olliance | group

Understanding and Engaging with the Open Source Community

June 4, 2003
R1.V5

Andrew Aitken
Managing Partner
Olliance Group, LLC
andrew@olliancegroup.com

Jim Negrette
Senior Consultant
Olliance Group, LLC
jim@olliancegroup.com

Robert Munro
Senior Consultant
Olliance Group, LLC
robert@olliancegroup.com

Abstract

The Open Source community-based software development model provides industry, government, and academia with significant opportunities to leverage IT investments and increase organizational effectiveness and efficiency by using and continually improving high quality Open Source software. The Open Source community ecosystem provides important software, services and support components that span virtually all commercial markets as well as governmental and educational domains. While Linux, Apache, and SendMail are well known examples of Open Source software, these solutions constitute only a very small fraction of the Open Source software available. Open Source software solutions exist for virtually all types of business applications. Open Source solutions integrating these and other business functions into powerful applications like ERP and CRM suites are also available. Additionally, Open Source software exists for a variety of vertical markets and is being widely used. These solutions, as well as many more, are available at various levels of functionality, stability, and both commercial and non-commercial support. There are many advantages to the adoption of Open Source software. The extent to which a given enterprise benefits from these advantages depends in large measure upon how they engage with the Open Source community. This paper briefly outlines the benefits of Open Source software and development processes. It also describes the rationale for, and benefits of, engaging with the Open Source community. Finally a high level process is suggested for deciding to deploy Open Source and how to engage with the community.

Table of Contents

What is Open Source Software?

Open Source software is characterized by its licenses, such that any user who modifies the software and then redistributes it either must or may make available to those who receive it the source code that defines the software and their any modifications. A number of Open Source software licenses exist in a continuum of legal rights and obligations ranging from the Berkeley Software Distribution (BSD) license, which permits but does not require providing source code, to the General Public License (GPL), which mandates source code availability for all users.

Open Source software has the following characteristics, some of which are not usually found in legacy commercial software:

- Control resides with the user
- Highly stable
- Proven security
- End-User input to evolving functionality
- Excellent quality
- Highly flexible
- No or reduced license fees
- No vendor lock-in
- Self-determined upgrade path
- Can run on less expensive hardware
- Very cost-effective
- Freedom of vendor choice
- Fast development cycles
- Ongoing evolution.

Open Source software can meet the software requirements of mainstream enterprises including financial services, retail, manufacturing, distribution, and transportation, and healthcare, entertainment, academia and all branches of government.

What is the Open Source Community?

In the broadest sense, the Open Source community includes everyone who uses Open Source software, worldwide. However, many users of Open Source software are passive consumers, that is, they never ask questions on mailing lists, participate in online discussions, post bug reports, fix software problems, or develop and contribute software source code. For the purposes of this discussion, we will use the following definition: *the Open Source community consists of those who participate in developing and improving Open Source software*.

The term "Open Source community" sounds rather monolithic, doesn't it? Actually, that's not the case at all. Instead, "Open Source community" is a term that refers to a great number of separate and individual groups of self-selected Open Source software contributors and involved users organized for self interests.

Self-Organizing

An individual Open Source community is characterized as a loose association of people who have a common interest in developing and using a particular piece of software. These people are typically geographically dispersed, communicating via email and various Open Source development tools in order to collaborate on the software that they are jointly developing, testing, using, and improving. Each loosely organized group – an Open Source community of developers and users – designs, codes, tests, documents, reviews, and uses all parts of the software being developed. Decisions are typically made by consensus,[1] with leadership exercised by those recognized as more committed, knowledgeable, and competent by the rest of the community. Such communities can have as few as two or as many as several thousand individuals contributing to the overall software development lifecycle – the more, the merrier.

Gift Economies

Those who contribute to an Open Source community typically are not compensated monetarily within the community for their work. Members join the community because they have an intellectual interest in

[1] The mode of consensus practiced within Open Source communities that facilitates decision-making is based upon objective criteria. The maturity of a software product is perceived in the resolution of serious bug reports and dwindling of minor ones. A software release is proposed, and lack of vocal objections is taken as general assent.

the software or because they have an association with some other organization that is interested in or actively using the software. Many Open Source community members are paid for their work directly by those enterprises that employ them, and their contributions to the community are simply made in the course of their normal employment. Intangible status and rewards within an Open Source community invariably come as a result of contributing significantly to the software developed by the community. In this way, Open Source communities function as "Gift Economies" with the most recognition given to those individuals who contribute most substantially and consistently to the project success and progress.

Self-Selection

Members of the Open Source community typically choose for themselves what piece(s) of programming or other development tasks they will work on. Typically one or more senior members of the community will organize a prioritized list of things that need to be done, but it is up to the individual members to choose their own self-assignments. It is considered good form to work on tasks from the prioritized list; consequently the recognition is greater for those who cooperate with the agreed-upon priorities. Most Open Source communities are fairly cooperative, but members are free to contribute however they wish.

Members of an Open Source community can take different roles, depending on their interests and skills. There is typically a cadre of core developers who write the most formidable constructs and algorithms. A larger peripheral group of less committed developers often exists that contributes interfaces, subroutines, device drivers, and the like. Some members may focus their efforts on functional or performance testing, while others might write documentation. Perhaps the largest group is involved users who contribute bug reports and feature requests. Everyone involved is free to make comments, argue for one approach over another, and generally discuss the direction and technical details of the software under construction. Peer recognition, honest and direct criticism, and the weight of one's own reputation among other members of the community combine to make these Open Source communities self-regulating.

Meritocracy

Because the members of Open Source communities are usually geographically dispersed, communicate almost exclusively by email and other Internet enabled electronic means, and typically haven't ever met each other face-to-face, the socio-economic hierarchies of the physical world don't apply in cyberspace.[2]

Thus, corporate position, economic status, and formal education don't interfere with the meritocracy of Open Source communities. The factors that matter in Open Source development projects are raw coding talent, incisive logic, hard work, and attention to detail. Most programming members of Open Source communities collegially compete with each other through their ideas, arguments, and especially their functioning software submissions. Concepts and solutions that are clearly superior win out through the consensus of critical, multi-level peer review and rigorous testing in a wide variety of different software and platform environments. Over time, those developers who consistently devise more elegant, complete, and better performing solutions gain well deserved peer recognition, stature within the project, and more responsibility in their own assignments as well as for reviewing the work of others. Since a very talented software developer can be substantially more productive and efficient than the average developer, this might have a lot to do with why Open Source communities can be more effective than proprietary software corporations and most independent in-house software development organizations.

Therefore, any teenager with a personal computer and an Internet connection can compete – on a level playing field – with seasoned systems development professionals in academia and industry. If she can write a more elegant widget implementation subroutine, or better grasp the Zen of task scheduling or virtual memory management, they should accept her patch, and the project will be that much better for it.

The dispersed, Internet connected nature of Open Source communities also fosters another characteristic of Open Source development: vigorous criticism. It's well known that email communications often lack the tact of personal interactions. Woe unto any Open Source developer who submits a badly formatted and gratuitously complicated, spaghetti-code module or patch to a programming gatekeeper on an Open Source project. Such an unwise would-be contributor is likely to get a rigorous emailed whipping that

[2] A famous *New Yorker* cartoon illustrates this perfectly. It shows a dog sitting in front of a personal computer with his paws on the keyboard. The caption is: "On the Internet, no one knows you're a dog."

they won't soon forget! Corporate programming managers cannot get away with such enthusiastic correction of subordinates in modern office environments with strictly enforced human resource policies. But Open Source contributors are all volunteers, so they either work up to the standards of their peers, or drop out, and someone more able takes their place. Open Source communities are very capable of enforcing high standards among their contributors, either quickly training or losing any poor performers.

Fast Development

Most Open Source communities tend to evolve and improve major software products on much faster cycles than commercial software development organizations. A new release of IBM's MVS or AIX, HP's HP-UX, Sun's Solaris, or Microsoft's latest flagship PC operating system is produced about every three to five years. In contrast, a new and significantly improved release of the Open Source Linux production kernel is produced every 18 to 24 months. Similarly blistering paces of software development have been observed with the Apache web-server as well as the KDE and Gnome desktop environments for Linux. It is simply the case that commercial software development processes can't match the pace of Open Source.

Quality Ensurance

Open Source communities typically follow a philosophy of "release early, release often" with daily build releases of the software within the community not uncommon. External releases are made on a schedule dictated by the relative maturity, level of bug-free quality, and functional stability of the software. Members of the community continually review the current build level of the software as well as designs, testing scaffolds, test results, documentation and other information contributed within the community.

Another piece of Open Source philosophy is characterized as "many eyes make all bugs shallow." The continual review process used by Open Source communities produces a "many eyes" effect of massively parallel peer review that has been demonstrated to produce very high quality oversight of the software development process and products. Constant, repetitive peer review, coupled with a release schedule tied to objective software quality rather than marketing deadlines, consistently results in Open Source software quality orders of magnitude higher than that of commercial releases of similar software.[3]

Open Source software is constructed primarily by those who want to use it, so they don't release it until they're satisfied that it's usable by the majority of the user community involved, that is, when it's done.[4]

Triumph of the Commons

Software produced by an Open Source community is released free of license fees under one of a number of Open Source licenses. The common elements of these licenses are that: (a) no license fees are charged and (b) all source code defining the software is made available without charge to users of the software.

In the cautionary tale known as "The Tragedy of the Commons," individual herd owners each attempt to maximize their individual returns from use of a common grazing land. As each herder grazes more beasts, the Commons quickly becomes overgrazed and falls into ruin. Open Source communities turn this story on its head, resulting in much greater returns for everyone involved than they could ever gain separately.

---

[3] Published studies conducted by Reasoning, Inc. have established just such software quality differentials between Open Source software and proprietary commercial software, specifically regarding several TCP/IP implementations.

[4] Commercial software organizations usually have "Quality Assurance" departments that typically oversee internal conformance with a set of rather ponderous and expensive software development lifecycle procedures.

But "to assure" means to encourage confidence, not necessarily to guarantee anything about completeness, correctness, accuracy, ease-of-use, security, stability, reliability, or performance. "Quality Assurance" in commercial software development can be a show of due diligence to satisfy corporate legal departments rather than actual focusing on producing software quality.

In contrast, "to ensure" means to make certain, so it's appropriate and borne out by actual results, to say that the rapid collaborative developmental practices of the Open Source communities result in effective *Quality Ensurance*.

Each member of an Open Source community contributes as much as they can afford in terms of time and effort, and they all gain the benefits of everyone's combined contributions equally in the software results. This powerful force multiplier of Open Source communities is seen as "The Triumph of the Commons."

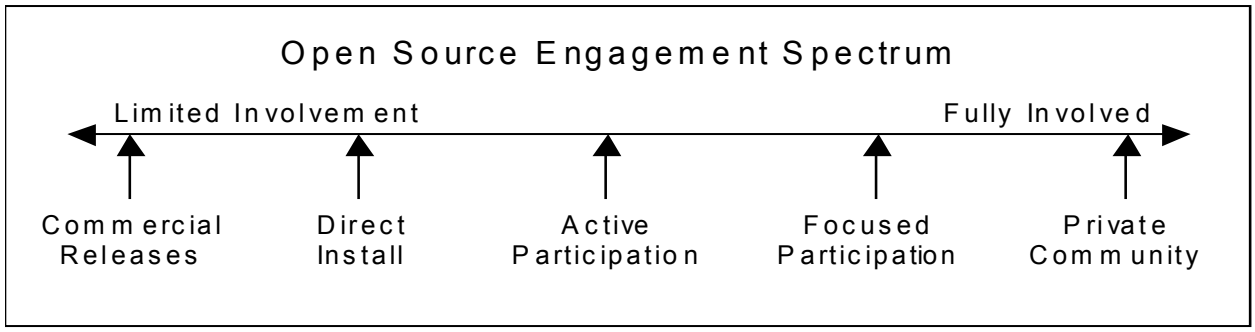
Engaging with the Open Source Community

Deciding how to engage with an Open Source community requires understanding the needs of your organization, the spectrum of possible engagement levels, your desired degree of influence and long-term outcomes, and how a position on that spectrum of engagement matches your enterprise resources and objectives. All points along the engagement spectrum allow the enterprise to make use of Open Source software and benefit from the advantages of the Open Source software development model. The key characteristic defining where an enterprise operates on the spectrum of engagement with an Open Source community is the level that the enterprise and its resources become directly involved and interact with that Open Source community.

Engagement with an Open Source community is a continuum, but an individual or organization can participate on multiple levels simultaneously or become more or less active as their needs dictate. For the sake of illustration, we define five points along the Open Source engagement spectrum:

- Commercial Release – Using software from a vendor that packages and supports Open Source software.
- Direct Install – Deploying the Open Source software directly from online repositories.
- Active Participation – Testing, filing bug reports, developing code components, and debugging problems within a chosen Open Source community.
- Focused Participation – Contributing full time development resources to work in and potentially lead an Open Source community.
- Private Community – Setting up an Open Source community to meet the needs of a user group that needs to be disconnected from the mainstream.

Again the key variable that determines where each of these points falls along the engagement spectrum is the level of involvement of the individual participant or enterprise within the Open Source Community.



Commercial Release

At this level of engagement an enterprise will purchase a commercial release of Open Source software. This means that there is a company that has built a business around packaging and supporting that piece of Open Source software. A good example would be the various commercial releases of Linux such as Red Hat, SuSE, or Mandrake. Thus engaged, an enterprise will experience the advantages of no license fees, high quality, and reduced vendor control. Under this scenario, the company will not be directly working within the Open Source community and as such will not benefit from the advantages of participation such as being able to influence the direction of the software. On the other hand, at this level of involvement the enterprise's experience of the software will be very similar to its experience of more familiar commercial software. This can be an appropriate initial point of entry for an organization that

has little or no experience with Open Source. Typically, commercially supported Open Source software will include simplified or single-step installers and technical support options that look and feel very similar to those available for commercial software. Participation at this level is characterized as being limited to the Open Source software that is commercially supported, with little if any interaction with the developing Open Source community.[5]

Direct Install

At the Direct Install level of engagement, the enterprise might elect to install and configure a somewhat more demanding Linux distribution, such as Debian or Gentoo, and might go to the various repositories of Open Source software such as FreshMeat and SourceForge to choose from the myriad of applications available. Open Source communities associated with these applications make software available at completeness and quality levels ranging from very immature pre-Alpha releases to very stable and reliable Production and Mature releases. When choosing the maturity level of a Linux distribution or software application, the enterprise is trading confidence of stability for having the latest functionality. Obviously the more mature releases will be more reliable. However a given enterprise may be willing to put up with a certain level of instability in order to gain access to a highly attractive but relatively new functionality. Additionally, firms in particular vertical industries such as financial services or distribution might be willing to trade off some stability for advanced features that will give them a competitive edge.

Almost all Open Source communities provide some form of technical support. In most cases this support will be more formalized in those more mature projects that are associated with larger, established communities. In some cases these communities even provide commercial technical support, but at least email support from the developers in the community is usually available either directly or via mail-lists, newsgroups, online forums, or even Internet relay chat (irc) channels.

At this level of engagement the enterprise will gain many of the advantages of using commercial releases but will have substantially more choices available. By being conservative, an organization can choose only those releases that are at a mature release level and enjoy a level of quality similar to commercial software releases. The organization will also enjoy a significantly expanded range of software choices. Using the advantage of being able to choose from the large number of applications available, it is possible for an organization to build much of its IT application stack entirely with Open Source software

Active Participation

Enterprises can engage in active participation in an Open Source community at several levels. Helpful comments, thoughtful feature requests, and documented problems or bugs are always welcome to the developers in any given Open Source community. Active testing of new releases provides the next level of involvement. Direct involvement in software source code development would be the subsequent level. The main advantage an enterprise can realize from engagement at this level is access to, and influence on, the latest functionality in a future application release. Obviously the more active level of involvement in the source code development effort an enterprise has, the more influence it can exert on the software.

By engaging in active participation within an Open Source community, an enterprise can gain important functionality in a key piece of software. Using the advantage of being able to choose from the large number of applications available, it is possible for an organization to build much of its IT applications portfolio from Open Source software, potentially adding functionality that is important to the enterprise, which it then either might or might not contribute back into the community.[6] Where the organization has

[5] This does not mean, however, that organizations using commercial Open Source software can afford to ignore critical security patches, any more than they can ignore similar advisories from commercial software vendors. If an enterprise is using a particular security protocol or service that incurs a security patch, it should consider installing it. An advantage of Open Source software, though, is that security vulnerabilities are typically obscure and much less likely to be exploited than the frequent, catastrophic security holes in some dominant commercial software products.

[6] A common provision of many if not most Open Source software licenses is that, if any user modifies the software and subsequently distributes it, they must make available all of the modified source code (or the original source code and their changes) to every other user to whom they distribute the software. But conversely, under such licenses the user is under no obligation to distribute their source code if they don't distribute the software to others. Most Open

special requirements for a particular software application, it can use its active involvement within that application's Open Source community to gain functionality that meets those special requirements.

Typically an enterprise would actively participate by having one or more of its employees directly participate in the Open Source communities of the applications that the enterprise considers critical to its business. These employees, typically software developers, would work directly with the source code that provides the key functionality as well as the more general capabilities of the application. This active participation has the additional advantage of enabling increased understanding of the key applications by the enterprise's employees. Often IT employees have a merely superficial understanding of important software provided by a commercial software vendor. Working directly with the source code of an important application can give employees a much better understanding of the software, which in turn enables them to provide much better support for that application within the enterprise.

Focused Participation

If a given enterprise has an application or piece of software that is absolutely critical to its success, that enterprise may choose to join or form an Open Source community organized around that software. This assumes, of course, that the application critical to the enterprise has relevancy to others outside the enterprise. In this case employees of the enterprise might fill roles directing the functioning of the community. As long as the spirit of Open Source cooperation is maintained, and the desires of the rest of the community are met, the enterprise will be able to build a robust and vibrant community and direct or strongly influence development of the software. Of course care must be maintained so that the needs of the enterprise do not overwhelm the needs of the rest of the Open Source community. If that were to happen it would be unlikely that others would continue to participate, leaving the enterprise with a community of only its employees. Such a disintegration of an Open Source community would sacrifice most of the real advantages to be gained by participating in that community.

One of the Open Source strategies currently being experimented with involves similar firms within a particular vertical industry such as healthcare, banding together to create communities around common core solutions. This can enable such firms sharing similar needs to support some standard commodity applications particular to and common throughout their vertical industry segment. Furthermore, such cooperating, yet possibly also competing firms, might focus their own internal efforts on unique in-house applications and value-added extensions and enhancements to the common core industry applications that can generate competitive advantages.

Private Community

Occasionally there might be the need to contain the development of a particular piece of software to a tightly controlled community. Some government or sensitive science development efforts could fall into this category. Obviously this development could be done commercially. However in these cases there is often a community of interested parties such as government agencies or scientific labs that want or need to be directly involved with the development of the desired software. In these cases, forming an Open Source community of these interested parties for the development of this software can provide all the advantages of Open Source development while still maintaining a secure environment. In this case this private community would use the collaborative development processes that are endemic to Open Source.[7]

Source participants contribute their modifications back into the community, if only for the practical reason that it enables them to avoid subsequent retrofits. However, no Open Source license extracts such disclosures if the user refrains from distributing the software to others. Such a license provision would be impractical to enforce.

[7] In a previous footnote, we explained that it's the distribution of Open Source software that triggers its requirement to provide software source code. A corollary of this is that one is only obligated to make the source code available to others to whom one has distributed the software. Therefore, Open Source is not antithetical to the security of closed communities of software developers and users. Thus we arrive at the contemplation of an apparent oxymoron in "private Open Source" software communities. Once one understands how Open Source licenses actually work, this makes perfect sense.

This community-developed software system would benefit from the "many eyes" peer review of this development process. The problems of accountability and lack of information sharing associated with commercial contractors could be minimized, greatly improving the efficiency of the development effort.

Software development resources might still need to be acquired for large projects. The major difference would be that these resources would be hired by the interested entities making up the community, and their employment costs could be spread out among the sponsoring members of the community. These resources would be capable of representing the requirements and needs of their employing entities to the community. Getting the software system the community wants and actually needs would be much more likely with a closed Open Source community developing the software rather than commercial developers.

Deciding to Engage

An enterprise considering engaging with Open Source communities will typically need to evaluate its requirements, enabling factors, constraints, and objectives in the following areas:

- Need for simplicity or toleration of complexity in installation, usability, and technical support
- Willingness to grow the enterprise IT organization into one that understands Open Source
- IT staff and end-user sophistication for dealing with unfamiliar and immature software
- Availability of appropriate resources to contribute to an Open Source community
- Need for "basic" versus "cutting edge" capabilities and functionalities
- Economic considerations and resource constraints
- Understanding of the enterprise risk profile.

There are substantial benefits available to an enterprise that adopts Open Source software. The most significant and long-term of these benefits result from engaging with the Open Source communities that produce the software. Those enterprises wishing to take full advantage of Open Source software will more fully engage. Those unable or unwilling to invest the resources or make the changes needed to fully engage will still benefit from using Open Source.

The first step in approaching engagement with Open Source communities is obviously adopting the use of Open Source software. The conclusion of this paper addresses a process to help enterprises make this important decision. This first decision is not a foregone conclusion, because different enterprises and IT organizations have differing objectives, resources, capabilities, enabling factors, and business constraints.

Conclusion

In this paper we've reviewed the definition and advantages of Open Source software, the characteristics of Open Source communities, advantages that can accrue to those enterprises and IT organizations that adopt Open Source software and possibly engage with the Open Source community. We've also looked at the factors that an organization needs to consider in contemplating the adoption of Open Source software and possibly engaging with the Open Source community. In conclusion, we offer a high level process for approaching the decisions that must be reached and acted upon to gain the opportunities and benefits of Open Source.

High Level Process

The process of deciding whether or not to employ Open Source software, then possibly planning for and adopting it follows these high-level milestones:

- ❑ Review enterprise IT objectives, resources, constraints, and enabling factors relative to Open Source.
- ❑ Assess existing IT systems and functions to see where they meet objectives and where gaps exist.
- ❑ Identify areas of business process complexity or high IT costs out of proportion to enterprise value.

- Look for appropriate opportunities for Open Source software and methodologies to help meet enterprise IT objectives.
- Estimate costs and projected returns of migrating specific platforms and functions to Open Source.
- Perform initial financial Return On Investment (ROI) and Internal Rate of Return (IRR) analyses to evaluate the feasibility and financial impact on enterprise IT budgets from Open Source migrations.
- Optionally, develop Total Cost of Ownership (TCO) projections for both status quo and Open Source transitions.
- Obtain executive management approval and sponsorship for planning Open Source initiatives.
- Create an Open Source Steering Committee ideally composed of executives, technologists, and line of business representatives.
- Identify key executives and technologists to lead Open Source efforts, and obtain training for them in Open Source concepts, resources and methods.
- Plan a series of projects to integrate Open Source software and methods into the IT infrastructure. (It is suggested that organizations start with simple and robust Open Source applications such as file/print or web servers before moving to integrate more complex applications like databases, ERP systems or CRM suites.)
- Initiate research and communication with the Open Source community.
- Develop an Open Source Engagement Plan including guidelines for using Open Source software and engaging with the community.
- Refine and confirm the preliminary ROI and IRR analyses of financial returns from Open Source; optionally, review the initial TCO projections; assess risks and returns, and make recommendations.
- Gain executive approval and sponsorship for conducting one or more Open Source migrations.
- Execute Open Source migrations and review the results in a progressive sequence of project cycles.

Ideally, this process will be followed sequentially. Depending upon the size of the organization, corporate culture, resources, and other factors, a simpler, more straightforward process can be implemented as well, as long as a few key steps are followed: performing financial and technical analysis, obtaining executive approval, and developing and adhering to an Open Source community engagement plan.

About Olliance Group

Olliance Group helps clients achieve a competitive advantage and economic gain by leveraging Open Source technologies. Olliance Group is an independent, Open Source strategy-consulting firm, offering a comprehensive set of education, business strategy, and technology planning services.

We are focused on helping enterprises plan and execute successful transitions to Open Source software and methodologies and engage productively with the Open Source community.