumer activity units" (http://www.businesspanama.com/investing/opportunities/ecommerce.php).

CIDETYS is partnering with the EELA-2 project (http://www.eu-eela.eu/), a European initiative that aims at building a high capacity, production-quality, scalable grid facility. CIDETYS is planning to organize conferences and workshops to promote grid computing and its benefits, to share experiences with Panamanian researchers in this area, and to encourage collaboration in specific areas. Experts of the EELA-2 project have confirmed their support to this initiative.

About the initial activities programmed by CIDETYS, the President of the Technological University of Panama, Marcela Paredes de Vasquez says: "The opportunities to create projects that bring social benefits are endless. CIDETYS is a noble initiative that has strong support from the government, universities and technology centers in Panama and international technology centers such as FUNDECYT in Spain. There is an opportunity to bring together other initiatives around ICT's and OSS from technology centers, companies and independent organizations in Panama, that can find a common ground and collaborate through CIDETYS".

However, the challenges are numerous. According to the Strategic Plan for the Development of Science, Technology and Innovation in Panama, prepared by the National Secretary of Science, Innovation and Technology in 2005, Panama has a low number of researchers, partially because there is little government investment in research and development (R&D). Panama's investment in R&D as a percentage of the Gross Domestic Product (GDP) ranks below the Latin American average. In addition, there are few people within the working population with master's or doctoral degrees. From a population of approximately 1.1 million, only 0.5% have master's or doctoral degrees, and from this percentage, only an average of 9% is enrolled in activities related to science and technology.

Despite this situation, the ICT sector has grown steadily during the last decade, partly due to foreign investments in telecommunications. Panama's economy is service based, with 82.2% of the GDP represented by the service sector. Panama's policy on ICT is to develop competencies in the service sector because this is the main area for the application of ICT. Given the impact of ICT on worldwide economy growth since the end of the last century, ICT are considered a determinant factor in the capacity of Panama's economy to compete worldwide.

Early Stages

In 2006, the City of Knowledge, a governmental agency, received a group lead by

Dr. Luis Millan Vasquez de Miguel, president of the Foundation for Science Development in Extremadura (FUNDECYT). During this visit, Dr. Vasquez de Miguel presented a conference about the benefits of the implementation of ICT and OSS in Extremadura and the development of the gnuLinEx operating system to be used in Extremadura´s schools. Different groups interested in OSS in Panama, including non-profit organizations such as the Software Libre Fraternity and governmental agencies such as the National Secretary of Innovation, attended this conference. All these groups were convinced of the need to join efforts to formally collaborate and promote the use of ICT and OSS in Panama. FUNDECYT offered to collaborate with the know-how to make this center a reality, based on their experience in Extremadura.

During the next two years the process of defining and creating CIDETYS followed, led by the City of Knowledge. The Technological University of Panama (UTP) was called upon to host the project. The president of UTP, Marcela Paredes de Vasquez, said: "The UTP is pleased to accept the presidency of the board of directors of CIDETYS. CIDETYS is an excellent initiative to join the efforts of different groups inside and outside the UTP focused on ICT´s, OSS and other open architectures. These areas are considered strategic by the university because of the potential to position Panama as a competitive nation in the area of technology innovation". CIDETYS is planning to start operations on October 2008 and will operate within the City of Knowledge of Panama.

A Story of Success

The implementation of gnuLinEx in Extremadura, Spain was successful and highly publicized (http://www.bios.org.bd/os_advocacy/spain.php).

In 2004, the European Commission chose gnuLinEx as one of the projects that best promotes a knowledge society while encouraging regional development.

Extremadura achieved a high degree of technological advancement through an aggressive plan that included two main axes: i) broadband Internet connection of all the cities and towns of Extremadura, covering a region of 41,000 km2 and 1 million people; and ii) technology literacy.

The results of this plan included:

1. Extremadura became the Spanish region with the highest rate of publicly available computers per person, with one publicly available computer with broadband Internet connection for every nine people.

2. 100% of the primary and secondary schools have desktop computers (1 computer for every two students) and they are all interconnected through a regional network.

3. More than 80% of the population above 16 years old has participated in one or more events sponsored by the plan.

Conclusion

CIDETYS is a Panamanian initiative to harness the potential of ICT and OSS to benefit the population and increase the competitiveness of the country. It is starting operations in October 2008 and one of its main goals is to collaborate with international technology centers to share experiences and knowledge. If you are interested in participating and knowing more about our work, please contact Monica Mora (monica.mora@utp.ac.pa) or Lydia de Toppin (lydia.holnes@utp.ac.pa) at the Technological University of Panama.

Monica Mora received a Master´s degree in Technology and Innovation Management from Carleton University. She has worked for the Technological University of Panama in different positions, including assistant professor and assistant of the President of this university. She is currently part of the technical committee of CIDETYS which was created to advise the Board of Directors and plan the first activities of the programme.

Recommended Resources

Technology Community Centers in Panama (in Spanish)
http://www.infoplazas.org.pa/sobreInfoplazas/queEs/

Panama's Continental Fiber-Optic Network Link
http://www.businesspanama.com/investing/opportunities/ecommerce.php

Interview with Dr. Luis Millan Vasquez de Miguel (in Spanish)
http://www.consumer.es/web/es/tecnologia/software/2006/09/14/155393.php

"Literacy unlocks the door to learning throughout life, is essential to development and health, and opens the way for democratic participation and active citizenship."

Kofi Annan, former
Secretary-General of the UN

Literacy Bridge (http://literacybridge.org), a non-profit technology startup, is using open source software (OSS), open hardware, and open content to solve some of the world's most challenging problems: global poverty and disease. Through the development and application of a digital audio device, Literacy Bridge's Talking Book Project is designed to make access to information available and affordable to those who have the fewest resources but the greatest need. This article summarizes the Talking Book Project and describes how six aspects of successful open source projects are being applied to improve global literacy and access to information. Most importantly, this project demonstrates the power of combining community and appropriate technology to change the world.

Readers of this issue of the OSBR may not appreciate the ease at which they are able to acquire knowledge to improve their productivity. While one portion of the world takes for granted the electricity and literacy skills required to read publications like this one, another portion lacks these prerequisites, yet has an even stronger need for efficient access to information. Recognizing an opportunity to apply technology and open source principles to this inequity, Literacy Bridge, a non-profit technology startup, launched the Talking Book Project.

The Talking Book Device is a digital audio player/recorder designed for the 2.6 billion people living on less than $2 per day. Most of these people have minimal literacy skills and live in rural areas without electricity or Internet access.

Unlike a common iPod or most other MP3 players, its power source is not dependent on grid electricity, and its audio content distribution is not dependent upon computers. This device also distinguishes itself with its rugged design, variable-speed playback, internal microphone and speaker, and an easily programmable interface.

To understand how this audio player/recorder will reduce poverty and disease, one should consider the problems and opportunities of distributing information and building literacy skills in the poorest regions of the world.

Building Knowledge in the Developing World: Problems and Opportunities

The Talking Book Project approaches global illiteracy with a short-term and long-term view. For the short-term, the Talking Book Project provides access to crucial and locally relevant information in a form that does not require literacy. For the long-term, the project provides a literacy education tool so that text-based information will soon be accessible.

As a critical foundation for education, literacy may be the most important strategic investment to eradicate poverty. However, literacy should not be a prerequisite for the efficient dissemination of knowledge to fight disease and malnutrition - not when nearly one billion adults cannot read, including 40% of all adults in South Asia and Sub-Saharan Africa.

Access to Information: An Immediate Solution to an Urgent Need

In the poorest regions of the world, the most efficient means of disseminating knowledge is by pickup truck. Each day, thousands of nurses and health officers of governmental and non-governmental organizations climb into pickup trucks and ride out to remote villages. Upon arriving, they gather people together and explain how to reduce the spread of HIV/AIDS, how to treat a dangerously dehydrated infant, and numerous other life saving messages. Traveling over nearly impassable roads and paying for costly fuel and precious staff time, this method is costly and inefficient, but it is currently the only option.

The Talking Book Device multiplies the impact of existing poverty reduction programs, just as the Internet has multiplied productivity in the developed world. Local organizations support the project because it saves them time and money and allows their health and development messages to reach more people. It also improves the quality of face-to-face visits by allowing a focus on the key messages, leaving detailed audio notes for later reference.

As is true anywhere in the world, people with a visual disability have an especially big challenge accessing information. In the poorest regions of the world, the challenge is even greater. Braille is hard to find, leaving blind children with little hope of obtaining an adequate education.

The Talking Book Device is designed for universal accessibility as no feature requires sight. The embossed buttons are of various shapes and sizes, and the device is designed with a vertical asymmetry to allow one to feel its orientation the moment it is grabbed.

Literacy: The Foundation of Education

Most parents today know how important it is to read to their children at a young age, even before primary school. A child's lack of exposure to reading in these early years can lead to a significant educational disadvantage many years later. For most families in the developed world, building early literacy skills is simply a

matter of dedication. But for families without a literate parent, children are disadvantaged even before they begin school. The children who are able to attend school (70+ million children cannot, primarily due to school fees) compete for a teacher's attention, often with 50 or 60 other children in the same classroom. To address this teacher shortage, many governments desperately recruit youth with just nine years of primary and secondary education and no training as a teacher.

The Talking Book Device enables children and their parents to practice reading when a literate parent or educator is not available. When paired with a book or any other source of text, such as an alphabet written on a blackboard, the user can engage in active reading practice and even reading comprehension questions and other interactive exercises. Once an educator or member of the community has recorded a reading, the student can listen to the recording, control the speed of playback, choose to have particular words defined, and jump from page to page, line to line, or word to word.

Applying Open Source Principles to the Talking Book Project

The Talking Book Project utilizes six open source principles: i) user-driven adoption; ii) open development; iii) open source applications; iv) decentralization; v) release early and often; and vi) acknowledge your contributors. The benefits provided by these principles are discussed here.

User-Driven Adoption

Users drove the open source adoption phenomenon. Acquiring and using OSS did not require purchase orders, strategic planning, or executive approval. If the user saw value in OSS, he or she had the power to acquire and start using it.

Since OSS is more likely to fit into an existing heterogeneous infrastructure than proprietary alternatives, millions of users found it added immediate value to their environment.

Similarly, the Talking Book Device is designed to fit the context of its users at a cost that is within their power to purchase directly. Targeted at $5 to $10, depending on volume, the price of a Talking Book Device will compare with that of a radio, the most commonly owned electronic device in rural areas. Some governments and aid organizations may choose to subsidize the device for the very poorest families, but the device's technology choices are aimed at individual ownership.

The Talking Book Device is powered by the most common and least expensive form of available energy: disposable, D-size batteries, typically used in flashlights and radios. Without access to electricity, rechargeable batteries would require new infrastructure. Literacy Bridge is actively researching various options for affordable and renewable energy. To spur adoption, priority was given to fitting into the existing context, then transitioning to a new power solution.

Open source users are more likely to promote the adoption of the software simply because they can easily distribute it to their friends. Likewise, Talking Book users are critical to its content distribution system. Although electronic networks are rarely accessible to the poorest rural areas, "people networks" can be leveraged for the same job. Therefore, each Talking Book Device includes an integrated USB plug and receptacle so that users can give audio content to their friends at no cost by simply connecting the two devices. This allows the user community to make the system more valuable.

Open Development

Many thriving open source projects owe their success to the contributions of a broad base of developers. Source code is available for anyone to review, test, and patch, and the code tends to be more modular, allowing for concurrent development. This, in turn, allows developers of varying skill sets to participate in the project.

The Talking Book Project includes several software projects, one of which encompasses the functionality of the Talking Book Device. Built around a small microcontroller designed for audio processing, the core functionality is programmed in C and low-level assembly code. Testing or patching this level of software requires having a chipset or Talking Book Device at your computer. To broaden the potential developer support, Literacy Bridge did the following:

1. To expand developer support beyond C developers, most device functionality was moved to a declarative XML-driven layer. Changing system menus, content navigation, and volume control are all possible by editing a text file. Just as the early stages of the Web attracted thousands of new HTML programmers, this similar markup language opens the door to a broader base of device programmers.

2. To test and run these XML-based features that control the device's audio interface, a Flash application was developed to simulate the hardware and low-level software. A text file can be run on the device or the Flash application with identical results. This expands development and testing beyond people who own the right piece of hardware.

One of the objectives of the Talking Book Project is to instill ownership in the project throughout various entities in the host country. Part of the reason is to promote the long-term engineering sustainability of the project. This open development model invites social workers, budding programmers, as well as embedded C programmers to participate in the engineering and maintenance of the device.

Open Source Applications

Building a platform in any industry requires an understanding of the importance of application development and distribution. Open source platforms not only provide easy access to interested application developers, but they sometimes even host repositories of applications or plug-ins built on their platform.

Likewise, the Talking Book Project relies on audio content for success. While Literacy Bridge believes that audio content is best left to the experts and citizens who speak the local languages and understand the local problems, facilitating the creation and distribution of that content is just as critical as designing the platform on which it runs. This philosophy led to the following actions:

1. Every device includes a microphone, so that every user can potentially become a content creator.

2. Audio content includes a control track that uses the same format and has most of the same flexibility as the system interface described above. Audio content can include embedded hyperlinks from one segment of content to the next. Content-programmable buttons allow interactive and entertaining applications – a universally important driver of user adoption.

3. For people who would like to develop interactive audio content but prefer to avoid declarative programming, Literacy Bridge is developing a Windows application with a graphical user interface for creating and editing control track files.

Users of this application will include university students, employees of non-profit organizations, and district officers of the ministries of health, education, or agriculture.

4. Content that can be useful to multiple regions of the world will be hosted on a web site to allow local governmental and non-governmental organizations to select and download the recordings that they believe will be useful for their regions and domains.

Decentralization

Open source projects tend to be open to decentralized models of control. Open source licenses encourage creation of derivative works, improving the ease at which software enhancements can be made.

Following this principle, the Talking Book Project encourages each local implementation to experiment with what works for their communities, sharing feedback with other implementations. The low-cost and scalable nature of the project also makes it easier for implementations to spring up wherever the demand is greatest. This is particularly important with respect to Talking Book kiosks.

The Talking Book Project includes networked kiosks to improve content distribution and discovery. These kiosks might be considered a cross between Wikipedia and the iTunes Music Store. They serve as community centers for uploading and downloading knowledge recorded in audio, and they also help users discover the content that is most likely to interest them. Kiosks may host complementary businesses, such as support businesses or solar-powered stations for renting rechargeable batteries.

If Literacy Bridge distributed kiosks across one entire country at a time, the bias to pick countries based on overall literacy and poverty statistics would cause pockets of severe need to be missed in otherwise less severe countries. By developing the kiosk system without a top-down command and control structure, new implementations can be driven by a more granular assessment of need.

Decentralization also reduces the ability for any one central force to attempt to shut down or control the distribution of information, just as the inclusion of a microphone on every device decentralizes the power to produce content.

Release Early and Often

The most successful open source projects publish their work as frequently as possible. Projects that hold back a release until they believe they have solved every problem tend to be less productive than the ones who take lots of small steps, accepting feedback along the way.

It is much more expensive to perform each development iteration for a hardware project than for pure software, but it still saves money and improves quality in the long run. After less than a year of research and development, Literacy Bridge is preparing to produce 100 Talking Book Devices for field testing in West Africa, followed by another 100 devices tested in India (contingent on individual donations). This small volume of 200 units will cost Literacy Bridge approximately $35,000 -- a significant investment for a small non-profit. However, skipping these pilot trials and relying only on a few small focus groups would have bypassed invaluable feedback to improve the future devices for hundreds of millions of other users.

Acknowledge Your Contributors

People contribute to open source projects for a variety of reasons. Some developers are just "scratching an itch", some are guiding a product they want to use, but most appreciate any public acknowledgment of their contributions.

As a startup non-profit charity with a small budget, Literacy Bridge does not have a single paid employee, but it has benefited from over 5000 hours of volunteer work. These volunteers are acknowledged on our web site, in our newsletters, and during public events.

Although Literacy Bridge has no payroll expenses, it must pay for prototype production, pilot program costs, and outsourced engineering work. These expenses are entirely funded by individual donors. As with volunteers, donors are recognized (http://literacybridge.org/about/donors.html) for their willingness to step forward and have their donations invested to make access to knowledge available to people with the greatest need.

As is probably true for most contributors to Apache, Linux, and other notable open source projects, Literacy Bridge's 160+ volunteers and donors are not contributing for the public acknowledgement. They are contributing their time and money to be a part of something that they believe will change the world.

Applying Community and Technology to Change the World

This article has focused on two key areas: community and technology. The power of these two forces has been demonstrated throughout history, with the OSS phenomenon as one recent example. Literacy Bridge is simply applying the same concepts to fight global poverty and disease.

Throwing technology at a problem has often failed to produce results, but when technology is used as a multiplier of existing community efforts, significant and sustainable change can be accomplished. In the case of the Talking Book Project, Literacy Bridge is using a community of individual donors, developers, and other volunteers to produce technology that multiplies the efforts of other communities throughout the world – communities of teachers, nurses, agriculture experts, and others. Together, these communities are applying technology to bring an end to global poverty and disease.

You can be part of this new approach to help end global poverty and disease by volunteering or donating to Literacy Bridge. See http://literacybridge.org/volunteer for volunteer needs or go to http://literacybridge.org/donate by December 31st to learn how you can become a Founding Donor of Literacy Bridge.

Cliff Schmidt is the Executive Director of Literacy Bridge (http://literacybridge.org), a non-profit organization empowering children and adults with affordable tools for knowledge sharing and literacy learning. Prior to founding Literacy Bridge, Cliff ran a successful open source consulting business, specializing in intellectual property issues and community development. He has served both the Eclipse Foundation and The Apache Software Foundation, where he was elected as a board director and appointed Vice President of Legal Affairs.

"...the OSGi Alliance is about managing this overwhelming sea of components and their interaction. It is about being in control of the thousands of dependencies that software has to internal and external artifacts. It is a framework that addresses the heart of the software development process. It is about managing the software development, deployment and the, most expensive one, the maintenance process."

http://www.osgi.org/blog/2007/04/why-osgi-technology-is-strategic.html

The TIM Lecture Series provides a forum that promotes the exchange of knowledge between university research and technology company executives and entrepreneurs. Readers outside the Ottawa area who are unable to attend the lectures in person are invited to view upcoming lectures in the series either through voice conferencing or webcast. Instructions for joining a lecture are available (http://www.talentfirstnetwork.org/wiki/index.php?title=Instructions_to_join_via_voice_conference_or_webcast).

On August 27, 2008, Dwight Deugo from Carleton University delivered a presentation entitled "OSGi and Server-Side Eclipse". This lecture introduced the fundamental concepts of OSGi, a component integration platform to provide interoperability of applications and services. It also discussed how the Equinox project has incorporated OSGi into the Eclipse platform and gave an overview of the impact it has had on Server-Side Eclipse. This report provides the key messages from the lecture. The slides from the presentation are available for download (http://www.talentfirstnetwork.org/wiki/images/d/de/OSGi_and_Server-Side_Eclipse.pdf).

Background & Framework

The first half of this lecture provided an overview of the OSGi Alliance (http://www.osgi.org).

According to its website, the "OSGi Alliance is a worldwide consortium of technology innovators that advances a proven and mature process to assure interoperability of applications and services based on its component integration platform. The OSGi Service Platform is delivered in many Fortune Global 100 company products and services and in diverse markets including enterprise, mobile, home, telematics and consumer. The alliance provides specifications, reference implementations, test suites and certification to foster a valuable cross-industry ecosystem. Member companies collaborate within an egalitarian, equitable and transparent environment and promote adoption of OSGi technology through business benefits, user experiences and forums."

Dwight explained the advantages of using OSGi in practical terms. Over the years, software has become more complex, increasing the need for collaborative frameworks. In particular, restarting software is problematic for distributed systems and systems that need to be always available. The OSGi attempts to solve this problem for Java by using bundles which can be individually installed, uninstalled, upgraded, started, and stopped without having to restart the rest of the system. Moreover, OSGi is an important framework as it represents the collaborative talent of a consortia of vendors.

The audience asked about other frameworks which attempt to solve this or similar problems. Jini (http://en.wikipedia.org/wiki/Jini) is another framework that attempts to solve the restart problem, but it was developed by one vendor (Sun). It is doubtful that OSGi will be replaced by Jini as the OSGi community is very strong. OSGi does not really compare to EC2 (http://en.wikipedia.org/wiki/Ec2) as EC2 uses virtual machines rather than bundles.

OSGi is not meant to replace Maven (http://en.wikipedia.org/wiki/Apache_Maven) as OSGi manages runtime dependencies rather than buildtime dependencies. From the presenter's experience, OSGi pulls down the first dependency it finds, meaning conflicting dependencies need to be addressed during the development process. Lazy starting in the bundle manifest is one way to optimize bundles. OSGi is specific to Java and Dr. Deugo was not aware of any frameworks based on OSGi for other languages. There are, however, many home-grown examples of non-OSGi solutions; an example would be the download feature of an antivirus project.

It was noted that there were very few telecoms represented in the consortia; a similar standard for this industry would be very useful.

Eclipse Equinox & Server Side

Dwight started this section with a demonstration of two OSGi implementations: knopflerfish (http://www.knopflerfish.org/) and Eclipse Equinox (http://www.eclipse.org/equinox/). The "goal of the Equinox project is to be a first class OSGi community and foster the vision of Eclipse as a landscape of bundles." He also provided several key messages for the Java developer:

- when converting to a server side implementation, you can either embed the web application into OSGi or give the WAR file to Tomcat (http://en.wikipedia.org/wiki/Apache_Tomcat) which treats Equinox as a servlet
- when creating OSGi servlets, you still need to follow the servlet API (http://en.wikipedia.org/wiki/Servlet)
- you can encapsulate existing servlets into an OSGi bundle, thus gaining all of the benefits of a bundle
- OSGi bundles force the developer to deal with services (e.g. stopping, starting) and dependencies early in the development cycle

Dwight Deugo received his M.C.S and Ph.D. degrees in Computer Science from Carleton University, Ottawa. He was the Editor-In-Chief of the Java Report and the Director of Java Services at The Object People before joining Carleton University in 1997. Dr. Deugo has immersed himself in objects for more than 18 years and has done extensive consulting in object-oriented systems, particularly in areas related to Java, Smalltalk and Eclipse. His research interests include Large-Scale Distributed Object Computing, Eclipse, Agents, Peer-to-Peer Computing, Evolutionary Computation (Genetic Algorithms, Genetic Programming, Artificial Life), Object-Oriented Systems and Software Patterns.

Recommended Resources

Modular Java Web Applications
http://www.scs.carleton.ca/~deugo/thesis/simon-kaegi/thesis-sk-final.pdf

The Equinox Community: Runtime Technology at Eclipse
http://www.eclipsecon.org/2008/sub/attachments/The_Equinox_Community__Runtime_Technology_at_Eclipse.pdf

"Keystones can increase ecosystem productivity by simplifying the complex task of connecting network participants to one another or by making the creation of new products by third parties more efficient."

Marco Iansiti and Roy Levien
http://hbswk.hbs.edu/item/3967.html

On September 3, 2008, Mike Milinkovich, Executive Director of the Eclipse Foundation (http://www.eclipse.org), delivered a presentation entitled "A Practitioners Guide to Ecosystem Development". This lecture for the TIM Lecture Series introduced the fundamental concepts of ecosystems and how the Eclipse Foundation matches the theory. This report provides the key messages from the lecture. The slides from the presentation are available for download (http://www.talentfirstnetwork.org/wiki/images/5/58/A_practioners_guide_to_ecosystem_development_Sep_3.pdf).

Ecosystem Best Practices

The Eclipse Foundation is a not-for-profit organization which is globally known for its success in creating and nurturing a multi-billion dollar, worldwide ecosystem model that spans hundreds of companies and thousands of products. Yet, this achievement was mostly accomplished in blissful ignorance of business ecosystem theory (http://en.wikipedia.org/wiki/Business_ecosystem), providing a working example of where theory does not match the Eclipse Foundation's experience. [**Editor's note:** see the Recommended Resources section at the end of this article for some theoretical references.] In this section of the lecture, Mike introduced those ecosystem concepts which proved successful within the Eclipse ecosystem.

The key component to an ecosystem is the platform foundation. In the case of Eclipse, this is a software development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. The keystone--in this case, the Eclipse Foundation--needs to evolve the platform. This is always a tricky balancing act as predictability is key to the rapid adoption of an evolving platform. Surprisingly, a flexible platform creates more value than one where all the details were planned. Modularity needs to be designed into the ecosystem's platform foundation, allowing for the creation of niches which in turn uniquely allows complementors to participate in platform development. Niche creation is important for ecosystem growth.

There are other factors which influence the success of an ecoystem. Transparency and openness provide a level playing field and encourage companies to join the ecosystem. Having a non-profit as the keystone is important, but the non-profit has to be capable of selling the business value of joining the ecosystem. Open source is a particularly powerful technique for establishing and rapidly growing ecosystems. Value is created by linking people to other people within an ecosystem and across ecosystems. How to better engage consumers is the next learning stage for ecosystems.

How to measure the health of an ecosystem is an emerging science and we need to create tools to measure ecosystem health. The components so far defined as being necessary to the design of a healthy ecosystem are: i) transparent intellectual property regime; ii) strong leadership and governance; iii) platform foundation that evolves; iv) architecture of niches; v) global reach mechanism; vi) orchestrators; vii) mechanisms to improve health of ecosystem, and viii) business outreach such as strategic selling.

However health is measured, the health of an ecosystem is more important than, and not necessarily related to, its size.

Other ecosystems lessons learned included:

- many still don't understand business ecosystems and confuse supply chains with ecosystems
- ecosystems are much more effective than vertical supply chains in innovation driven industries
- the provincial and federal governments do not understand open source or ecosystems
- you win by letting go
- even the keystone can't control an ecosystem
- the vision may die if champions leave the ecosystem
- don't attempt to predict winners; instead, enable the creation of niches

Eclipse Ecosystem

This section provided insight into how the Eclipse ecosystem works. Eclipse began as a rewrite of IBM's proprietary VisualAge for Java which was strategically released as open source in order to gain market share for the Java IDE. This strategy has succeeded in that several competing products, such as Borland's J Builder, are now built with Eclipse. It is hard to say if Eclipse would have been as successful if it had been launched by a small company instead of being launched by IBM. Sun isn't involved in the Eclipse ecosystem as it is perceived as an IBM strategy to remove Sun's control of Java.

The Eclipse platform is suitable for both B2C (http://en.wikipedia.org/wiki/B2C) and B2B (http://en.wikipedia.org/wiki/B2b). Further, the Eclipse Public License (EPL, http://opensource.org/licenses/eclipse-1.0.php) is a copyleft license which is less viral than the GPL, making it attractive to many companies. Most companies that make money with Eclipse wrap their products (for example, Lotus Notes) around Eclipse, though some companies ship a product that expects Eclipse to be pre-installed.

One challenge faced by the Eclipse Foundation is how to quantify the user base. To gauge the size of the ecosystem, the last version of Eclipse shipped with an opt-in data collector and Eclipse staffers use social networking tools such as Facebook and Linkedin. The Eclipse Foundation would like to make users more sticky so that new users don't just download once and never return or become contributors to the ecosystem.

Another challenge is the annoyance caused by complementors' changes that break the platform. The Eclipse Foundation does not manage this as the negative feedback from the rest of the ecosystem generally encourages the problem to be quickly fixed. There have been discussions about a certification program for Eclipse components, but this will probably never gain concensus as many large companies provide their own certification and some companies would fail the certification process.

Open Source

The final section discussed the role of open source within a business ecoystem. The natural affinity can be seen with the view that open source is as much a business phenomenon as it is a social phenomenon.

Partners entering an ecosystem still have to move up the open source maturity model (http://www.osbr.ca/ojs/index.php/osbr/article/view/351/312) one step at a time.

While standards bodies differ from ecosystems, open source is complementary to standards organizations as it speeds up the adoption of the standard. The increasing trend towards the transparency embodied in open source means that standards bodies who lack transparency are a dying breed.

Mike concluded the lecture with a discussion on the types of contributions that occur within the Eclipse ecosystem. Most of the contributions are from commercial entities. Of the contributions classified as individual (i.e. non-corporate), many of these contributors are paid by a company whose corporate policy prohibits using a work email on a public mailing list. A small percentage of individual contributors aren't paid by a company and building their own reputation is a key motivator for contributing. Sadly, the percentage of government contributions is very low. There are some EU funded projects, but no projects are directly funded by the US or Canadian governments. Academic contributions are also low; a notable exception is Mylyn (http://www.eclipse.org/mylyn/) which came out of a thesis from the University of British Columbia.

Mike Milinkovich is the Executive Director of the Eclipse Foundation. In the past, he has held key management positions with Oracle, WebGain, The Object People, and Object Technology International Inc. (which subsequently became a wholly-owned subsidiary of IBM), assuming responsibility for development, product management, marketing, strategic planning, finance and business development.

Recommended Resources

Business Ecosystem as the New Approach to Complex Adaptive Business Environments
http://www.tut.fi/units/tuta/tita/tip/Peltoniemi_Vuori_eBRF2004.pdf

Digital Business Ecosystems
http://www.bioteams.com/2008/07/31/new_digital_business.html

Business Ecosystem: A Conceptual Model Of An Organisation Population From The Perspectives Of Complexity And Evolution
http://www.tut.fi/units/tuta/tita/tip/2004_reports/Peltoniemi_business_ecosystem.pdf

Ecosystem Strategy: Keystones and Dominators
http://www.keystonestrategy.com/pdf/EcosystemStrategy.pdf

October 27-31

ACM International Conference on Multimedia

Vancouver, BC

ACM Multimedia 2008 covers all aspects of multimedia computing: from underlying technologies to applications, theory to practice, and servers to networks to devices. The technical program will consist of plenary sessions and talks with topics of interest in:(a) Multimedia content analysis, processing, and retrieval; (b) Multimedia networking and systems support; (c) Multimedia tools, end-systems, and applications; and (d) Human-centered multimedia.

http://www.mcrlab.uottawa.ca/acmmm2008/

November 3-December 8

Eclipse Training

Ottawa, ON

The Eclipse Foundation, in partnership with Eclipse members, is offering a series of training classes. This is your opportunity to learn Eclipse techniques, tips and tricks from experts. The instructor-led training courses will feature classes on Eclipse Basic RCP, Eclipse Advanced RCP, Equinox OSGi and Eclipse Modeling. Courses are available at cities across the globe, with team members from Ottawa's Code9 (http://code9.com) presenting in Ottawa, Austin and Portland.

http://www.eclipse.org/community/training/2008fall.php

November 12

Webcom Montreal

Montreal, QC

Webcom is the source of inspiration for any e-marketing specialists, communicators, developers, bloggers, decision makers and entrepreneurs. Over 400 people meet and exchange on the newest web strategies.

http://www.webcom-montreal.com

November 13-14

StartupNorth

Toronto, ON

Created for and by entrepreneurs, StartupNorth aims to educate and inspire by connecting you with other entrepreneurs, mentors, and the ecosystem of support needed to create and operate a successful startup in Canada and the world.

http://www.startupnation.ca/

September 2

Library and Archives Canada: A Core Partner of the Open Library Environment (OLE) Project

Ottawa, ON

Library and Archives Canada (LAC) is participating in the Open Library Environment (OLE) Project which will develop a design document for a next-generation open-source library automation system. LAC's contribution will be significant, will bring an added perspective to the project, and will provide another opportunity to find innovative solutions to how both library and archival collections are managed and made accessible.

http://www.librarytechnology.org/ltg-displaytext.pl?RC=13516

September 23

Open Source Documentary at the Festival du nouveau cinéma

Montreal, QC

The National Film Board of Canada and its co-producers will be out in force at the 37th Festival du nouveau cinéma, with thirteen films, including five world premieres. Also enjoying its world premiere is Rip: Remix Manifesto, the world's first open source documentary, exploring copyright, music and remix culture. Using footage from a variety of sources, Brett Gaylor looks at what happens when software meets culture and law, and includes interviews with Girl Talk, Cory Doctorow, Lawrence Lessig and Gilberto Gil. Produced by Mila Aung-Twin (EyeSteelFilm) and Kat Baulu (NFB).

http://www.nfb.ca/press-room/communique.php?id=19058

October 6

Canadians Launch Internet for Everyone Campaign

Victoria, BC

Telecommunities Canada (TC) today launched the "Internet for Everyone" campaign that seeks to put a national ICT strategy back on the federal agenda. As part of any such national strategy, the primary concern of TC members, community-based practitioners who are supporting this campaign, will be the question of digital inclusion. Once a leader in Internet access, Canada is now facing a harsh reality as the early promise of achieving universal digital inclusion has not been realized. According to a recent OECD study (2007), Canada went from 2nd to 10th place on the list of connected nations with only 26.6 broadband subscribers per 100 inhabitants.

http://www.internetforeveryone.ca/en/news/canadians-launch-internet-for-everyone-campaign.html

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

November 2008	Health and Life Sciences
December 2008	Enabling Innovation
January 2009	Enterprise Participation
February 2009:	Commercialisation
March 2009:	Geospatial
April 2009:	Open APIs

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.

A unique Master's degree for experienced engineers.

http://www.carleton.ca/tim

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

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.