OSBR.CA

The Open Source Business Resource

JANUARY 2010

JANUARY 2010

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
Leslie Hawthorn
Chris Hobbs
Rikki Kite
Thomas Kunz
Michael Weiss

The editorial theme for the January issue of the OSBR is "success factors". Which factors separate the open source projects that provide quality software and receive wide-spread adoption from other projects which are not well maintained? What traits should a business look for when considering which open source software to use? How does a company decide which open source project to contribute to, partner with, or use as a base to build its products or services?

The authors in this issue explore: the importance of well defined processes, the value of documentation to end users, the diverse tasks of a community manager, the value provided by participants who don't contribute code, and how a community can assist in creating training materials. Each concentrates on a particular success factor, and as a whole, provide a fuller picture of what to look for in a successful open source project or company.

George Neville-Neil, a member of the FreeBSD Project's core team, examines how one particular open source project has developed processes which provide its users, customers, and partners with a product that is stable, reliable, and long lived.

Janet Swisher, a professional technical writer, describes the importance of user assistance to the success of open source projects and offers some suggestions on fostering community contributions to open source user assistance.

Brent McConnell, a Community Consultant at Collabnet, presents some of the actions open source community leaders can take to concurrently deliver results and a system that encourages productivity and longevity.

Mekki MacAulay, Principal of OS-Strategy, discusses how passive participants in open source ecosystems play an important role in value creation in the ecosystem.

Belinda Lopez, Training Project Manager for Canonical, explores curriculum creation models and some of the conditions that are necessary for successful collaboration between creators of existing open source documentation and commercial training providers.

Mike Milinkovich, Executive Director of the Eclipse Foundation, answers the question "How can a community be considered "open source" if its primary objective is to promote commercialization?".

As always, we encourage readers to share articles of interest with their colleagues, and to provide their comments either online or directly to the authors. We hope you enjoy this issue of the OSBR. Starting on January 8, we will offer a weekly column written by open source experts, in addition to the monthly issue of the OSBR. Our first columnist will be Stephen Huddart, Vice President of the J. W. McConnell Family Foundation. You can view the column on the OSBR website (http://www.osbr.ca) and blog (http://osbrca.blogspot.com).

The editorial theme for the upcoming February issue of the OSBR is Startups. Submissions are due by January 20--contact the Editor if you are interested in a submission.

Dru Lavigne

Editor-in-Chief

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 BSD Magazine and is the author of the books BSD Hacks, The Best of FreeBSD Basics, and the upcoming Definitive Guide to PC-BSD.

FREEBSD PROJECT'S PROCESSES HELP COMPANIES

"An integral part of our business strategy is to fund FreeBSD developers to continually improve and refine FreeBSD. Our internal infrastructure and services are based on FreeBSD, allowing us to easily identify areas in FreeBSD that need improvement. The BSD license allows us to both use and contribute code freely, and allows our customers to do the same. FreeBSD has benefited from years of open sharing and collaboration which have resulted in a stable, mature, high performance operating system."

Josh Paetzel, Director of
Information Technology, iXsystems

The processes that open source projects use to produce new work and maintain the quality of their code base is a subject that comes up infrequently in discussions of open source. One reason for this is that engineers and programmers are usually loathe to deal with issues that are not directly related to the piece of code or technology that they are working on.

Successful businesses know that good processes lead to continued success. The attributes that attract a business to an open source project are stability, reliability, and longevity. Stability gives a business the confidence to invest time into developing products on the project's platform, safe in the knowledge that the next incremental step in development won't be torpedoed by some unforeseen change. Reliability is often not associated with open source and many projects are perceived as being too cutting edge for a business to build upon. Longevity is of value as many businesses are inherently conservative in their approaches, attempting to reduce the risks of adopting any technique or technology. One way to reduce risk is to work with an open source project that has a proven track record of delivering quality products, on schedule.

This article attempts to dispel the myth of the perceived tension between a formally run business and the apparently less formally run open source projects with which a business interacts. We describe how one particular open source project has developed processes which provide its users, customers, and partners with a product that is stable, reliable, and long lived.

Commit Process

The FreeBSD Project (http://freebsd.org) develops and supports a complete operating system and set of development tools. The resulting software meets the needs of a wide ranging user community which encompasses software developers, systems integrators, appliance developers, embedded systems designers, Internet service providers (ISPs), web hosting companies and just about anyone who needs a solid, Unix-like, software development and deployment environment. While it is possible to develop a rudimentary operating system for a small community of hobbyists with few rules or regulations, providing a system with the depth and breadth of FreeBSD has lead the FreeBSD community to develop a set of internal processes. These processes serve the community as well as the community's customers, many of whom build their products and businesses on top of FreeBSD.

When discussing an open source project, stability is most relevant to the organization of the project itself. The process by which contributors add to the system, how software releases are made, and how the project governs itself all contribute to whether or not the project as a whole is stable and determines its longevity. The FreeBSD Project has a well defined process by which contributions, and contributors, become a part of the Project itself.

FREEBSD PROJECT'S PROCESSES HELP COMPANIES

The most common reason for adding code or documentation to the system is to address a user need that is not met by the currently released software. This need may be as simple as a software patch or as complex as a new device driver, network protocol, or compiler. Most users can not directly change the code and documentation in the Project's repositories and need to make their need known to the community. FreeBSD has several groups of people who work directly on the source and documentation that make up the Project, including the source code, third party software (known as ports, http://www.freebsd.org/ports) and documentation. A committer is someone who has a commit bit, which gives them the ability to commit changes to the Project repository. The process by which someone becomes a member of the community is an intrinsic part of the stability of the project itself.

Let's take a simple example. A developer who is using FreeBSD needs something added to the system. The developer joins a few FreeBSD mailing lists (http://freebsd.org/community/mailinglists.html) to mention their need and that they are willing to contribute code in order to get the new feature added. The developer becomes known to some segment of the community. The developer sends patches over the mailing lists and one or more Project members check and commit the contributed changes into the system. The developer continues to work with the community until at some point, quite naturally and with little fanfare, the people who have been committing this person's changes decide that they should have direct access to the source code. The developer is "proposed for a commit bit" when someone from the Project sends an email to the Project's core team asking that the developer be made a part of the Project.

The core team (http://freebsd.org/doc/en_US.ISO8859-1/books/dev-model/process-core-election.html) is a group of Project members who are periodically elected to help organize the Project. One of the responsibilities of the core team is to accept or reject new members. Because every developer is known by some segment of the community before they are proposed, there are rarely instances of someone being refused a commit bit. One of the important measures of a new committer has to do with temperament. The Project promotes excellent software developers and the community wants members that have the social skills to "play well with others." Developers who are unable to get along with others are not proposed for commit bits, no matter how brilliant their code, because everyone on the Project knows that poisonous programmers (http://www.youtube.com/watch?v=ZSFDm3UYkeE) are not the people they want to deal with on a daily basis.

Being accepted as a committer does not end with the commit bit being granted. A new committer is given a mentor, usually the person who proposed them for the commit bit, through whom they must pass all their proposed changes. The mentor reviews all changes, ensuring they match the Project's development processes. The FreeBSD Project has a consistent coding and documentation style (http://freebsd.org/doc/en_US.ISO8859-1/books/developers-handbook) and the mentorship period is used to pass this knowledge on to the mentee. After some amount of time, the mentee is freed from mentorship by the mentor. This process which governs how a user or outside contributor becomes a member of the Project, with its multiple levels of vetting and its open and accountable nature, has contributed greatly to the Project's stability.

Release Engineering Process

Another process which contributes to the stability of FreeBSD as a platform for development is the Project's release and branch management process. What can be changed between and within releases is governed by a set of rules. There are always two major branches of development in the source code control system: CURRENT and STABLE. All changes go to CURRENT first, meaning that this development branch contains code that is still being tested. The STABLE branch is the one that the Project expects downstream entities, such as systems integrators and appliance developers, to base their products on. If a new feature in CURRENT is also needed in STABLE, that change is merged from CURRENT (MFC'ed) into STABLE. Ensuring that all new code goes into CURRENT first contributes to the stability of the overall system. New changes are fully tested within CURRENT without negatively impacting the broader community.

When a major software release is made, the version numbers for CURRENT and STABLE change. For example, with the release of version 8.0, STABLE is now the 8 branch and CURRENT is the 9 branch. Once a major release has been made, all of the system calls and kernel application programming interfaces (API, http://en.wikipedia.org/wiki/Api) will not change within that branch. Any change that would cause binary incompatibility must go into CURRENT and will never be placed into a released branch. As bug fixes and non-destabilizing improvements are made to the system, minor version releases are made. For example, the 6 branch had releases numbered 6.0 through 6.4. The FreeBSD Project takes its commitment to API and kernel programming interface stability seriously, meaning that any code developed within a set of release branches will run on any future release of the same branch.

Releases are officially supported by the Project for two years from their point of release. Due to the pace of development, the only changes being made to an older branch are critical bug fixes, such as security advisories and bugs which might cause system instability. If a branch is still heavily used by systems vendors and other businesses, there is an extended end of life (EOL) process whereby that branch continues to receive bug fixes after the normal two year period has ended.

Code Tracking Process

Systems grow more reliable if they have the support of appropriate infrastructure and processes. The FreeBSD Project has, over time, acquired the resources to provide the community with access to a well maintained source code and documentation management system, as well as build farms and testing facilities. The entire code history of the project can be viewed on line (http://www.freebsd.org/cgi/cvsweb.cgi), or checked out from the project's source code repository. All of the changes to the system, going back to the initial import of code from the Berkeley Software Distribution, have been recorded and preserved.

Build clusters, donated by Sentex Data Communications, run an application known as tinderbox (http://wiki.freebsd.org/Tinderbox). The tinderbox system downloads the operating system code every day, or on some faster machines every few hours, and rebuilds that code. If the build fails, an email is sent to the mailing lists so that the offending problem can be found and fixed quickly. The tinderbox system ensures that development does not move ahead with partially broken code.

No software system can exist for any period of time without being subject to the occasional bug. The FreeBSD project

maintains a bug tracking database (http://freebsd.org/support/bugreports.html) that is open and accessible to the entire Internet so that any user may record a problem they find with the system. Project members are able to read and respond to problems in the database and there is a bug busting team to make sure that bugs do not get stale from lack of attention.

Security issues are special types of bugs, and these are often treated outside the normal bug tracking system so that they do not become 0 day exploits (http://en.wikipedia.org/wiki/Zero_day_attack). The Project has a security officer who is the main point of contact for all security related issues (http://security.freebsd.org). The security officer works with a security team to investigate security issues and to properly record, report and repair vulnerabilities found in released systems. The policy of the FreeBSD Project favours full disclosure of vulnerability information after a reasonable delay to permit the safe analysis and correction of an issue, testing of the fix, and coordination with other affected parties. The coordination aspect is important to vendors who use the system in their products. While a project should never hide security issues, it has a responsibility to make sure that effected parties are able to properly patch their systems before the vulnerability becomes more widely known.

Other Success Factors

The FreeBSD Project has been producing releases for over fifteen years. What factors have contributed to its longevity? One of the key factors in the success of the FreeBSD Project has been its inclusive and democratic nature. Unlike many other open source projects, FreeBSD has never been owned or controlled by a single individual. The core team is made up of 9 members of the FreeBSD community, chosen through a voting process where anyone who has made a commit to the system in the last year is eligible to vote. The core team wields little direct power: it can not hire or fire project members, has no budget, and manages no schedule.

The main responsibility of core is to act as a focal point for the non-technical aspects of the Project such as thinking strategically about the Project, vetting new developers, tracking commit bits which might be retired or reactivated, acting as a public face for the Project, and helping to work out any non-technical problems that may come up in the day to day running of a large project. The core team also creates "hats" for significant areas of responsibilities within the Project. When a new area of responsibility crops up, core looks within the community to identify the person, or people, who would be best suited to wearing that particular hat. While core can suggest to someone that they put on a hat, they have no ability to force anyone to do so. Much as new developers are part of the community by the time they are proposed for a commit bit, the people who wind up with new hats were usually already doing the job in an unnamed and unrecognized capacity. Giving a person a hat is usually more about recognition than management.

A significant challenge for many open source projects is the need for a public, legally incorporated entity in order to work with vendors, sign contracts, retain copyrights and generally interact with the rest of corporate culture. The FreeBSD Foundation (http://www.freebsdfoundation.org) fulfills this role for the FreeBSD Project, acting as a legal entity that can perform all of the duties necessary for the Project to interact with both companies and individuals.

One important area that has lead to the longevity of the Project has been its commitment, from the very beginning, to top notch documentation. While many projects expect new users and integrators to "read the source", the FreeBSD Project has an entire team (http://www.freebsd.org/docproj) that works exclusively on documenting the Project, both its internal rules as well as its outward facing manual pages, user guides, FAQs (frequently asked questions) and the like. Documentation is currently produced in twenty different languages including non-Western languages such as Mandarin, Japanese and Korean. The documentation team maintains documentation both outside and within the source tree, such as the manual pages which are available on every installed system. Documentation gives the project coherence to those who build systems on top of it. Without it, the Project would not have been usable to enough people to survive for very long.

An operating system and development tools are not sufficient for all the every day needs of the users of FreeBSD. A tremendous number of external programs are made usable on FreeBSD by the members of the ports team. Like the documentation and the source teams, the ports team is self-organizing and is coordinated by the people who wear the ports hats. The ports team now manages over 20,000 software ports (http://freshports.org) which can be installed on FreeBSD, giving the user community access to a full range of applications, from email, to web browsing, to specialized libraries that help FreeBSD system integrators to build products.

Summary

The FreeBSD project provides a stable and reliable platform which systems integrators, ISPs and developers can use to produce their own products. This stability and reliability has been achieved by the careful selection and promotion of an effective set of processes covering all aspects of the Project, including source code control, release management, who can be a Project member, and how the Project itself is governed.

The processes described in this article can act as a guide for businesses who are looking to evaluate the stability of an open source project they are considering interacting with. Members of other open source projects may also find practices that they can integrate into their own processes.

George Neville-Neil works on networking and operating system code for fun and profit. He also teaches various courses on subjects related to computer programming. His professional areas of interest include code spelunking, operating systems, networking and security. He is the co-author with Marshall Kirk McKusick of The Design and Implementation of the FreeBSD Operating System and is the columnist behind ACM Queue's "Kode Vicious." Mr. Neville-Neil earned his bachelor's degree in computer science at Northeastern University and is a member of the ACM, the Usenix Association and the IEEE. He is an avid bicyclist and traveler who currently resides in New York City.

Recommended Resources

How the FreeBSD Project Works (video)
http://www.youtube.com/watch?v=nNkqKdLm1rU

How the FreeBSD Project Works (pdf)
http://2007.asiabsdcon.org/papers/P08-slides.pdf

FreeBSD Release Engineering
http://www.freebsd.org/doc/en_US.ISO8859-1/articles/releng/article.html

"When you ask someone to create a manual, be sure they know who and what it's for. Be sure they know that the goal is not simply To Accurately Document The Thing, but to Help The User Kick Ass."

Kathy Sierra
http://tinyurl.com/l7ufz

This article describes the importance of user assistance to the success of open source projects and offers some suggestions on fostering community contributions to open source user assistance. The term "user assistance" encompasses all the ways that users get help in figuring out how to use a product, spanning the traditional categories of both documentation and support. User assistance provides opportunities for participation by community members who are not software developers. This in turn relieves the burden on developers for filling these roles while broadening the community. Projects should support the differing motivations of members in these roles while providing leadership and direction, removing barriers to contribution, and engaging in concerted efforts. Licensing for open source documentation should likewise be open, to support user freedom and foster community collaboration. Leaders in open source user assistance need to share ideas across projects in order to improve their offerings.

Importance of User Assistance

In the fall of 2009, the poor state of documentation for Linux and for open source projects in general became a trending topic in blog posts and online magazine articles. Writers lamented the lack of documentation relevant to user's need, the surplus of disparate and outdated information, the lack of responsiveness of developers to user problems, and the poor attitude of developers toward documentation. However, few offered much in the way of constructive advice, with the notable exception of Bruce Byfield's article on “Information sources for documenting free software,” in Linux Magazine (http://linux-magazine.com/Online/Blogs/Off-the-Beat-Bruce-Byfield-s-Blog/Information-sources-for-documenting-free-software).

This article describes the importance of user assistance to the success of open source projects, and offers some suggestions on fostering community contributions to open source user assistance.

Creating good user assistance is important to open source projects for multiple reasons:

- it helps users become effective in using the product, which spurs adoption of the software
- people who might not otherwise be able to contribute to the project can help with user assistance, which expands and fosters the community around the project
- it can free developers from dealing with support issues
- user assistance efforts can provide filtered feedback on areas for product improvement

User Assistance is More than Manuals

The term “documentation” tends to limit thinking to traditional manuals and perhaps on line help.

The term “user assistance” encompasses all the ways that users get help in figuring out how to use a product. User assistance can include:

- text embedded in the software, including field labels, window titles, and error messages (for end users) and comments in the code (for developers)

- online documentation accessed from within the application
- formal manuals or books
- smaller standalone documents such as how-tos, FAQs, manual pages or tutorials
- video tutorials or demos
- mailing list or forum questions and answers
- live chat with experts or other users

These different forms of user assistance vary in their degree of formality, granularity, searchability, and responsiveness. Information in any of the categories listed can target a variety of audiences such as new users, expert users, programmers, or administrators. Most can be delivered through a variety of media, including blogs, wikis, traditional websites, or paper documents.

Users turn to different forms of assistance at different times in their usage of a product. Using Google to search forum postings is not a substitute for other forms of user assistance; rather it is a point in a continuum of user assistance. It can be useful to think of a user's experience of a software product as a conversation that the user engages with the product, the community, and various artifacts. Over time, ideally, the user not only seeks answers to questions, but also offers information that he or she discovers. Anne Gentle discusses this process in Conversation and Community: The Social Web for Documentation (http://xmlpress.net/publications/conversation-community).
While her book is aimed at professional technical writers rather than open source community members, it can be helpful in seeing how social media can feed into and gain from traditional documentation.

Many Roles can Contribute

Open source projects are typically started by programmers to “scratch their own itch.” Programmers have the skills to create the code, but may or may not have appropriate skills to grow a community or create user-focused documentation. Even if they have the skills, those other activities may not be the most effective use of their time. The responsibility of a project leader is to find other participants who do have the appropriate skills and to set them loose. The need for user assistance is an opportunity for involving and growing the project's community. Creating user assistance can be a way for people who are new to the project or otherwise not directly involved in programming to contribute significant value.

While the project's programmers have in-depth technical knowledge of the product, they are often not the best people to write the user assistance. This has little to do with writing skills, and a great deal to do with perspective. It can be difficult for someone with an intimate knowledge of the technical details of the product to adopt the perspective of someone who has never seen it before. This holds true even for software libraries whose audience is other programmers. The programmer using a library's application programming interface (API) may lack the conceptual framework that is “like water to a fish” for the programmer who wrote the library. It is often more useful for someone other than the original programmer to write the user assistance, using the original programmer as a “subject matter expert” for technical details.

An important role in creating large-scale documentation is reviewers to review drafts. Unlike programs, documents cannot be automatically checked beyond the rudimentary checking provided by spelling-checkers.

Human reviewers are needed for copy-editing and proofreading, to ensure that the document follows the conventions of the language, and for technical review, to ensure that it is technically correct. Reviewing can be an entry-level role for community members who don't yet feel sufficiently knowledgeable to create documentation.

The problem for some open source projects is not that there is too little information, but that there is too much, it is too widely or unevenly distributed, and it is too poorly coordinated or maintained. In such cases, curators are needed to discover and sift through the information that exists, catalog it, evaluate it, and provide pointers in a centralized location. This type of inventory can uncover areas that need updating and gaps where information is needed but does not yet exist. These needs can then feed the task list for writers and editors.

Another role can be for rewriters to repurpose and remix existing content in more accessible forms or for different contexts. For example, a mailing list thread for troubleshooting a problem contains back-and-forth and extraneous information that, for later readers, interfere with extracting the useful nuggets. A community member could abstract that thread into a troubleshooting topic on a wiki. For this to happen, someone must take on the task of watching for candidate topics on the mailing list. As an example of remixing, a volunteer with an educational background could create variants of existing documents for grade-school children, with supplementary information for educators.

Community and Personal Growth are Motivators

Unlike programmers, writers looking to "scratch their own itch" tend to find more expressive outlets than technical documentation, such as novels or poetry. In "Why do people write free documentation? Results of a survey" (http://onlamp.com/pub/a/onlamp/2007/06/14/why-do-people-write-free-documentation-results-of-a-survey.html), Andy Oram found that the two highest-rated motivations for writing open source documentation were community-building and personal growth. Reputation-building and enjoyment of writing were ranked relatively low as motivations.

It may be more motivating for writers to know that their work is being used and appreciated by the community than to get personal kudos for their work. Some automated tools exist to provide such information, such as website traffic logs to find which pages are most often read, and page ratings (supported by some web content management systems) to find which topics are most useful to users. This information can be shared with writers, and used to prioritize updates. An often-read but poorly-rated page should get more attention than a highly-rated but little-read one.

The motivation of personal growth is more intrinsic: people often write about a topic in order to learn it more thoroughly themselves. What project organizers can do to support this motivation is to tolerate and even encourage user assistance contributors who have less than expert technical skills. While existing experts may be too busy with other concerns to write documentation, other community members can become experts through the process of documenting. Alternatively, an expert can become a better writer through the practice of writing, especially when given constructive feedback from reviewers.

User Assistance Requires Leadership

In order to create a growing base of users and a thriving community, user assistance cannot be an ad hoc effort. User assistance encompasses both “support” and “documentation” as traditionally conceived. It may be practical to separate support and documentation activities, but close coordination between the two areas leads to better experiences for users. Support channels are a source of actual user questions that can become new documentation topics, while existing documentation can be a resource for support contributors.

Projects that are bigger than a handful of developers need leaders specifically responsible for support and documentation efforts. A support leader helps ensure that user questions are answered in a timely manner and that flame wars (http://en.wikipedia.org/wiki/Flame_war) and spam (http://en.wikipedia.org/wiki/Spam_(electronic)) are contained. A documentation leader coordinates efforts to plan, create, and maintain documentation. This person may actively participate in creating and editing documentation, or may focus on clarifying the community's vision for the documentation and guiding contributors.

For projects that have a corporate sponsor, the community may expect the sponsor to create the documentation. Whether this expectation arises may depend on the dynamics of the larger project. It is more likely to occur if the open source product was created primarily by the sponsor and released as open source, and less likely if the project evolved first. In the former case, the sponsor may fund development of primary documentation, while accepting community contributions. Such contributions are likely to be granular, such as specific configuration how-tos or troubleshooting topics.

Or, responsibility for documentation may be shared between the sponsor and the community. According to Jean Hollis Weber, Documentation Co-Lead for OpenOffice.org, the project's corporate sponsor, Sun Microsystems, maintains the online help that is accessible within the product, while the community project creates and maintains end-user manuals. The community documentation project also handles documents aimed at programmers and system administrators, but Weber reports that much of that content is contributed by Sun employees.

Lowering Barriers Enables Contribution

Community involvement in user assistance increases if barriers to involvement are kept low. User assistance activities should be advertised areas for involvement on the project's website, with a clear means of contacting leaders and discovering needed tasks.

Designating leaders for user assistance efforts does not absolve developers of involvement in these activities, but shifts it. Rather than directly providing support and writing documentation, developers need to be responsive to contributors in these areas, supporting them and answering their questions. For example, providing regular binary builds not only supports end-users, it also helps community members who are trying to help end-users. It should not be necessary to check out the source tree from version control and build the software from scratch just to contribute to documentation.

The technology used in documentation efforts can be a barrier if the tools require significant training for community members. Historically, many open source projects have used open but arcane documentation formats such as LaTeX

(http://latex-project.org) or DocBook XML (http://en.wikipedia.org/wiki/Docbook), which are unfamiliar to non-programmers. Some projects are now turning to less challenging formats. For example, the documentation for the Python programming language is now written in reStructured Text (http://docutils.sourceforge.net/rst.html), a wiki-like plain-text mark-up syntax, instead of LaTeX. The Gnome Documentation Project is starting to use Mallard (http://live.gnome.org/ProjectMallard), a much-simplified, topic-oriented alternative to DocBook.

The advent of wikis has made it possible to open a project's documentation to contributions by just about anyone. However, wikis bring their own host of issues. It is not reasonable to assume that creating a wiki will lead to documentation magically being written, or that it will automatically be comprehensive, well-organized, or easily searchable. A wiki is simply a platform and a documentation effort that uses a wiki still requires management at the social level. A wiki is inherently unstructured, and so a structure must be imposed on it to support users in finding information. A defined structure also helps writers in knowing where to plug in new pieces of information that they want to contribute.

A project that provides a purpose-built, wiki-based platform for open source documentation is FLOSS Manuals (http://en.flossmanuals.net). The site has been customized to support collaborative creation of comprehensive texts, with a WYSIWYG (http://en.wikipedia.org/wiki/WYSIWIG) HTML editor, an embedded IRC (http://en.wikipedia.org/wiki/Irc) widget, and document structuring tools. Documents can be rendered as HTML or PDF, and can be embedded in another website or uploaded to a print-on-demand service.

While the current site is a one-off instance, efforts are underway to reimplement the same features and more as a new open source web application called "Booki" for "book wiki". The project welcomes any documentation effort that discusses open source software, whether for a single application or spanning multiple applications.

Concerted Efforts Bring Ongoing Results

Open source programmers have long used "sprints" or "hackfests" as a way to boost a project by gathering contributors for a brief but concentrated effort. The same principle applies to documentation as well as to code. The FLOSS Manuals project has honed and refined the idea of a "book sprint", originally created by Tomas Krag to write Wireless Networking in the Developing World (http://wndw.net). A set of people who are interested in creating a book on a particular topic gather in a given place for a short period, typically two to five days. At the end of the sprint, the book is published to the FLOSS Manuals site. Holding a book sprint is not a requirement for creating a book on the FLOSS Manuals site, but experience has shown that books that were created in books sprints have the greatest level of ongoing participation.

Dave Greenberg is a co-founder of the CiviCRM project (http://civicrm.org) who participated in a FLOSS Manuals book sprint to create a book, Understanding CiviCRM (http://en.flossmanuals.net/CiviCRM), to supplement existing CiviCRM documentation. He says "The online doc (on the wiki) is procedurally oriented – lots of how-to's, but we got feedback that ... it didn't provide evaluators and new users with the big picture. 'Is this software right for my organization/client?' 'What can I do with it?' etc. … we wanted the book to help decision makers and consultants who were

evaluating software solutions to have some concrete examples of who, how and what." He reports that CiviCRM is planning to hold two more sprints in 2010 to update the first book and to write another with conceptual information for developers.

In contrast to short-term sprints, another strategy for concerted documentation effort was a "documentation marathon" to document the NumPy API (http://www.numpy.org), organized by Joe Harrington over the summers of 2008 and 2009. Participants were recruited through community mailing lists, and rewarded with T-shirts for significant levels of contribution. These efforts resulted in 78% of the API categories reaching the goal of having complete content ready for review.

The lesson for any open source documentation project is that setting and achieving a significant goal can have lasting effects on the project.

Open Source Software Needs Open Documentation

The principles that underlie free and open source software also apply to documentation. Users of open source software need to be able to read, study, modify, and copy the documentation of that software. While there is a place in the open source ecosystem for traditionally published, restrictively copyrighted books (even as publishers become more accepting of open licenses), the primary documentation for an open source product needs to be as open as the software itself. Restrictive licensing of documentation inhibits contributions to, collaboration on, and translation of that documentation. Open source projects where the official documentation is restrictively licensed may find users creating their own, unofficial, open-licensed documentation.

Like other issues related to open source licensing, the question of which license to use for open documentation is a matter of debate, with arguments for and against various specific licenses. Dual-licensing may be a reasonable choice in order to satisfy competing requirements such as compatibility with distribution guidelines and other licenses. It is logical for documentation that is distributed along with software to use the same license as that software, or a compatible license. For documentation that is distributed separately from the software, more flexibility is possible. Where documentation is created collaboratively, use of a specific license should be a condition for accepting contributions.

Good User Assistance is Achievable

In response to the blogs and articles about poor open source documentation, commenters cited examples of projects whose documentation they consider helpful, including OpenOffice.org, GnuCash, PHP, MySQL, Gnome, Ubuntu, OpenBSD, and OpenSolaris, which indicates that these projects are succeeding at helping at least some users "kick ass." The issues related to producing helpful user assistance are not unique to open source projects. Proprietary software companies face them as well: some proprietary software user assistance is excellent, and a large amount of it is poor. Employees of proprietary software companies share best practices through organizations such as the Society for Technical Communication (http://www.stc.org) and the Association of Support Professionals (http://asponline.com). Community members involved in open source projects can learn directly from these organizations, but also can and should share with and learn about best practices from each other.

In that spirit, June 2009 saw the first-ever conference on Writing Open Source for open source documentation leaders to gather and share ideas. That conference resulted in a website (http://www.writingopensource.com) that promotes collaborative discussion on topics ranging from writing to technology and community building. Good user assistance is achievable, but requires concerted effort, strong leadership, and a focus on user needs that is strengthened through dialogue and collaboration with user communities.

Janet Swisher's first experience with online collaborative writing was compiling the Twin Peaks FAQ for the Usenet group alt.tv.twin-peaks in 1991. She has been a professional technical writer for over ten years, at various technology companies in Austin, Texas. She has contributed to open source documentation for OpenOffice.org and the Python-based Enthought Tool Suite, and for a number of open source projects through FLOSS Manuals. She blogs about topics related to technical communication and open source software on her "Techie Tech Writer" blog at http://www.janetswisher.com.

"...success comes entirely from people and the system within which they work. Results are not the point. Developing the people and the system so that together they are capable of achieving successful results is the point."

Leading Lean Software Development
http://www.poppendieck.com/llsd.htm

Recently, that quote stirred some controversy among my peers. The part about "results are not the point" was hard for some people to understand and come to grips with. Aren't results always the point? Well, as with most things, "It depends". The people and community that evolve around an open source software project will ultimately determine its success. Even if the core team launches the project with spectacular productivity and results, this phase of evolution will be fleeting if the necessary processes and community to make the project a long lasting success are not put into place.

This article presents some of the actions open source community leaders can take to ensure not only results, but a system that encourages productivity and longevity.

The Law of Attraction

One of the fundamental principles of nature is that objects tend to attract other like objects. The term homophily refers to the tendency of individuals to associate and bond with others of a similar bent. This same principle of attraction is what pulls communities together and keeps them together. People are attracted to others that have similar interests or problems to overcome. It is that commonality that creates the link, the attraction, that holds communities together.

Unfortunately, many projects and businesses forget this basic principle. They instead believe that communities form around products, brands or buzzwords.

They forget that people want to belong to groups that they share some interest with. Providing a shared interest doesn't necessarily translate into building a vibrant, action oriented community. In order for a community organizer to stimulate results in a community, the following ingredients are needed:

- a mission that will attract others that are passionate because ...
- passionate users create excitement for a cause and ...
- excitement elicits action and results from the community

The goal is not simply to build software but to attract users that share a passion for a particular subject. It is this belief in the cause that will ultimately determine whether or not a community is successful.

Leadership

Leaders are people who see the world from a different and new perspective. Leaders dream of a future that is different from today. A leader's vision of tomorrow is inspiring and solves real problems for real people. But leadership goes beyond this by introducing others to a future that they can embrace as their own. The ability to make the vision their own is what draws people to an open source project and moves them to action.

How does a leader craft a message that resonates with the community? Listening is the key that unlocks not only the problems of the users but also their perspective. Leaders must understand where the pain points are and what motivates users. A leader's goal is to provide just enough of a blueprint for tomorrow so that users are able to finish crafting the story for themselves. This gives them ownership and enthusiasm to solve the problem.

This quote from the French writer and aviator, Antoine de Saint-Exupery, is especially important for community managers as it relates to creating a vision of the future that people believe in and want to become a part of: "If you want to build a ship, don't drum up the men to gather wood, divide the work and give orders. Instead, teach them to yearn for the vast and endless sea."

The other aspect of leadership that is often overlooked is the art of coalition building. As the message begins to resonate within a community and adapts to each user's needs, leaders need to manage the alternate messages that form within the community. Leaders have to continually revise the vision to include any new or divisive stories that develop. New leaders will emerge within the community that could have agendas that differ significantly from the original vision. These leaders may eventually harm the community if their ideas are not embraced early on and elements of their story are incorporated. Embracing and incorporating input builds a stronger community and additional leaders to help within the project. The community will be stronger with them than without them.

In Community We Trust

Trust influences nearly every interaction we have during any given day. Every communication, every action, every conversation is shaped in some way by the trust and reputation inferred on the interacting party. Trust is the currency that communities, both online and offline, trade in. Without trust, lasting relationships can not be built or maintained. Part of a community leader's job is to build reputation and trust for the people associated with a community.

Trust is not something you can ask for as it is earned through actions and competence. It defines relationships between people, governments, communities, and businesses. The text book definition of trust is "…reliance on the integrity, ability, or character of a person or entity". You can rely on someone or something when you have a history of past experiences by which you can infer future experiences. Without these past experiences, people have no way to place you within their trust metric. They resort to lumping you in with "the rest" or basing their trust on any reputation you may have.

As a community leader, you must build trust in you and your project. People trust people who get things done. If you say you're going to do something and never quite get around to it, your reputation will suffer and hence the community's trust in you. Remember, actions always speak louder than words.

Any Fool Can Criticize

Benjamin Franklin once said that "Any fool can criticize, condemn, and complain, and most fools do". One of the things that keeps people from getting involved in open communities is a fear of criticism. Criticism that they'll ask the wrong questions and criticism that they'll do something wrong. There are probably dozens of reasons people are afraid to participate and they almost always relate to being afraid of something. It is a leader's job to see that the community is a hospitable place for new people to participate.

Many project veterans may not have the patience to allow foolish questions to pepper the project's mailing lists or forums. They think that everyone should put in the same due diligence they did to understand the project and its code.

But, if you want the community to grow, you will need to set the example of always having a cool temperament, even with newcomers who may not have done their homework before asking a question. This is not to suggest that you coddle newcomers, but that you need to ensure that responses to questions are civil.

Recognition

Mary Kay noted that "There are two things that people want more than money and sex...recognition and praise". Especially early on, you'll need to work hard to ensure that every little contribution to the project is warmly welcomed. This may mean that you have to work with contributors to rewrite a patch or help them fill out a bug report. The name of the game is getting people to open up and get involved. This typically involves coaxing and lots of encouragement.

Don't be afraid to recognize new participants and draw attention to their accomplishments. If you are constantly praising your community users and helping them feel good about the work they are doing, you will find that members have a greater sense of responsibility towards your community efforts. Greater responsibility equals more action which results in a productive community. Communities run on recognition. This doesn't mean that you need a user rating system or a User of the Month classification. You simply need to express honest gratitude publicly for what community members are doing. Try it and you'll notice a remarkable difference in how the community starts behaving.

Simplify, Simplify

Henry David Thoreau once said, "Our life is frittered away by detail. Simplify, simplify". He was expressing a concern with the complexity of life while encouraging

people to strip away the unnecessary and to focus on the important. Communities sometimes forget that they have to present themselves in pure and simple terms in order to grow. The message must be simple. The ability to communicate should be simple. The tools must be simple. This is not because people can't understand complexity, it is because they don't have the time to. In order to grow a community, concentrate on the most important elements that have an impact. Simplify as many things as possible to get to what truly makes your community unique and beneficial.

An example of where projects sometimes fail in this area is by creating too many options for member communication. Don't implement every communication technology you can find as that will only make it harder for your members to find and participate in conversations. Communication tools should help your members to communicate, not distract them with choices. You should ask a single question when analyzing your community's tool choices: "Will this technology facilitate human interactions?". Always remember that communities are about people, not technology, and that simpler communication strategies are usually better.

Blog, Baby, Blog

With the move to social networking sites like Facebook.com and Twitter.com, the buzz around maintaining a blog has diminished. However, blogging is still one of the easiest and best ways to reach an audience with a message. Twitter and Facebook are important tools to help connect your project with a larger audience, but blogging is still the best way to create thought leadership around a project's mission and vision of the future. When blogging, show your passion for your subject.

Blogging about what you are doing is only the first step as you still have to attract people to the blog. Fortunately, blogs rank high in Google's PageRank algorithm (http://en.wikipedia.org/wiki/PageRank). The key with Google is not to go after the first page of results for a generic term like "collaboration" or even "collaboration software", but to find a search term that still gets a decent amount of traction. In the case of collaboration software, it is far easier to reach the first page of Google results for "collaboration community of practice" or "collaboration success" than for just "collaboration". When you title your blog, use the search terms you want to be found under, such as "Creating Communities of Practice Through Collaboration".

Don't just focus your outbound marketing on Google. Start investing in Twitter and Facebook to grow an audience for your message. These tools may not be the best for articulating your project's value proposition and mission, but they are great for helping you find pockets of users who share your passion. Make sure that you are following (http://help.twitter.com/forums/10711/entries/14019) and joining groups that have users who are attracted to the same problems and passions as your project and make sure you let these groups know when you've posted something on your blog. The key to using social networks is that you have to add value to your network by helping them solve their problems without becoming a marketing drone for your project.

Work With Other Projects

Being an active and productive citizen of other projects is a great way to introduce users to your project or solution. If you have a reputation for helping others and contributing to projects, people will be happy to lend a hand when you need it.

You may even already have some followers if you are actively participating in other communities. I recently heard the founders of GitHub.com talk about their startup experiences at the Open Source Bridge conference (http://opensourcebridge.org). They specifically mentioned their involvement with the Ruby on Rails community as one of the reasons GitHub had a successful beginning. If that's not a testament to playing well in the sandbox, nothing is.

Wrap Up

Building a community of passionate users is no small task. If you manage to do it, you will have worked harder than you ever have in your life because community building is a process that never stops. That is why it is so important to tap into a passion--not only the passion of a large set of users, but also your passion. The work is long and hard and often doesn't seem fruitful, but if you stick with it and let your passion for the project and the problem you are solving shine through, you'll do just fine.

Brent McConnell is a self-described "Community Guy" who has worked in and around open source software and communities since 1997 when he happened upon a copy of Slackware. He is currently a Community Consultant with CollabNet (http://www.collab.net), helping developer communities with adoption and reuse on the CollabNet TeamForge platform. He's also been the Community Manager for the Kablink (http://kablink.org) Open Collaboration platform, and iFolder (http://ifolder.com). Before all this "community stuff", he held jobs at Lucent Technologies, Compaq, and HP in various levels of engineering responsibility. He blogs regularly on community issues at http://mindby.com.

"I reject the notion that any user is a free-loader or a leech. At the very least, they are vectors for your software, getting it out there in real-world environments to show to other potential users."

Brian Proffitt, Community Manager of Linux Foundation

Passive participants in open source ecosystems should not be viewed as leeches as they contribute value to the ecosystem. Every eyeball has value. By better understanding the roles of passive participants in the ecosystem, keystone companies can assign resources, such as community managers, more effectively and better leverage the value these participants create. The next challenge is to better quantify the value of passive contribution.

This article discusses how passive participants in open source ecosystems play an important role in value creation in the ecosystem. It examines why the value they add is not well captured by current measures and suggests areas of future research, the outcomes of which would enable keystone companies to better position themselves.

Passive Participants in Open Source Projects

In her recent keynote speech at OSCON 2009 (http://infotrope.net/blog/2009/07/25/standing-out-in-the-crowd-my-oscon-keynote), Kirrily Robert talked about the participation of women in open source projects. She interviewed female participants in two open source projects: Dreamwidth (http://dreamwidth.org) and AO3 (http://archiveofourown.org). She reports: "So, what can we learn from this? Well, one thing I've learnt is that if anyone says, "Women just aren't interested in technology" or "Women aren't interested in open source," it's just not true. Women are interested, willing, able,

and competent. They're just not contributing to existing, dare I say "mainstream", open source projects."

What about passive participants? Is contribution to the code base the only way to measure participation in a software project? Gender equity issues aside, is it fair to assume that someone who doesn't contribute to the code of an open source project is not interested in technology or not interested in open source? Are contributors the only stakeholders in an open source ecosystem? Do they generate all of the value?

Franck Scipion of 55 Thinking (http://www.55thinking.com) talks about numerous other roles for participants in an open source ecosystem (http://slideshare.net/55thinking/understand-open-source-ecosystems). He describes roles that include new user support, collaboration facilitator, know-how sharer, evangelist, trainer, event organizer, donor, and users. From the perspective of code commits, these participants are all passive and these contributions to the open source ecosystem are not measured by the traditional scales. Yet they are clearly doing something important to contribute to the health of the ecosystem. Why are these roles marginalized?

Bill Snyder at InfoWorld writes about "open source leeches" (http://infoworld.com/d/open-source/fight-over-open-source-leeches-399). He describes the debate over the perception that those who don't contribute back in the form of code contributions are considered "open source vampires". This perspective only makes sense if you assume that roles are clearly delineated as developer and user. In this model, developers shoulder all the work and manage the projects. They contribute source code, debug problems, debate the merits of new features, and add them as needed.

Developers are the key players who chose the direction of a particular project and are the primary actors responsible for its success or failure. Open source projects are by developers, for developers.

Following this model, passive participants merely use the software. The developer community regards users as people who only take what was freely available and make no real contributions back to the project or community. Non-developers are often spoken of in disdain as problematic and opportunistic. Passive participants are thought to have nothing to offer open source projects. They are just freeloaders benefiting from the open licensing terms.

Where does this perceived divide come from? It may be the social barriers that grew in developer communities that made it such that only people who could write code, and who fully understood the internals of the open source project in question, could participate in development discussions. There is still frequent debate about open source elitism (http://linuxinsider.com/story/65853.html) that is thought to separate the classes of participants. In this divide, developers rebuke users for simply taking the work of others and giving nothing in return and the users rebuke the developers for not understanding the mainstream's wants and needs and for keeping development restricted to a small circle of people.

This divisive model misses the complexity of open source ecosystems. While some of its points may have validity, value generation in the ecosystem is not controlled purely by these factors. Passive participants are essential to value generation.

Users Provide Value

The emergence of the participatory Web has changed the nature of engagement in open source projects and is in the process of changing the definition of the roles different participants play. At the same time, open source projects are continually building upon one another, and combining in new and innovative ways to address business needs. Ecosystems have formed around some of the larger open source projects, and users make a significant contribution to the overall health and success of the projects.

In 1998, Netscape paved the way as the first widespread commercial open source success. It redefined the open source game, showing that companies could participate in, and even lead, open source projects and be successful. Netscape turned the traditional software business model on its head. It was in this breeding ground of new potential that the traditional model of participation in open source projects began to erode.

Prior to the open sourcing of its code, Netscape had been giving away its browser for free. The management team quickly realised that in order to compete with Microsoft's Internet Explorer browser, Netscape needed to put its browser in the hands of every single potential Internet user, and that the best way to do this was to give it away for free. The large number of users, in turn, would help fuel sales of Netscape's server products to companies who wanted to build an Internet presence and conduct e-commerce. The non-paying users of Netscape's browser played an integral role in the company's strategy.

When Netscape released its browser code as open source, many users fell into the traditional open source roles of developers and non-contributing users.

But, new user roles began to emerge. The advocate user came into being. These users do not contribute directly to a program's source code or development. But, after using the program, they spread the word by recommending it to their friends and colleagues. In this manner, they increase awareness of the product and bring in more users who may, in turn, make contributions in code or in another ways.

Advocates are just one class of user that has begun to emerge as an important player in open source projects. They add value to the project not through code contributions or feature testing and implementation, but by spreading the word about the project and adding value to the brand by increasing its awareness. In the case of Netscape, its brand was its most valuable asset. The strength of its brand led directly to its acquisition by AOL in the spring of 1999, in a deal worth over $4 billion dollars.

The popularity of a brand can be directly influenced by the number of people who use the brand's product. By allowing users to use and freely distribute the product, they become distributed marketing and promotional agents for the company. Companies should encourage and nurture user participation to improve their branding strategy and the reach of their market. Users are particularly useful for viral marketing through word of mouth referrals. This form of marketing greatly hastens the adoption of new products.

More than ten years later, observers of open source participation still see average users as unable to help themselves, let alone contribute anything meaningful. It is assumed that all users of open source software must have a profile that is comparable to a developer's in order to do anything other than passively consume.

Emerging User Roles

The participatory Web has put the Internet into the hands of the user. It enables anyone to readily generate and post content in a broad array of contexts. It has helped thousands of communities to grow and thrive around interactive products and services. For open source projects, it has also created a new class of user: the non-code-contributing user.

Originally, the only contributions a participant in an open source project could make were in the form of code. They could debug a problem or implement a new feature and submit the revised source code to be included in the project's next release. The technical barrier to entry effectively prevented non-programmers from participation. In recent years, non-programmers can add value to open source projects without ever writing a line of code.

Aside from the value that users generate with word of mouth and other promotion of open source products, projects benefit from the complementary works that non-code-contributing users create. In the Netscape context, this might come in the form of web pages. Every user who creates a web page and puts it on the Internet for other people to access is indirectly adding value to the Netscape browser by increasing the number of pages the product can be used to access. If there were only 10 web pages in the world, the Netscape browser would not be very useful. As the number of web pages grows into more content that any given person might want to access, that growth adds value to the tool they will use to access that content.

A more direct example can be seen with the virtual world Second Life. Users purchase land and build on it in a 3D environment. This building is more akin to graphic design or visual arts than it is to programming. Users create objects and share them with other users. They then use the space and their creations to offer services and games, run businesses, trade currency, and generally create all kinds of complementary products and services that make the experience of using Second Life's virtual world more appealing. Second Life released the code of their viewer as open source. This move promoted the development of hundreds of diverse third party applications and interfaces such as 3D headsets, terminals for the blind and Skype plug-ins. It has spurred the development of complementary assets that add value to their core product and brand. The parent company, Linden Lab, continues to generate significant revenue from network services provided within Second Life, and has attracted tens of thousands of new users through its open source efforts.

Users have played a central role in the Second Life ecosystem. Without the critical mass of users creating interesting spaces and using the project, all of the development work and code contributions that went into the product would go unused. These participants played a role in the coming into existence of all the derivative and complementary products that have and will emerge from the ecosystem. Without the passive participants, these products would have never existed, and the value capture for the companies that created them would not have occurred.

Suggestions for Companies

Since its founding in 1998, the Open Source Initiative (http://opensource.org) has encouraged the software industry to re-evaluate intellectual property strategy.

Many companies have been in a similar position to that of Netscape in 1998. They have an existing product, a growing user base with different motivations and skills, and revenue generation mechanisms. By taking a fresh look at the different classes of people who participate in the ecosystem they have built around their product, and broadening the operational definitions used to classify them, they are now in a position to be able to begin to quantify the value of passive users. The industry is full of data that is suitable to case studies of such situations.

Managers of companies considering open source strategies can begin planning and analyzing how to structure their open source projects to get the optimal benefit of all the different classes of users. By better understanding the different user motivations, and how different users add value to their core offering, they can increase their likelihood of success and strengthen their brand. Open source users contribute through code development, marketing and promotion, and complementary product creation. They also act as idea generation factories to help the company innovate better and build products and services that better meet the needs of their customers.

The biggest challenge in leveraging this untapped resource is changing the mentality of developers in the open source communities who have long enjoyed the center stage. Technology author Chris Pirillo describes the issue succinctly (http ://chris.pirillo.com/users-vs-developers): "What would the world of software be like if the inmates were running the asylum? I'd argue a lot more useful, and a lot more beautiful. But users are usually in the back seat when it comes to the evolution of a utility – from beginning to end. Let me put it to you this way: software is useless if there isn't anybody using it. The world of software is getting larger by the day, and more people are finding new and different ways to improve lives with digital code. Programmers suffer from a miscalculation of a user's wants, needs, and desires. As a power user, I expect better, I expect faster, I expect smarter, I expect more. When I see a new piece of software that holds promise, I call out its shortcomings in the hopes it will be closer to perfection with the next revision."

It is this culture gap that must be overcome to get the most out of the user base of an open source project. Dedicated community managers are one option as they can help focus the energy of the community towards achieving shared goals. They can also help increase intergroup communication, and help the community grow.

Further, the distinction between passive and active participants may be blurry. It is unclear how to best separate classes of participants, as their roles might be circumstance based. Is a code contributor both a developer and a user? Does it matter how much code they contribute? Further research into understanding roles would help quantify the dimensions of contribution.

Traditional consumer marketing metrics can also be used to learn about one's user base. By better understanding the users, companies can create more useful products that better meet user needs. Different types of users behave differently, and it may be possible to encourage them to participate in ways that relate to their interests. For example, innovative users tend to adopt technology more readily, don't mind bugs and crashes as much, and are willing to put in the time to help report errors and suggest improvements. Users who are highly involved with the product are more likely to be able to identify novel uses for the product as they have integrated it more

in their life. Loyal users may not be technically savvy, but will gladly wave the banner of the company, promoting the product far and wide and bringing in new users. There are many other passive participants such as event promoters, designers of complementary products, documentation creators, and financial donors.

By carefully partitioning the participants of an open source community using standardized measures, a keystone company could assign its community management and marketing resources more effectively. It could better leverage the inherent value in the user community, and potentially improve the value creation in the ecosystem.

The Challenge of Assessing Value of Passive Participation

Passive participants add value to an open source ecosystem. The challenge is in assessing that value. What is the dollar amount, on average, that each participant adds to the ecosystem? How does that amount vary based on the type of participation? Is it easier to extract value from an open source ecosystem that has more passive participants? The answer to these and many other related questions is of great interest to companies as it defines their positioning strategy and community management practices. If it were possible to better quantify the value of passive contributions, the model of value creation in open source ecosystems would be strengthened, and would improve the ability of keystone companies to strategically position themselves in the ecosystem.

Summary

Companies that make the mistake of discounting the passive participants in their open source community miss out on a valuable resource. It is time to reshape the classical definitions of roles in open source ecosystems. Passive participants should not be viewed as leeches. They contribute to the ecosystem in many ways other than code. As our understanding of how open source ecosystems work improves, the next challenge is to better quantify the value of passive contributions. By better understanding the value of every eyeball in the open source ecosystem, keystone companies can make better strategic positioning decisions, and create more value in the ecosystem.

Mekki MacAulay is the Principal of OSStrategy.org, a consulting firm that helps companies improve their competitive advantage and strategic positioning in a world embracing open source. Mekki is also the president and founder of MekTek Solutions (http://www.mektek.ca), an IT services company based in Ottawa, ON. Mekki holds undergraduate degrees from Carleton University in Computer Systems Engineering, and Psychology, and a Master's degree in Technology Innovation Management. His research interests focus on open source adoption (http://osstrategy.org/OpenOfficeAdoptionVSMSOffice.pdf); open source ecosystem value creation, extraction, and keystone company positioning; and quantifying the value of passive participation in open source projects.

"NO MATTER WHAT – DO NOT create training from scratch. Really! Before you sit down to create that next manual, quick reference, user's guide, STOP. Throw your question out to your online social network for help and you will be amazed at all of the information that will come your way. These are, after all, information professionals."

http://tinyurl.com/yz28jsu

Training programs for open source software provide a tangible, and sellable, product. A successful training program not only builds revenue, it also adds to the overall body of knowledge available for the open source project. By gathering best practices and taking advantage of the collective expertise within a community, it may be possible for a business to partner with an open source project to build a curriculum that promotes the project and supports the needs of the company's training customers.

This article describes the initial approach used by Canonical (http://www.canonical.com), the commercial sponsor of the Ubuntu Linux operating system, to engage the community in the creation of its training offerings. We then discuss alternate curriculum creation models and some of the conditions that are necessary for successful collaboration between creators of existing documentation and commercial training providers.

First Attempts

One of Canonical's first attempts at engaging the larger Ubuntu community was a joint effort on the Ubuntu Desktop course (http://ubuntu.com/training/desktop). Canonical decided to offer the 7.10 version of the course under a Creative Commons CC-BY-SA-NC license (http://creativecommons.org/licenses/by-nc-sa/3.0). This license allows sharing and remixing of the materials for non-commercial (NC) use as long as the copyright holder is attributed and a share-alike license is used on remixed works. Mark Shuttleworth, founder of Canonical and the Ubuntu Foundation, announced the effort in 2007 on his personal blog (http://www.markshuttleworth.com/archives/134). He began with: "Is it possible to have training materials that are developed in partnership with the community, available under a CC license, AND make those same materials available through formal training providers? We're trying to find out at Canonical with our Ubuntu Desktop Course."

It didn't take long to hear the outcry from many regarding the NC aspect of the materials. Canonical needed to protect their materials, yet wanted to openly engage the experts in the Ubuntu Documentation Team (Doc Team). Many Doc Team and other community members refused to take part because of the selected license. The Ubuntu documentation and wiki sites are all under the less restrictive CC-BY-SA license, meaning that the training materials license disallowed using much of the existing materials.

In the first round of the effort, Ubuntu community members helped develop the course topics. Canonical then hired a training development firm to write the materials and take over 500 screenshots for inclusion in the more than 400 page course book.

For the open source documentation toolchain, Canonical tried to mirror the existing process used by the Doc Team to try to encourage existing team members to contribute.

This also meant that the training vendor had to have their staff learn the toolchain which consists of Ubuntu, Doc Book (http://docbook.org), Bzr (http://bazaar.canonical.com) and Launchpad (https://launchpad.net). As the vendor had no experience with any of those tools, it was quite a learning curve.

Coming from a traditional, proprietary training development background myself, I knew this was not an easy path. I recall trying to make documentation changes using the toolchain and finally gave up. In desperation, I made a long distance call to London to the project lead so that my contributions could be included before the final deadline. When I joined Canonical in September of 2008, one of my first tasks was to update the course to the 8.04 Long Term Support (LTS) version. The results in terms of community involvement were similar to the first effort as there was still much resistance due to the license. In the end, instead of using a vendor, I ended up writing much of the material myself and then hired an external copy editor and Doc Book expert to do the final formatting. What did work well was that within days of release, several teams began translating the materials. The Romanian team was first, followed by the Russian and Portuguese teams.

The upcoming 10.04 LTS version will only be offered as an eLearning version of the course. The eLearning version was developed at the same time as the classroom-based version but is under a standard copyright. At this time, there are no plans to change the license to a fully open, CC-BY-SA. For now, the business decision is for further course development to be under a traditional proprietary model with standard copyright.

Training Models

One of the models I will be speaking about at Linux Conf Australia in January 2010 is a hybrid that takes advantage of a more community-based approach (http://www.lca2010.org.nz/wiki/Miniconfs/Education). From the commercial side, a company needs products to sell, while the community is often more interested in learning and knowledge sharing. Wikis and documentation are great resources but they are not comprehensive training products ready for classroom use. There is no shortage of subject mattter experts (SMEs) in the Ubuntu community but few, if any, have the expertise to develop classroom materials. This is where a hybrid approach can hopefully accommodate both efforts.

In a hybrid model, training documentation should be licensed in a way that is compatible with existing materials. The license should provide course authors with the freedom to use existing materials as well as contribute back new materials that they develop in the process. The only part of the materials that a company would hold full copyright to are the learning exercises, lab activities, reviews, quizzes and course support files. In an ideal classroom setting, these ancillary files and materials are the key to providing a comprehensive learning experience. The community gets better documentation while the company can still provide a unique and valuable product for its customers.

Some will argue that everything should be provided under a fully open license. We argue that the hybrid approach satisfies corporate decision makers who see their intellectual property as the sole value in the training environment.

It also shows the community that the corporation is willing to feed back into the greater documentation efforts. There are considerable financial savings as well. Instead of hiring course developers to create materials from scratch in order to hold full copyright, the community provides an existing source of material and experts in the subject area. Instead of spending time and effort to write technical documentation, the course developer can spend more time in developing quality and authentic classroom exercises.

In many regards, this was the approach we used at NASA. When we developed training materials for the Human Spaceflight program at NASA's Johnson Space Center, we were fortunate to have some very unique training facilities and classrooms. All of the materials for the International Space Station program are in the public domain and available under the U.S. Freedom of Information Act. Yet, unless you have access to those facilities, you cannot replicate the training experience. The same approach could be used by open source companies: focus on the training environment, not the documentation.

One of the most difficult tasks in developing quality training experiences is coming up with authentic lab exercises and case study scenarios. Many people are reluctant to document when they make mistakes but those experiences are what make great learning experiences for others. In my role as a learning consultant, I actively try to engage system administrators to talk about their pain points and problems. How to deal with those issues becomes best practices which can be fed into course development.

For example, during a kick-off meeting for one of our course development groups, one system administrator announced that he was actively working on a hacker attack on one of his systems back at his office. He wasn't exactly thrilled when I told him "that's a great case study!". How he reacted, the troubleshooting process, who and where he went to for help were exactly what we were trying to capture. Too many times a problem is solved and no record of the process is kept. Think about Grace Hopper's famous entry "Relay # 70 Panel F (moth) on relay. First actual case of bug being found" (http://en.wikipedia.org/wiki/File:H96566k.jpg). Document everything!

Two other notable models for developing open source training can be found in the Ubuntu Learning Project (https://wiki.ubuntu.com/Learning) and the Flossmanuals.net project. The Ubuntu Learning Project is an entirely community-based team using the Moodle.org learning management system to develop classroom and online materials. While the project is still gaining momentum, it has decided to offer the materials under the CC-BY-SA license. The server is currently being run by volunteers but they have asked that Canonical point a sub-domain, learn.ubuntu.com, to their offerings. Similar to the Doc Team's efforts, these materials would be available for all to use.

The Flossmanuals.net project offers a wiki-based book authoring system and accompanying book sprint model. The one week book sprint model was successfully used by the OLPC team to develop the OLPC Laptop Users Guide (http://wiki/laptop.org/go/Simplified_user_guide). The guide is available as a free download in either HTML or PDF format and is also available for purchase through the online printer, Lulu.com.

This innovative model brings together SMEs, instructional designers, copy editors and formatting experts into a central location (participants can be both local and remote) to write a complete book or manual in just one week's time. Key contributors are rewarded by having the project sponsor their travel and the project has a complete book in just one week.

Looking Towards the Future

Open source training is an area where great innovation is still possible and greatly needed. Many training development tools are still in the realm of proprietary software vendors and finding course developers who are familiar with open source toolchains is a challenge. As open source becomes even more widely adopted, we look forward to seeing more training developers join open source communities to help grow the models and toolchains available.

Belinda Lopez is a Senior Learning Consultant, currently working as the Training Project Manager for Canonical, the commercial sponsor of the Ubuntu Linux distro. For the past 15 years she has helped create innovative learning solutions for everyone from preschoolers to astronauts. Before moving into the open source world, she was an Instructional Designer and Curriculum Developer in the Human Spaceflight Training program at NASA's Johnson Space Center. Prior to that amazing experience, she worked in the Center for Technology in Teaching and Learning at her alma mater, Rice University. Belinda has been following the Ubuntu Linux project for several years, being drawn into the project by the potential to use Ubuntu in education and learning environments. She has since become active in the Ubuntu Women's Project, the Ubuntu Learning Project and Edubuntu and hopes to encourage others to become more active in the various open source communities.

Q. How can a community be considered "open source" if its primary objective is to promote commercialization?

A. Recently, we saw probably more conversation than any of us wanted about the notion that Eclipse is a trade association and therefore not an open source community (http://www.eweek.com/c/a/Linux-and-Open-Source/Is-Eclipse-an-Open-Source-Community-or-Trade-Association-895397). I believe that perspective to be misguided as it implies those two states are somehow mutually exclusive. They are not. And it is our community's embrace of both that makes Eclipse unique.

The Eclipse Foundation is and always will be a trade association. It is also and always will be an open source community. This duality is built into our bylaws, our organization and, I would assert, our DNA. Consider the following sentence from the first paragraph of our Bylaws: "The purpose of Eclipse Foundation Inc., (the "Eclipse Foundation"), is to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services."

That sentence captures the very essence of the Eclipse Foundation. Our mission is to both move the technology and community forward and to work on its commercialization. The "trade association" of member companies financially support the operations of the Eclipse Foundation. Over 70 of them also provide committers who work on projects. There are relatively few obligations that an Eclipse member company undertakes when they sign the membership agreement, but one of the most important is to create a commercial offering based on Eclipse technologies. It is that obligation which completes the loop from open source to commercialization to trade association and back. Those trade association members are not strangers: they are companies that are intimately involved in and committed to the success of the entire Eclipse community.

There is no doubt that the focus on commercialization places added burdens on Eclipse projects. Our development and intellectual property processes require real work to comply with. But there is value in that labour, and the value is in the added use, adoption, commercialization and plain old respect that the Eclipse brand brings to a project. Not every Eclipse-based open source project needs to be hosted at the Foundation. For some projects, our processes may be too heavyweight. But those projects are still a valuable part of the broader Eclipse ecosystem.

The Eclipse community is also an open development community. I strongly believe that our development process has all of the attributes of openness, transparency and meritocracy that open development requires. Our unique approach to open source development is what enables things like the annual release train, which is arguably the best run, most predictable feat of software engineering on the planet. And let's not forget that although many projects at Eclipse are supported by developers working at member companies, there are also many projects with active participants who are here as individuals.

But there is also no denying that we have our challenges. Every project would love to have more resources and more community involvement. We need to make it easier for newcomers to contribute. There are projects who frankly don't do a great job of welcoming contributions.

We have to attract more resources committed to evolving the core platform. We have a major new release of the platform coming next year. The staff and the Board of the Eclipse Foundation recognize all of these challenges and are working very hard to address them.

The balance between a trade association and an open source community makes Eclipse unique in the software industry. We have always been both, and that has always been an important part of our success. We are different, and in my mind that is a very good thing. I believe that we should all be very proud of the organization that we have created.

This Q&A is based on a blog post that was originally published on the Life at Eclipse blog. You can read the orginal post at http://dev.eclipse.org/blogs/mike/2009/12/11/it%E2%80%99s-a-desert-topping-and-a-floor-wax/.

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.

Knowledge Exchange Briefing Paper on Journal Business Models

Copyright: Knowledge Exchange

From the Introduction:

Over the past few years many studies have been published on the cost and economic benefits of journal business models. Early studies considered only the costs incurred in publishing traditional journals made available for purchase on a subscription or licensing business model. As the open access business model became available, some studies also covered the cost of making research articles available in open access journals. More recent studies have taken a broader perspective, looking at the position of journal publishers in the market and their business models in the context of the economic benefits from research dissemination.

http://www.knowledge-exchange.info/Admin/Public/DWSDownload.aspx?File=%2fFiles%2fFiler%2fdownloads%2fOpen+Access%2fKE_Briefing_Paper_on_Journal_Business_Models.pdf

Technical Guidelines for IT Professionals

Copyright: The National Computing Center

From the Executive Summary:

Free and Open Source Software (OSS), has a long history, but is now transforming both the Software Industry itself and user interest. Historically the remit of the technicians, it is now widely recognised as offering a competitive opportunity compared to many proprietary solutions, and many organisations now maximise its use under both 'free' and commercial business terms. The emergence of Open Source high quality enterprise software has already transformed the leading internet businesses. Equally, many suppliers of enterprise solutions have turned to Open Source for part of their portfolio, or have based their entire (commercial) business model on Open Source.

Organisations failing to examine and assess the opportunity to use this software are missing out on an important resource. Reduction in cost, whilst imperative in the current financial climate, is only one potential benefit, as these Guidelines will discuss.

http://www.theopenlearningcentre.com/ncc-oss-web.pdf

Open-Source Business Intelligence Tools Production Deployments Will Grow Five-Fold through 2012

Copyright: Andreas Bitterer of Gartner

From the Description:

This Gartner report entitled, "Open-Source Business Intelligence Tools Production Deployments Will Grow Five-Fold through 2012" by Andreas Bitterer, presents key findings and recommendations based on analysis of current market traction and adoption trends. Open-source BI tools are not only the default option for cash-strapped organizations, but they are becoming more of a mainstream deployment option for all kinds of BI usage.

http://www.pentaho.com/five_fold_growth/

Modeling a Paradigm Shift: From Producer Innovation to User and Open Collaborative Innovation

Copyright: Carliss Y. Baldwin and Eric von Hippel

From the Executive Summary:

We are in the midst of a major paradigm shift: technological trends are causing a change in the way innovation gets done in advanced market economies. In addition to the model of producer-based design—the idea that most important designs for innovations would originate from producers and be supplied to consumers via goods and services that were for sale—two increasingly important models are innovations by single user firms or individuals, and open collaborative innovation projects. Each of these three models represents a different way to organize human effort and investments aimed at generating valuable new innovations. HBS professor Carliss Y. Baldwin and MIT Sloan School of Management professor Eric von Hippel analyze the three models in terms of their technological properties, specifically their design costs and architectures, and their communication requirements. The researchers argue that as design and communication costs decline, single user and open collaborative innovation models will be viable for a steadily wider range of design. These two models will present an increasing challenge to the traditional paradigm of producer-based design—but, when open, they are good for social welfare and should be encouraged by policymakers.

http://hbswk.hbs.edu/item/6325.html

January 12

Innovation Night

Hamilton, ON

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

http://innovationnight.ca/

January 14

How to Identify, Record, and Submit an Eligible SR&ED Project

Ottawa, ON

Are you working on new, improved, or technologically advanced products or processes? Did you know that your project could be eligible to access the largest source of refundable tax credits in Canada? The SR&ED tax incentive program sounds complicated and without the basic knowledge and skills it CAN be. We want to make it easy for you to gain access to a piece of the 4 billion dollar pie!

http://sred-tlc.eventbrite.com/

January 15-16

EpCon

Waterloo, ON

One of Canada's largest student technology conferences of 2010 is coming to Waterloo, Ontario on January 15-16th. EpCon will bring together North America's leading tech gurus and 300 student tech enthusiasts from schools across Canada to discuss innovation and development of various technologies.

http://epcon.epictech.org/

January 19

Silicon Halton Meetup

Oakville, ON

Silicon Halton is a group dedicated to connecting and creating strong/local relationships for hi tech entrepreneurs and leaders in the Halton region. The short term goal is to build a strong and engaged network to help us all realize how important and vibrant the hi tech community is in our neighbourhood.

http://events.linkedin.com/Silicon-Halton-Meetup-3/pub/183521

January 26-30

Vancouver Innovation Camp

Vancouver, BC

Why are ideas like measles, waffles and mirrors? Why do successful people generally fail more often than less successful ones? At Vancouver Innovation Camp, you'll discover how answers to these questions are related to learning entrepreneurship and innovation skills.

http://www.innovationcamp.org/

January 27

Net Gain 4.0

Toronto, ON

While our attention was focused on the creation of online panels and conducting online surveys, social media has rapidly spread throughout the internet cloud. This trend cannot be overlooked by market researchers. With blogs, Twitter, LinkedIn, Facebook, YouTube and even Flickr, how viable are these new media forms as valid data points from which to collect and analyse data? Are market researchers becoming outdated as analysts in the internet cloud? Are we just voyeurs that monitor and watch unstructured conversations in 140 characters or less? Will Google Wave and MS Looking Glass form the core of market research? Join us as we search for answers.

http://www.mria-arim.ca/NetGain4

February 9-10

Reboot

Victoria, BC

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

http://www.rebootconference.com/privacy2010/

FreeBSD, the OS choice for many…

FreeBSD is a robust and scalable operating system, ideally suited for servers, embedded devices, security products, and desktops. Its high performance, stability, security, standards compliance, and trouble-free system administration makes it the OS of choice for many ISPs, universities, network appliance vendors, and enterprise corporations. FreeBSD makes it easy to browse and install from its repository of over 20,000 no-cost applications. Customizing FreeBSD to your specific needs is a snap!

FreeBSD is Free.

That's right. Free. As in no license fees, no cost per seat, no upgrade cost, no development kit costs.

What if there was a non-profit foundation that supported the development of FreeBSD for the benefit of the entire FreeBSD community?

There is.

The FreeBSD Foundation is a non-profit organization that is committed to supporting and building the FreeBSD Project and community worldwide. The Foundation board is comprised primarily of volunteers that are actively involved in FreeBSD as well as respected researchers in their areas of expertise.

The FreeBSD Foundation is constantly working behind the scenes, supporting the FreeBSD Project and community in three major areas: developer communication, handling legal issues, and funding development projects.

We help fund and coordinate developer summits, sponsor BSD related conferences, and promote FreeBSD at technical conferences. We promote the FreeBSD operating system and publish press releases in concert with the FreeBSD Project marketing team.

We represent the FreeBSD Project in the execution of legal contracts, license agreements, copyrights, trademarks, and other legal arrangements that require a recognized legal entity.

We also fund development projects to support emerging technologies such as solid state disks, USB 3.0, machine and network virtualization, highly parallel processors, clustering, and data replication.

The FreeBSD Foundation is a 501(c)(3) non-profit organization that is committed to supporting and building the FreeBSD Project and community worldwide. Founded in March 2000 to fill the need for an outside organization that could support the community's vision and growth, The FreeBSD Foundation exists to serve the FreeBSD community world wide.

The Foundation board is comprised primarily of volunteers that are actively involved in FreeBSD and are respected researchers in their areas of professisonal expertise.

The Foundation's fund-raising efforts are essential to keeping FreeBSD free.

The cost of keeping FreeBSD free is growing all the time. Private donations fund 100% of the FreeBSD Foundation's efforts.

Join the growing list of donors and users of FreeBSD
P.O. Box 20247, Boulder, CO 80308, USA
Phone: +1-720-207-5142, Fax +1-720-222-2350
Web: **www.freebsdfoundation.org**


MASSIVE INNOVATION
Next session starts February 2
Apply at www.LeadtoWin.ca
Lead to Win
Exciting new businesses that are in Lead to Win now!
selectstart
CheckinWiz
media
catwalk networks
SCTi
EclipseSource
Castel RESEARCH
zeebumobile
Videotelephony Inc.
NOVABRAIN TECHNOLOGIES Inc.
data bridge
EnTeraSec security
polynate
Accentway
EvenuSix
Triple Through Sustainability
SMART ROTOR SYSTEMS
SiliconPly
cornerportal
MARKTEK
Clearview
NewEraHealth
MarkOff Security

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

Febuary 2010:	Bootstrapping Startups
March 2010:	Mobile
April 2010:	Cloud Services
May 2010:	Consulting
June 2010:	Niche Markets

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.

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.

Coral CEA is a member-based company whose mission is to assist companies of all sizes with the commercialization of communications-enabled applications (CEA). We are creating and anchoring a business ecosystem that leverages a unique, technical platform that provides advanced ICT building blocks to members. Visit http://www.coralcea.ca to become a member.