Electronic Communications of the EASST
Volume 20 (2009)

Proceedings of the
Third International Workshop on
Foundations and Techniques for
Open Source Software Certification
(OpenCert 2009)

Understanding how OSS Development Models can influence assessment methods

Richard Taylor

17 pages

Guest Editors: Luis Barbosa, Antonio Cerone, Siraj Shaikh
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Understanding how OSS Development Models can influence assessment methods

Richard Taylor[1]

[1] rjtaylor@qinetiq.com, http://www.qinetiq.com/
QinetiQ Ltd.

Abstract: One of the most important aspects of OSS that distinguishes it from COTS is both the variety and specific characteristics of the development models used. Understanding these development models will be critical to the effective design of assessment approaches. This paper documents the more common development models used by OSS projects and explores the complex landscape of stakeholders that these models expose.

Keywords: ECEASST, Open Source Software, Assessment Methods

1 Introduction

One of the most important aspects of OSS that distinguishes it from COTS is both the variety and specific characteristics of the development models used. Understanding these development models will be critical to the effective design of assessment approaches. This paper the more common development models used by OSS projects and explores the complex landscape of stakeholders that these models expose.

For an OSS project to be successful, it must attract developers and other contributors. This paper also explores what are the common factors in successful OSS projects that attract and retain contributors.

The open OSS development environments provide many options for innovative assessment metrics. A description of this ’anatomy of OSS projects’ is presented as a vehicle to explore the tools and techniques that successful OSS projects employ.

2 Comparison to COTS

COTS development models are opaque to most of the stakeholders in the COTS market place. Most software vendors closely guard the details of how they organise their software development activities. In contrast, OSS development is conducted in the open. This disparity provides an advantage for understanding OSS development. There are likely to be very many different internal organisational patterns for COTS developments. These development patterns can affect the through-life properties of the applications that they produce. When assessing the COTS applications that are proposed for a system it would be useful to be able to assess the characteristics of the development model that was used in their development. However, in many cases it is not possible to include the development model of a COTS application in its assessment because of the secrecy that the closed development model entails. For OSS, the ability to explore the different approaches to development and the routes by which OSS projects find their way into widespread use may help to achieve an accurate assessment.

COTS development models rarely involve collaborative software development between multiple vendors (there are exceptions to this but they are unusual). In contrast, OSS development almost always involves collaborative software development. Distributed, collaborative, development requires specific architectural approaches that have an impact on the organisation and processes that OSS projects use. Assessment approaches may be able to exploit these aspects of OSS projects to gain more insight into the projects characteristics than would be possible with closed development models.

3 Different Models

OSS projects emerge and develop in many different ways. Understanding these different patterns of project evolution can help to explain some of the differing properties that these projects exhibit. The following list characterises a number of lifecycle models that can be observed. The list is not exhaustive but it does provide a reasonable coverage:

- *Over the fence*. These projects start their life as conventional COTS developments within software companies. At some point (often after a period of normal software sales), the company decides that it is no longer in their interest to continue with development of the application or that the ongoing development would be better served by a different business model. At this point the application license is changed to an OSS license; the source code is released and the company walks away (sometimes the company will continue with some low level support for the code base). If there is sufficient interest in the application, a new OSS development community can form to continue its development. Examples of OSS projects that have started in this way include Firebird (this was originally the InterBase application developed by Borland) and Firefox (which can trace its development back to the Netscape web suite that was released as OSS by the Netscape Corporation).

- *Corporate R&D*. These projects are initially developed by the R&D departments of large corporations. At some point, the company has to decide whether to stop work on the research work or to move to full production. In recent years, some companies have chosen OSS as a third path for some of their research prototypes. This has been typified by IBM. Examples of this are the Eclipse rich desktop development platform that started life as an IBM R&D project and Derby, an embedded Java database that was an IBM product called Cloudscape.

- *Escape from academia*. Many OSS projects begin life in academia or some other government sponsored research organisation. The motivations for releasing these projects as OSS are varied. Often it is because there is no obvious commercial path to exploiting the application. Other reasons include: a researchers wish to continue the development of a project after a research grant has completed; a desire to collaborate with other researchers; or a desire to make results available to a wide community. There are a huge number of projects that started in this way, notable examples are: PostgreSQL, a enterprise class relational database system that is based on work started at the University of California and the BSD family of UNIX Operating Systems. These include FreeBSD, OpenBSD and NetBSD. The BSD family all derive from the Berkeley Software Distribution (BSD), an early version of UNIX that was released as OSS by the University of California.

- *From little acorns*. These projects often start as small hobbies for a single developer. He (and it almost always is he) releases his work under an OSS license in the hope that it might be useful to someone else. Over time a group of likeminded people gather around the project and it starts to gain more users. Sometimes these projects can grow into very large initiatives. The most widely know example of this type of OSS project is the Linux Kernel. The Linux Kernel was started as a hobby by Linus Torvalds in April 1991. In August of the same year he posted a message to an Internet news group explaining what he was working on. Within a very few years the Linux Kernel had become one of the three most popular Operating Systems.

- *Community Power*. Occasionally projects are formed by an existing community to develop a particular application. Sometimes this is because someone has an idea that needs a large team to realise it, so they set out to create the team of volunteers from the start. Sometimes it is because the wider OSS community feels that a particular problem needs to be solved and there is sufficient will to form a large project to tackle it. A good example of this later motivation is the Mono project. The Mono project was formed to provide an implementation of the Microsoft .NET platform on the Linux Operating System. .NET was seen as a threat to the continued growth of Linux because applications that use .NET could not be used on Linux. The Mono project was specifically formed to address this problem.

- *Start small stay small*. The majority of OSS projects start as small initiatives by a small group of developers that share a common interest. The applications are small enough that they can be successfully developed and maintained by this small team. These teams might comprise as many as 100 contributors or as few as just a lone developer. Typically, they will be around 10 core developers. These projects will often continue to be developed at this level of team size for many years.

- *Community release*. Some software companies are releasing their source code under an OSS license as a means to encourage a community of developers. There are many reasons why this approach might be adopted. A common approach is for the company to retain copyright ownership of the source code (or at least maintain rights to re-license any contributed code). This enables the company to release close source versions that can be charged for along side the OSS ”community” releases. Examples of this model are widespread including Sun Microsystems OpenOffice.org.

The organisation of complex software developments is a challenging problem. OSS projects are faced with unique issues: internationally distributed developers; complex IPR frameworks; extremely tight fiscal constraints; competing developer motivations and user expectations; etc. It is remarkable that any OSS projects are able to thrive at all. As OSS projects rely so heavily on volunteer effort (even though many contributors are paid by an employer to work on a project, the project as no mechanism with which to compel them), those involved must be content with the structure of the decision making process, otherwise they would not continue to contribute. There are almost as many different decision making structures as there are OSS projects but a number of common organisational patterns can be observed:

- *Dictatorship*. Some projects are controlled by a single individual. This is common among the smallest of OSS projects but it is also a pattern that works in some of the very largest of projects as well. The Linux Kernel is a good example: there are discussion forum for each of the sub-systems of the kernel and any changes must be agreed through consensus on these lists first, then there are a small number of senior developers that must be convinced of the merit of the change. However, in the end the final decision lies with Linus Torvalds. Dictatorship projects only thrive when the leader is recognised by all the developers as the main technical lead of the project.

- *Cabal*. Probably the most common structure is characterised by a small group (usually less than 10) of developers acting as a collective decision making body. Entry into the group is controlled; acceptance of a new member usually requires the entrant to demonstrate their commitment and technical competence through regular contributions to the source code. The decision making process within the controlling group differs between projects, some projects have formal voting procedures; others find agreement through informal discussions.

- *Corporate governance*. Many of the largest OSS projects have developed formal governance structures. These typically include marketing organisations, IPR ownership bodies, steering groups etc. Many projects have discovered that these structures are required as the number of people involved starts to grow and especially when corporate interests start to get involved. A good example of this is the development of the K Desktop Environment (KDE). KDE began as an informal OSS project organised along the lines of the cabal model. As the project grew, many of the developers started to voice concerns that their voices were not being heard. The motivation of one of the companies behind part of the project (Trolltech, now owned by Nokia) was also questioned. To address these concerns and to put the project on a firm footing the project leaders established the KDE Free Qt Foundation and the KDE e.V (a registered association under German law). Similar patterns can be seen in other projects, and the common themes of these organisational structures are discussed further below.

- *Company control*. Projects that are driven by a single company are often controlled exclusively by that company. These projects often require contributors to assign their copyright over to the company if they wish to submit changes. Why would a contributor be prepared to provide a company with such free work? The key to getting contributors to give their effort to such projects lies in the ability of any disgruntled contributor to fork the project. The importance of the ability to fork is covered in more detail below. There are many examples of these company controlled projects; MySQL is a well known example.

An assessment the whole-life-cost implications of using an OSS application as part of a larger system should use knowledge of the development model as a tool in that assessment. Modifiability is an important element of whole-life-cost. The organisational structure of the project will affect the cost of getting changes made.

Regardless of which organisational model is adopted there are common issues that must be dealt with by all OSS projects:

- *Clear IPR ownership*. Many OSS projects choose to leave the copyright ownership with the contributor, others require contributors to assign their copyright to either a not-for-profit body or the company backing the development. Whichever approach is used, it is important that it is clear to all contributors exactly where copyright ownership will reside.
- *Rigidly controlled project lifecycle rules: release cycles, testing etc.*. All projects live and die by the project lifecycle management. Each of the organisational models described above have different decision making structures. All but the very smallest have formal lifecycle management processes. These processes may not always be well documented but they are formal nonetheless.
- *Strong reliance on configuration control tools*. At the very heart of just about every OSS project is the Source Control System (SCS). The SCS provides a technical mechanism for the control of the application lifecycle. It is the touchstone for all the developers on the project. The decision about which SCS tool should be used can be extremely contentious. For example, the long running arguments over the use of BitKeeper for the Linux Kernel became quite divisive and in the end led to the development of a new SCS called GIT.

One of the key features of the OSS licenses is that they allow the possibility for disaffected developers to ’fork’ the project. A fork is the establishing of a rival project to continue the development of the same source code. A group of developers takes a copy of the projects source code and sets up a new source code repository under a new name. A ’fork’ is often brought about because some developers are unhappy with the organisational model or some technical decisions that a project has adopted. This ability to ’fork’ provides a powerful counter balance to the power of the organisational and technical leaders of a project. Many contributors are reassured that their contributions are in safe hands because they can see that if all else fails they can simply carry on development in a new fork of the project. This freedom helps to engender a culture of trust amongst the development team.

There is a fine balancing act that needs to be played by any OSS development. If the rules of the project, governed by the license regime, enables project ’forking’ then the project organisation must achieve broad consensus on any decisions otherwise the disaffected might simply go off on their own. If the rules of the project do not allow ’forking’ it will not be seen by most developers as an OSS project at all and the benefits of collaborative development will be significantly reduced. An assessment of OSS projects might use the existent, or otherwise, of forks as a proxy measure for effective organisational management.

4 Legal Bodies

The larger, higher profile, OSS projects tend to adopt the Corporate Governance or Company Control models. These projects usually create a legal body that holds the ownership of the source code and provides organizational, legal and financial support. This is usually a not-for-profit organisation but may be a normal commercial company. The legal framework differs depending on which country the body is created in. Examples include the Eclipse Foundation, the GNOME Foundation and the Apache Foundation

These legal bodies provide a clear separation that makes the motivations of the people involved explicit. The IPR owning body can be a 'not-for-profit' body or a company. The choice of who owns the IPR makes a clear statement about the intentions of the project. A 'not-for-profit' body will generally be trusted not to use the IPR ownership for commercial purposes that go against the wishes of the developers. However, if it is a company that retains the IPR ownership, it is clear to everyone that their contributions might be used by that company for commercial gain. Both models can be successful. For example, contributions to MySQL are owned by MySQL AB whereas contributions to GCC are owned by the FSF (a 'not-for-profit' body).

Not all large OSS projects have an IPR owning body but it is a growing trend in the largest projects. Projects that do not have such a body leave the ownership of the code in the hands of whoever wrote each line. For many projects this means that the only way that the license on the code can be changed is to trace the author of each line of code and ask them to agree to the license change. For many projects this is not a practical proposition. The implication of this is that any sizable project that does not have an IPR owning body cannot change its license.

OSS assessment methods may be able to use the status of IPR ownership as a measure. However, it is not clear how this measure should be interpreted.

5 Stakeholders

The OSS development model has a significantly more complex arrangement of stakeholders when compared to a conventional COTS development. The tables and diagrams below show these stakeholders and their relationships to one another. These are the stakeholders as seen from the perspective of those outside of the development itself. If you were inside a COTS development, as a developer for instance, you would see many of the stakeholder roles that are present in the OSS model. However, these roles are purely internal to the COTS development model. The exposure of these roles in OSS models may provide opportunities for assessment measures.

Figure 1 shows a general model of the stakeholder landscape in a typical COTS marketplace.

Table 1 describes each of the COTS stakeholder roles.

In a conventional COTS development model, the stakeholders that would typically seek to apply assessment methods (e.g. CMM) are the Integrator and occasionally the End User. The relatively simple stakeholder landscape limits the number of exposed interactions and provides few opportunities for the gathering of direct assessment measures. However, it does have the merit that the interactions between the stakeholders are well understood and clear to all parties.

Figure 2 shows the complex interactions of the stakeholders in the OSS marketplace. This is a general model intended to expose all of the different roles. Not all OSS projects will have people fulfilling each of these roles.

Table 3 describes each of the OSS stakeholder roles.

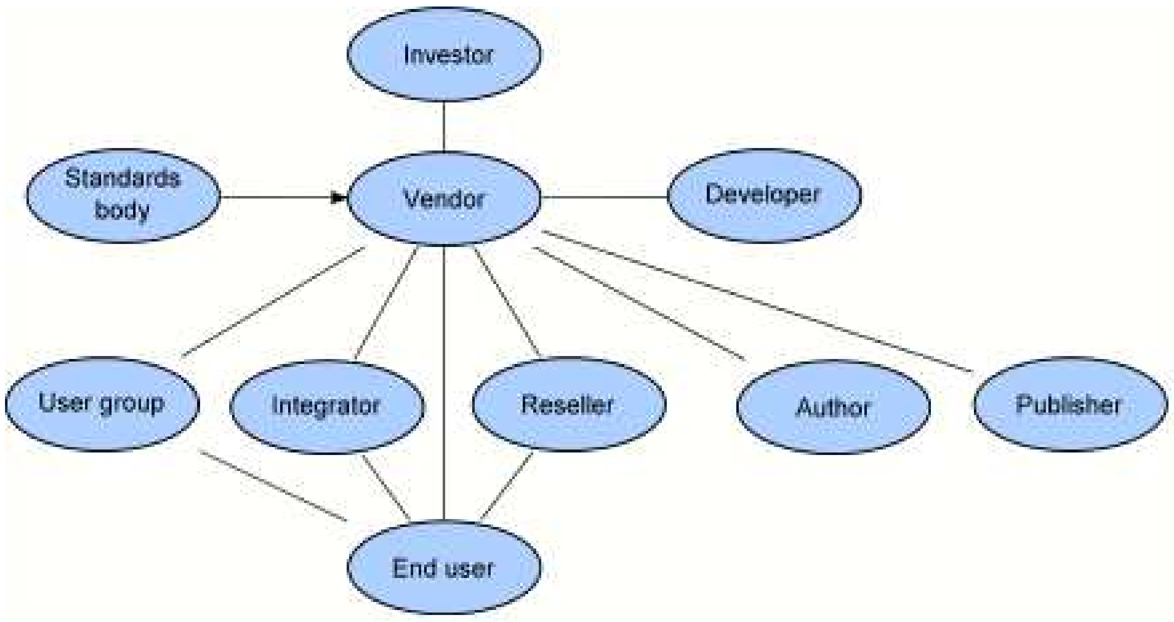

Figure 1: COTS Stakeholders

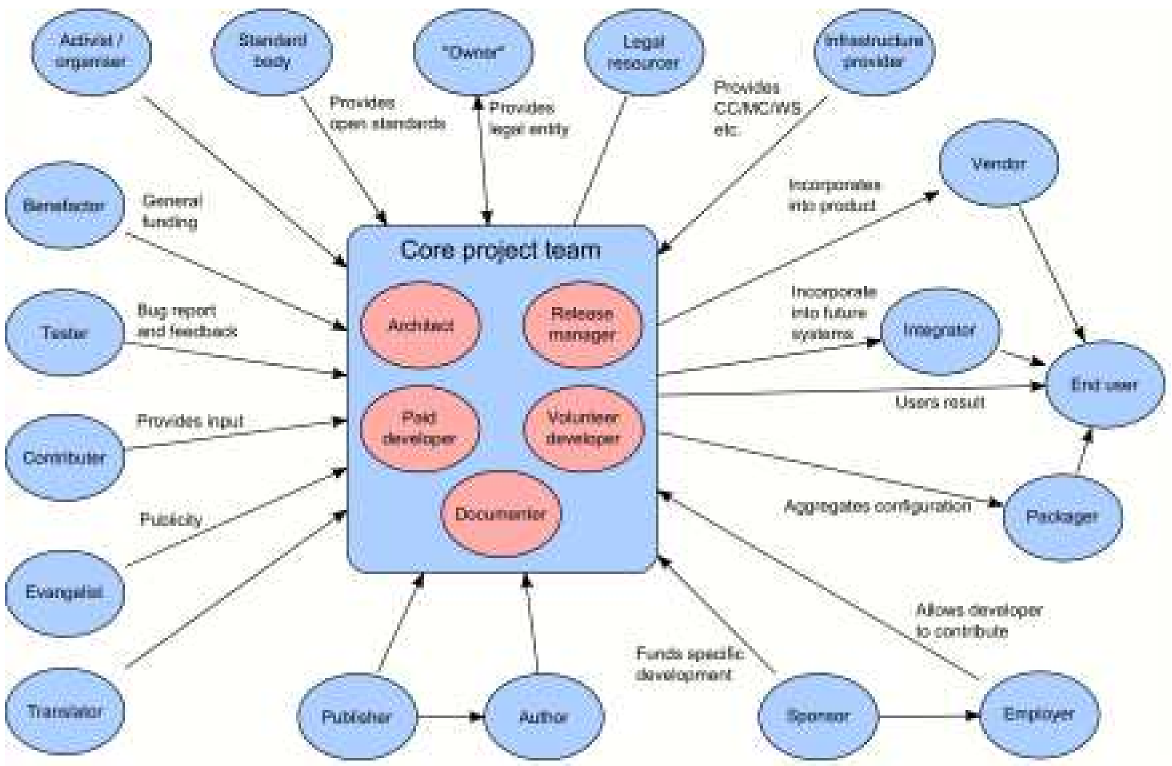

Figure 2: OSS Stakeholders

Table 1: COTS Stakeholders

Stake Holder	Role / Motivation
Investor	Provides financial backing to vendor. Primarily interested in medium/long term return. Will influence vendor to exploit maximum value from COTS applications.
Vendor	Primary owner of software copyright. Motivation is to build a profitable business from the development the COTS application. Will concentrate on largest / most profitable users needs.
Developer	Employed by vendor to develop an application. Professionally motivated relationship with project.
Author	Authors books about application. Often a professional author writing for a large user community.
Reseller	Pre-sales support and distribution channel for vendors. Profits from margin on application sales price.
End User	User of application. May provide some feedback to vendor, sometimes via Reseller, Integrator or User group. Rarely has a direct contact with Developers.
Integrator	Integrates application into larger systems on behalf of clients. Often has a large role in providing feedback to vendor. Isolates end user from support and integration issues.
User group	Collective voice of end users. Sometimes organised and run by the Vendor sometimes run as a not-for-profit body by highly motivated end users. Lobbies Vendors. User group often voices the concerns of End Users when the Vendor is not looking after existing customers whilst it tries to drive sales to new clients.
Standards Body	Sets standards that Vendors then implement. Usually not-for-profit bodies, either independently constituted (e.g. ISO, IETF etc.) or run by a consortium of End Users. The level of implementation of standards by a Vendor is strongly influenced by the strength of the user community. If there is good competition in a market segment and the user community demonstrates its desire for standards compliance, the Vendor will often have to comply. Where there is a de-facto monopoly or a few large Vendors, standards compliance is usually much less.

Table 2: OSS Stakeholders

Stake Holder	Role / Motivation
Architect	Controls the direction of development, lays down overarching structure and guiding principles. Often acts as final decision maker. All OSS projects have at least one architect. They may also be amongst the most prolific developers but this is not always the case. Small projects will typically have just one person who performs this role, larger projects will have architects who control various sub-systems and may well have formalised decision making structures.
Paid / Volunteer Developer	The developer fulfils the most important role in an OSS project. Without the developer, there is no code. The majority of developers are motivated by a personal interest in the application or the particular problem that it is trying to solve. The complex motivation of volunteer developers has been addressed by a number of academic studies. Many developers are paid for their work by Sponsors that wish to influence the direction of development. A good example are developers that work on the Linux kernel that are employed by IBM. IBM have a direct interest in the development of the Linux kernel because they use it on their own products so they deploy their own employees on the project. It must be presumed that having developers working on the software and respected in the project gives IBM the best chance of influencing the development.

Release manager	OSS projects are often highly distributed. They will frequently have developers working on different continents, on different timescales and these developers have often never met one another. Their development work can be sporadic, as volunteer developers find the time to work on tasks. OSS projects also often have complex interdependencies with other OSS projects. An end user application will depend on many OSS libraries, and those libraries will themselves depend on other lower-level libraries and so on. Also, as with conventional software development, there is usually the need to maintain multiple lines of development and support beta-testing. Large projects usually have people that are dedicated to coordinating this release process. The process happens completely in the open so that anyone can follow what it going on and everyone can see precisely what software makes it into a release. Release management has developed in the OSS community to be able to deal with these large, complex distributed organisational structures. The competence of the Release Managers can be a significant feature in the ability of OSS projects to scale.
Documenter	Project documentation, including web site management is often performed by people that want to contribute to the project but do not have technical programming skills. Documentation is seen by many OSS projects as vital to expanding their user base. Other projects are much less interested in end user documentation. Attracting and retaining skilled technical documentation people on OSS projects is something that many projects find a challenge.

Standards body	These play a slightly different role in many OSS projects than they do in a conventional COTS environment. OSS developers view standards in a different light to conventional Vendors. This is primarily because OSS projects gain little benefit from locking their users in with non-standard interfaces. OSS developers view standards in one of two ways: either as a means to achieve interoperability with existing applications; or as free design work. The later reason perhaps requires a little explanation. When faced with implementing a new function, an OSS developer is generally interested in minimising the amount of work required. If there is a standard already written that explains how to achieve the required functionality it is simpler to implement this standard rather than invest effort in designing an alternative approach. This is coupled with a certain pride in following standards.
Activist / organiser	Larger OSS projects need people that provide organisation and coordination. These may not necessarily be developers.
Benefactor	There are many organisations that provide support to OSS projects. Many companies allow their workers to use company infrastructure to work on OSS projects and many universities allow their academics and students to do the same. There are also the occasional donors that contribute financial support to an OSS project. Once example is the Google Summer of Code, this Google led initiative funds students during their summer break to work on OSS projects. In 2007 this program funded over 900 students to work on more than 130 different OSS applications.

Tester	There are a number of types of testers that typically play a role in OSS projects. The current development version of the project is always available to anyone that wishes to try it out. This encourages those that have an interest in the project to experiment with unstable versions and they provide feedback to the developers. Many developers start their involvement with a project by test driving development versions and attempting to fix problems that they find. Many projects release test versions for formal test cycles and encourage early adopters to report problems before stable releases are made. The people that use these test versions and report problems are vital to the quality of the projects. Finally, OSS projects rely on their users to report problems and to help with debugging activities. Users are often encouraged to be actively involved in the testing process.
Contributor	Anybody that provides their time, infrastructure or money to an OSS project is a contributor.
Evangelist	OSS as a movement has been established by people that feel strongly about the benefits that it brings. Some are politically motivated, others are much more pragmatic. Some of the growth in OSS can be attributed to the evangelism of these people. Activists such as Richard Stallman and Eric Raymond have travelled widely to educate users, companies and developers. As OSS has until recently lacked any significant marketing money, and still cannot compete with the COTS vendors in marketing spend, it has relied heavily on individuals to spread the word.
Translator	An increasing number of OSS projects are translated into many different languages. For example, Abiword has been translated into more than 50 languages. The major OSS application frameworks have developed mature infrastructure to support translation of user interfaces. The translations rely on native speakers volunteering to translate all the text used in the user interface. The translators are often users rather than developers.
Owner	The copyright owner of the source code. This may be a company, a not-for-profit body or it may be distributed amongst all the authors of the individual lines of source code. Organisations like the FSF will accept copyright ownership of some projects, which can be useful, if project teams do not wish to set up their own not-for-profit body.

Legal Resource	Some organisations offer legal support to OSS projects. Bodies such as the Apache Foundation provide legal support for those projects that are part of a specific collective. Other bodies like the Electronic Frontier Foundation offer support more broadly.
Publisher	The take-up of OSS applications has been helped by the availability of conventional printed technical books. The publishers of these books, such as OReily, have played an important role by supporting authors. These authors are often amongst the lead developers of the projects.
Author	There are many conventional print books that document OSS applications. The authors of these books are often developers on the project, but many are professional technical writers.
Infrastructure provider	Most OSS projects rely heavily on the availability of free infrastructure for source code control, issue tracking, mailing lists etc. A number of organisations provide this infrastructure. For example SourceForge, Inc, and BeriOS. The funding model for these services is somewhat difficult to work out. Some certainly make some money from carrying advertising and some use the OSS infrastructure service as a means of advertising other commercial services. Some are simply altruistic and there may be some financing coming from the major commercial players.
Vendor	The vendors (often called distributors), such as Redhat Inc and Novell, provide a conventional shop front for OSS technology. They provide integrated collections of packages, documentation and support contracts. Vendors relationships with individual OSS projects can, at times, be fractious. Vendors need to find workable business models to ensure that they stay in business whilst at the same time maintaining the support of the volunteer OSS developers. Vendors are amongst the most important Sponsors and Employers of OSS developers.
Integrator	Integrators work on behalf of customers to integrate OSS, as well as COTS, into larger systems. Integrators generally have sufficient technical resources to be able to tailor OSS applications for their customers needs and, if required, to offer tailored support.

End User	End users of OSS applications are no different from end users of COTS applications. However, an OSS end user can also choose to play one of the other stakeholder roles. OSS projects are often sustained by a steady flow of end users that become interested in the way that the application is developed and contribute as Documenters, Testers, etc.
Packager	OSS projects release source code but End Users like the convenience of nicely packaged applications that are pre-compiled for their chosen Operating System. There are a number of different packaging formats in the Linux world as well as a different form of packaging for MS Windows and for the Apple Mac. Most OSS projects do not have the resources to be able to produce all these packages. Instead they rely on Packagers. Sponsor Organisations that require particular features to be added to a project may sponsor a developer to make the required changes. The distinction between a Sponsor and a benefactor is that a Sponsor is providing support for specific business reasons, whereas a benefactor is usually acting in a more philanthropic way.
Employer	Many OSS developers are supported either explicitly or implicitly by their employers. A recent analysis of changes made in the last year to the Linux Kernel suggested that as much as 70% came from developers working with at least the tacit support of their employer.

The richness of the stakeholders, their motivations and roles they play in the OSS development model provides an opportunity for new assessment approaches. By a careful understanding of how each of the stakeholders interacts within a given project, it may be possible to design direct measurements that shed light on its properties. There is a stark contrast with a closed COTS development where the limited exposed stakeholder interactions constrains the scope of direct measures.

6 Maximising Participation

OSS projects often live and die on their ability to attract participants. Maintaining enthusiasm and fostering a collaborative attitude is critical to the success of a collaborative project. Studying successful OSS projects shows that the following are common amongst those that attract the most developer effort:

- good code structure to help people understand the software;

- low cost of entry to participants i.e. no expensive tools, no closed networks, no complicated signup procedures, extensive use of open source support tools, restricted use of meetings, deskilled build process;
- open communication channels, Mailing lists, News, IRC, web site, wiki;
- perception of activity, people more likely to join if project is perceived as dynamic - release early, release often, use open communication channels for development discussion;
- friendly to newcomers;
- many levels of participation - user, bug submitter, documenter, translator, occasional bug fixer, coder, architect, etc.;
- open requirements.

These project characteristics are rare amongst traditional software development projects because there is little need to attract participants. However, they are vital for OSS projects and can form the bed rock of an successful assessment approach.

7 Anatomy of a Project

As OSS has evolved, it has developed a set of common support technologies that provide an infrastructure to support the distributed collaboration upon which projects depend. These technologies need to provide a mechanism for geographically disparate teams to work closely together both to develop the software and to organise the projects.

- *Web site*. Most OSS projects have a web site. This provides: advertising for the project; documentation; tutorials; download and installation instructions; descriptions of what the project is trying achieve; contact points; and links to other resources. The web site is usually the first point of contact with new users and prospective contributors.
- *Wiki*. Many projects now also have a project wiki. A wiki is a web site that can be altered by anyone. OSS projects use a wiki as an informal repository of discussions, user oriented documentation and a social networking centre for project contributors.
- *Bug/Issue tracker*. Bug/issue trackers are used to capture and record bugs and requested features. Bug trackers used to be a feature of only large OSS projects but the ready availability of bug tracking software made available by sites such as Sourceforge has led to many smaller projects now relying heavily on the bug tracker to help organise development. End users can report bugs directly in to the bug tracker and projects then often have a formal process by which bugs undergo triage before being assigned to a developer. In some projects, it is the developers responsibility to select the bugs that they feel they can address, in others bugs are assigned to developers. Bug trackers provide automatic notification mechanisms so that everyone involved can track the lifecycle of a bug with minimum effort. One aspect of OSS project bug trackers is that anyone can look at the

current list of bugs / issues. This openness can be hard for some corporate developers to come to terms with.

- *Mailing lists*. OSS projects frequently have a number of email mailing lists. Common lists are 'announce' (which only carries announcements of releases etc.), 'user' (which carries discussions between users about how to use the software) and 'developer' (which carries discussion about development issues including future architectural developments). The mailing lists are often the heart of the OSS development. The use of mailing lists for this purpose has grown out of the necessity to deal with discussion between groups of people that cannot meet and are often in different time zones. They have proved to be a very successful means of collaboration, but they do require all participants to be open and be prepared to put their views in writing. This can be a challenge for corporate developers because of the potential legal issues that might arise from inappropriate correspondence.

- *Instant messaging*. Instant messaging (or chat) has been a feature of OSS development long before it became widely used by teenagers and corporate Intranets. OSS projects often have chat rooms (usually using a chat technology called IRC (Internet Relay Chat)). These are used to provide support to users and as a social resource for developers. Chat forums between developers help to cement personal relationships because they are much less formal that mailing lists. It helps to engender a sense of community.

- *Source repository*. The source code repository is the most important part of an OSS project's infrastructure. It provides the mechanism for multiple developers to work simultaneously on a large software code base. The unique requirements of OSS projects have driven the development of a number of source code control technologies in recent years. The source code repository supports the separation of stable and development branches and, can in some cases enable a completely distributed development model that does not even require a central source code repository. As a general principle, anyone can access an OSS project's source code repository; anyone can browse the current state of the development braches or download any previous version of the code base. However, only those authorised by the development team can make changes to the code and all changes are accounted for by the source code repository software.

With all these supporting infrastructure components, the objective is to use open, multiplatform, standards. Most OSS projects have no resources to purchase proprietary products and most contributors will be put off if they have to put their hands in their pockets before they can get involved in the project. The principle is to make the barrier to using or contributing to the project as low as possible. This provides a real opportunity for automated assessment tools. The number of infrastructure applications (e.g. Mailing List servers, source code control systems etc) that these would need to work with are small and their interfaces are open.

8 Summary

OSS projects differ from COTS projects in their need to effectively manage collaborative development and attract participation in the project. While each project is unique, most OSS projects tend to fit one of a number of development models with common characteristics.

The OSS development model has a significantly more complex arrangement of stakeholders when compared to a conventional COTS development.

Many OSS projects employ a governing body independent of the code developers. This body, often known as a Foundation, is largely responsible for promoting participation in the project and ensuring successful evolution. The Foundation provides non-software development activities such as IPR management, marketing, promoting the use of standards, and support for implementation of a consistent, structured development process.

To support collaborative development among geographically disparate teams, OSS projects typically deploy a standard set of supporting technologies. These include a web site, mailing lists, bug tracking facilities, a wiki and a source code repository. These services are often managed by the projects foundation body.