Open Governance Index

Measuring the true openness of open source projects from Android to WebKit

Published July 2011

Created by vision mobile

Part-funded under webinos

This work is partially funded by webinos, an EU-funded project under the EU FP7 ICT Programme (#257103)

About VisionMobile

VisionMobile is a leading market analysis and strategy firm, for all things connected. We offer competitive analysis, market due diligence, industry maps, executive training and strategy, on topics ranging from the industry's hottest trends to under-the-radar market sectors. Our mantra: distilling market noise into market sense.

VisionMobile Ltd.
90 Long Acre, Covent Garden,
London WC2E 9RZ
+44 845 003 8742

www.visionmobile.com/blog
Follow us: @visionmobile

About webinos

This research was partially funded by webinos, an EU-funded project under the EU FP7 ICT Programme (#257103).

Webinos is an EU-funded project aiming to deliver a platform for web applications across mobile, PC, home media (TV) and in-car devices.VisionMobile is a member of the webinos consortium. More info at www.webinos.org

License

Disclaimer

VisionMobile believes the statements contained in this publication to be based upon information that we consider reliable, but we do not represent that it is accurate or complete, and it should not be relied upon as such. Opinions expressed are current opinions as of the date appearing on this publication only, and the information, including the opinions contained herein, are subject to change without notice.

Use of this publication by any third party for whatever purpose should not and does not absolve such third party from using due diligence in verifying the publication's contents. VisionMobile disclaims all implied warranties, including, without limitation, warranties of merchantability or fitness for a particular purpose. VisionMobile, its affiliates and representatives shall have no liability for any direct, incidental, special, or consequential damages or lost profits, if any, suffered by any third party as a result of decisions made, or not made, or actions taken, or not taken, based on this publication.

v.1.11

Contents

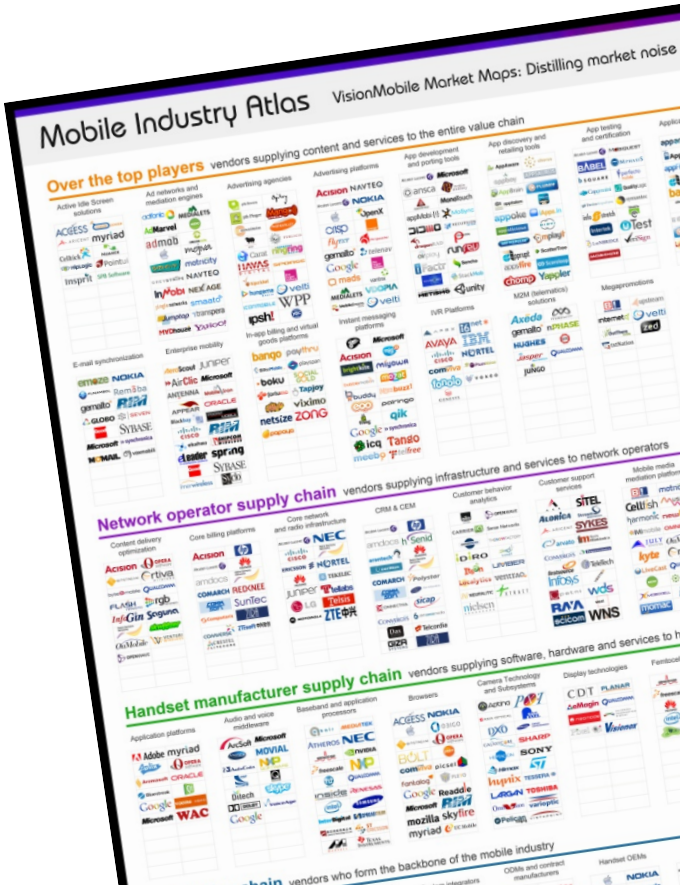
Authors

Liz Laffan, BABS, MA(IPE).
Liz Laffan is a Research Partner at VisionMobile. Liz has been working in the telecoms and mobile industry for over 20 years, with large telco organisations, start-up technology ventures, software development and licensing firms. Liz's interests lie in open source software governance and licensing and in particular how best can commercial organisations interact with open source projects. She can be reached at liz@visionmobile.com'

Also by VisionMobile

Mobile Industry Atlas | 4th Edition
The complete map of the mobile industry landscape, mapping 1,350+ companies across 85+ market sectors.

Available in wallchart and PDF format.
www.visionmobile.com/maps



Key messages

Business as usual?. Use of open source software in the mobile space is now business as usual. Much has been written and debated regarding open source licenses – from the early days of the GPL license to the modern days of the Android platform.

Openness as Governance Despite the widespread use of open source, from Android to WebKit, there is one very important aspect of open source projects that has been neglected: openness and how to measure it. Openness goes far beyond the open source license terms and into what is termed Governance.

Governance goes beyond licenses. While licenses determine the rights to use, copy and modify, governance determines the right to gain visibility, to influence and to create derivatives of a project, whether in the form of spin-offs, applications or devices.

Governance captures all the hard questions. The governance model used by an open source project encapsulates all the hard questions about a project. Who decides on the project roadmap? How transparent are the decision-making processes? Can anyone follow the discussions and meetings taking place in the community? Can anyone create derivatives based on the project? What compliance requirements are there for creating derivative handsets or applications, and how are these requirements enforced? Governance determines who has influence and control over the project or platform – beyond what is legally required in the open source license.

Governance determines openness. In today's world of commercially-led mobile open source projects, it is not enough to understand the open source license used by a project. It is the governance model that makes the difference between an "open" and a "closed" project.

From Android to WebKit. To quantify governance, we researched eight mobile open source projects: Android, MeeGo, Linux, Qt, WebKit, Mozilla, Eclipse and Symbian. We selected these projects based on breadth of coverage; we picked both successful (Android) and unsuccessful projects (Symbian); both single-sponsor (Qt) and multi-sponsor projects (Eclipse); and both projects based on meritocracy (Linux) and membership status (Eclipse).

Open Governance Index. We quantified governance by introducing the Open Governance Index, a measure of open source project "openness". The Index comprises thirteen metrics across the four areas of governance:

1. Access: availability of the latest source code, developer support mechanisms, public roadmap, and transparency of decision-making

2. Development: the ability of developers to influence the content and direction of the project

3. Derivatives: the ability for developers to create and distribute derivatives of the source code in the form of spin-off projects, handsets or applications.

4. Community: a community structure that does not

Open Governance Index (% open)	
Android	23%
Qt	58%
Symbian	58%
MeeGo	61%
Mozilla	65%
WebKit	68%
Linux	71%
Eclipse	84%

discriminate between developers

The Open Governance Index quantifies a project's openness, in terms of transparency, decision-making, reuse and community structure.

Does openness warrant success? Our research suggests that platforms that are most open will be most successful in the long-term. Eclipse, Linux, WebKit and Mozilla each testify to this. In terms of openness, Eclipse is by far the most open platform across access, development, derivatives and community attributes of governance. It is closely followed by Linux and WebKit, and then Mozilla, MeeGo, Symbian and Qt. Seven of the eight platforms reviewed fell within 30 percentage points of each other in the Open Governance Index.

Best practices of open governance. Our research identified certain attributes that successful open source projects have. These attributes are timely access to source code, strong developer tools, process transparency, accessibility to contributing code, and accessibility to becoming a committer. Equal and fair treatment of developers – "meritocracy" – has become the norm, and is expected by developers with regard to their involvement in open source projects.

The Android Paradox. Android ranks as the most closed project, with an Open Governance Index of 23%, yet at the same time is one of the most successful projects in the history of open source. Is Android proof that open governance is not needed to warrant success in an open source project?

Android's success may have little to do with the open source licensing of its public codebase. Android would not have risen to its current ubiquity were it not for Google's financial muscle and famed engineering team. More importantly, Google has made Android available at "less than zero" cost, since Google's core business is not software or search, but driving eyeballs to ads. As is now well understood, Google's strategy has been to subsidise Android such that it can deliver cheap handsets and low-cost wireless Internet access in order to drive more eyeballs to Google's ad inventory.

More importantly, Android would not have risen were it not for the billions of dollars that OEMs and network operators poured into Android in order to compete with Apple's iconic devices. As Stephen Elop, Nokia's CEO, said in June, 2011, "Apple created the conditions necessary for Android".

A. Open Source Economics

The beginnings

Although open source software (OSS) has been used in the PC space since the 1980s, it has only proliferated in the mobile space since the early 2000s. But where has open source come from?

In the early days of personal computers, software was originally given away free or shared between hobbyists, prompting a young Bill Gates in 1976 to complain, "As the majority of hobbyists must be aware, most of you steal your software," and, "One thing you do is prevent good software from being written"[1]. The context for this letter was that programmers were using Microsoft's Altair BASIC software without having paid for it, and that such unauthorized copying discouraged Gates and other developers from investing time and money in creating high-quality software.

Nearly four decades on, and we now know that this argument holds little ground. It is estimated that it would cost around $1.4billion to create the Linux kernel itself[2]; around $10.8 billion to build the Fedora 9 Linux OS[2], and around $89m to create the WebKit browser engine[3]. OSS has now more than ever demonstrated its value to the mobile software industry.

Cultural roots

The "free software" movement – as distinct from "open source software" – was started by Richard Stallman, who in 1989 authored one of the most widely used open source licenses, the GNU GPL . Stallman also founded the Free Software Foundation (FSF), and wrote large amounts of code, mostly related to EMACS and the GNU system[4]. Stallman passionately believes that source code should be "free," such that users and developers can do with it what they choose, a belief embodied in the "four freedoms" that underpin Stallman's philosophy of free software:

1. The freedom to run the program, for any purpose (freedom 0).

2. The freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition for this.

3. The freedom to redistribute copies so you can help your neighbor (freedom 2).

4. The freedom to distribute copies of your modified versions to others (freedom 3). By doing this, you can give the whole community a chance to benefit from your changes.

Moreover, the Free Software Foundation maintains that "free software is a social movement. For the free software movement, free software is an ethical imperative, because only free software respects the

users' freedom"[5]. As such, other considerations like business needs and the drive to create revenues and profits are secondary to this philosophy.

Not surprisingly, this approach proved unpopular with commercial software developers, especially given that most software was licensed in binary form only. This was because commercial software owners believed strongly that providing source code would devalue the software, with the secret sauce being accessibility to source code.

The commercial benefits of making source code "open" were further advocated when Eric Raymond wrote the seminal book "The Cathedral and the Bazaar" in 1997. Raymond contrasted the traditional in-house software development model (the cathedral) with the open source development process (the bazaar), based on his observations of the Linux kernel development process and his experiences managing an open source project (fetchmail).

Raymond introduced the term Linus's Law, which states that "given enough eyeballs, all bugs are shallow". That is, the more widely available the source code is for public testing, scrutiny, and experimentation, the more rapidly all forms of bugs will be discovered. This viewpoint was seen as more pragmatic than that of the FSF, acknowledging the benefits of an open source development model whilst not mandating 'software freedoms' (the right to make, run and distribute changes in source code) to the same extent as the free software movement.

Later in 1998, Eric Raymond and Bruce Perens founded the "Open Source Initiative" (OSI) , tasked with promoting open source software use by and for commercial organisations, as well as the open source community. Today, the OSI manages a 10-point definition[6] of open source software, and maintains a list of licenses it deems to be in compliance with this definition. Currently there are around 70 OSI-approved licenses listed, including the popular General Public License v2.1 and v3.0 (GPL); Lesser General Public License v2.1 and v3.0 (LGPL); Eclipse Public License v1.0 (EPL) and the Apache License v2.0 (APL)[7].

In summary, the OSI definition advises that open source licenses should provide users with access to the source code and permit free redistribution of that source code, including the ability to create modifications and derivatives, which can also be freely redistributed. Moreover, an open source license should not have any field-of-use restrictions, nor discriminate against any persons or organisations. Finally, the license must not place restrictions on other software that is distributed along with the licensed software.

What on earth does open source mean?

As we've seen, open source is rooted in both legal and cultural contexts. Today, however, the term "open source" is used in far more practical ways:

Open source as a development methodology. In a commercial context, open source is a collaborative software development methodology. Much like industry consortia or commercial partnerships, open source is a technique for collaboratively developing software building blocks across multiple commercial entities. As such, open source is a mechanism for sharing both risks and costs of collaborative software development.

Open source as a product decision. In software products, companies often need to make a decision between building a new software component in-house or buying the component from a third party. Open source is a mid-level option between build and buy that allows a company to source software from an external "community" and co-develop it.

Open source as a marketing tool. Open source software can be leveraged as a means of building a reputation for benevolence and good intentions, attracting positive press, and even drawing followers to a software platform. This is best exemplified by Google's use of open source licensing to help create a fanatical developer following around Android, even if the entire development process happens behind closed doors.

Having seen what open source means in practice, it is equally important to note what open source software is not.

Open source is not free of cost. Open source software development is not free-of-cost; there are costs with regard to customisation, adaptation, third party component integration, software support, maintenance of private software branches, trademark or compliance requirements, legal/technical due diligence, and in some cases membership or sponsor fees. Indeed we have seen how Microsoft is able to extract a patent fee from Android manufacturers due to the alleged infringement that Android carries against Microsoft patents[8].

Open source is not a community builder. Open source is often compared to "build-it and they will come," implying that developer communities will naturally form around software than is released under an open source license. Yet this couldn't be further from the truth. Software developers are human by nature and as such are intrinsically self-centred. Developers will take an interest in an open source project only if it provides value in some way or "scratches an itch" in hacker-speak. This is why successful open source projects – e.g. Linux, GTK or WebKit – address a common need such as building a vendor-neutral operating system, a graphics software stack or a browser engine.

Understanding Projects, Distributions and Platforms

Open source software comes in many forms: projects, distributions and platforms

Projects. An *open source project* comprises of one or more applications – for example Mozilla is an open source project that has created a number of applications, of which Firefox, the open source web - browser, is one. Another example of an open source project is Eclipse, which started out primarily as a tools package, but now hosts around 240 distinct open source projects.

Distributions. The Linux kernel is also an open source project, but it nearly always comes pre-packaged as part of a *software distribution* comprising various libraries and files in order to make the code useful. A distribution is typically not 100% complete, in that it is designed for customisation, such that end users can easily add and remove components. Examples of Linux distributions include Ubuntu, a complete desktop Linux operating system; Debian which is a free operating system (OS) for computers and Fedora, a Red Hat-sponsored, Linux-based operating system. The term "distribution" is not limited to Linux distributions, though, nor even to complete operating systems, such as the Berkeley Software Distribution (aka, BSD Unix). Instead, a distribution is simply a bunch of software

typically distributed together. Another example might be busybox, a "Swiss Army knife" of embedded system shell utilities intended to be compiled selectively into a single executable.

Platforms. The term an *open source platform* is used to refer to a complete software package which includes all the necessary applications, files and libraries such that it runs by itself with limited requirement for additional software. Examples of open source platforms are Android and the older Symbian platform.

Working upstream vs. downstream

There is much interdependency across open source projects. For example, Android comprises around 185 different components, a large number of which are separate open source projects. For example, the Linux kernel, WebKit, Open GL ES and SQLite are all maintained separately.

Open source development language uses the notion of a river, in which "upstream" projects (e.g. WebKit, GTK or BusyBox) find their way into "downstream" distributions and platforms.

In open source software, both upstream and downstream software will be developed in parallel. The decision on how to interact with a "live" upstream project – knowing when to fork (i.e. take a copy of the upstream project) and when to merge back the changes – can make the difference between success and failure of the downstream project.

For example, Nokia had forked WebKit from the upstream repository, only to acknowledge that it should not have done so due to the rising costs of keeping up with the innovation in the upstream WebKit project. Based on these learnings, the MeeGo open source platform actively advocates that changes should be made directly to the upstream projects, such that all downstream projects, including MeeGo, can benefit from such changes. This of course makes much sense given that the users of the project then spend less time maintaining separate branches. These forking and merging complexities should not be sidelined as they will critically impact the success of the project in the long run.

The three phases of open source adoption in the mobile space

In the mobile industry, open source software has transitioned from "geekware" to business-as-usual in the short space of 10 years. This transition has taken place in three phases.

2000-2005: The years of experimentation

The early days of mobile open source were led by mobile device manufacturers (OEMs) who were cautiously experimenting in a sandbox, quite separate to their revenue-generating handset portfolio. At that time, OEMs were concerned about the use of open source software from a legal perspective, due to lack of indemnity and warranty provisions, as well as potential patent infringement issues, and the "viral" nature of copyleft open source licenses such as the GNU GPLv2.0. This early phase saw the development of Motorola's Linux-Java platform, the Mizi Linux platform (a full software stack used by Samsung in 2003) and the Linux-based MOAP-L platform developed by DoCoMo in partnership with NEC and Panasonic. Nokia also ventured into open source at this time with the creation of the Maemo platform which was used in the Nokia Internet Tablets. To this date Maemo remains Nokia's most

widely deployed foray into the open source space despite the manufacturer's attempts with Symbian, MeeGo and Qt.

2006-7: Open source goes mainstream

Open source adoption by the mobile industry came into the fore with the creation of the LiMo Foundation in 2007. Founded by the "who's who" of mobile open source at that time – Motorola, NEC, Panasonic, Samsung, NTT DoCoMo and Vodafone – this was the first formal Linux-based software platform for mobile phones with major industry support behind it. In June 2008 the Linux Phone Standards organisation (LiPS) was folded into LiMo, giving it further momentum within the industry. Despite heavy investment, LiMo remained a niche offering, with practically no common software across the so-called "LiMo compliant" handsets.

The year 2007 was also when Intel launched Moblin (Mobile Linux), an open source operating system for embedded devices. Later in 2009, Intel turned Moblin over to the Linux Foundation, with Moblin now part of the Nokia- and Intel-sponsored MeeGo open source project.

2008+: Open source = business as usual

There hasn't been any bigger milestone in mobile open source than the introduction of the Android operating system. In November 2007, Google along with the "Open Handset Alliance," a consortium of 79 hardware and software vendors, announced the search giant's entry into mobile. In 2008, Google released the Android SDK, and offered the $10m Developer Challenge, which was catalytic in attracting developer mindshare. The HTC-built G1, the first Android phone, was launched with T-Mobile in the U.S. in October 2008. The rest is history. By June 2011, Google reported over 500,000 Android phones were being activated daily, with shipments of Android handsets exceeding the iPhone and exceeding the levels of smartphone market leader Symbian.

Prompted by Google's open source move, Nokia announced in June, 2008, that it would be creating the Symbian Foundation by buying-out the remaining Symbian shares and taking the platform open-source. The Symbian Foundation came into being in April, 2009, and the Symbian open source platform was launched in February, 2010, with source code available under the Eclipse Public License. However, it was doomed to be a short-lived project, with Nokia announcing its closure in November, 2010.

In parallel, we saw Trolltech being acquired by Nokia in 2008 for its Qt open source development platform and WebKit browser engine. The latter nearly reached a phenomenal 500 million handsets by the end of 2010.

Today, open source software is "business as usual" in the mobile industry. But, open source is neither a natural evolution nor a one-way street for mobile software, as it was seen in 2008 when Android and Symbian Foundation were launched. In the two-horse race of mobile platforms, we know that proprietary platforms like iOS can be as successful as open source platforms like Android. And while open source plays a key role in developer attraction, it does not by any means warrant success.

Licensing fundamentals and Copyleft vs Copyright

How does open source software work, and how is it different to proprietary ("closed") software?

Copyright and patents are crucial to understanding how open source software works. Today, there are around 70 OSI-approved open source licenses, all of which are based on copyright law, and most of which include patent grants. Moreover, copyright and patents are used as control points within the mobile and consumer electronics industries with regard to what can and cannot be done with software.

There are currently over 20 outstanding patent or copyright infringement cases ongoing in the courts amongst the major device manufacturers. In June, 2011, Nokia won a two-year battle against Apple for infringement of over 46 Nokia patents; Apple must now pay to Nokia an undisclosed sum estimated to be in the 'hundreds of millions of Euros'[9]. Apple has meanwhile initiated patent infringement claims against HTC, the top maker of Android phones, claiming that HTC infringes around 25 Apple patents. Apple is also seeking to ban U.S. imports of HTC manufactured personal electronic devices – which would be a very serious blow to both HTC and Google's Android[10]. Another notable dispute is the Nokia Qualcomm suit and counter-suit regarding various patent infringements, which resulted in Nokia paying around $400m per annum to Qualcomm, according to some estimates[11].

Given that the very foundations of mobile software licensing are based upon ownership of copyrights and patents, and the huge sums of money that licensing of software generates, cross-industry litigation is likely to be the norm for the foreseeable future.

So what is copyright? Copyright is the universal set of rights granted to the creator of the software (or in many cases their employer). Put simply, if you create a new piece of software, you (or your employer) are the copyright holder.

The creator or copyright holder is the only person or legal entity that can grant others the right to copy, distribute and adapt the software. Moreover, the copyright owner is the only one who can license these rights to third parties.

Whereas copyright does not protect ideas, patents do. Patents are granted on a state-by-state basis for a limited period of time. As in copyright, the patent owner can license the software for third parties to copy, distribute, or adapt.

Copyleft – one of the main innovations of the Free Software movement – is a word play on copyright. Copyright law is used by an author to prohibit others from reproducing, adapting, or distributing copies of the author's work. In contrast, copyleft allows an author to give out copies of a work with permission to reproduce, adapt or distribute, and in addition requires any resulting copies or adaptations to also be bound by the same license agreement.

Copyleft is embodied in the GPL license, written by Richard Stallman. Stallman calls the GPL "copyleft" since it does the opposite of copyright licensing – instead of stopping you from sharing the source code, it obligates you to share source on request. Therefore it is important to understand that while copyleft software is freely available to access, it is not "free" from obligations or restrictions. For a detailed analysis of the GPL 2.1 and its successor the GPL v3.0, see VisionMobile's research paper "GPLv2 vs GPLv3: The Two Seminal Licenses, Their Roots, Consequences and Repercussions" available on the VisionMobile website.

The next table summarises the most popularly used OSS Licenses. The terms "strong copyleft," "weak copyleft" and "permissive" indicate the extent to which the license mandates specific redistribution license terms for users of the software.

Popular open source licenses

License	Type	Projects using this license	Usage in OSS projects (est.)
GPL v2.1	Strong copyleft	Linux kernel, Qt	45%
LGPL v2.1	Weak copyleft	WebKit, Qt	8%
MPL v1.1	Weak copyleft	Firefox web browser, Thunderbird email client	1%
EPL v1.0	Weak copyleft	Eclipse Projects, Symbian	0.7%
MIT	Permissive	- Xorg (an open source implementation of X11, aka the X Window System)	8%
BSD v2.0	Permissive	WebKit	6%
Apache v2.0	Permissive	Apache Software Foundation, Android, Subversion	5%

Source: VisionMobile

It is worth noting that the above seven most popular licenses account for just over 70% of open source projects. This is also one of the key advantages of open source software; there are a few, commonly used OSS licenses, as opposed to the millions of proprietary licenses in existence (one for every piece of commercial software and its licensors).

Trademarks

Trademarks play an important role, not just in promoting brand awareness, but more importantly as a control point in mobile open source projects.

A trademark is typically a name, word, phrase, logo, symbol, design, image, or a combination of the above used by a business organisation or other legal entity to identify products or services as unique. Trademarks are usually registered on a country-by-country basis, and empower the owner of a registered trademark to protect that trademark from unauthorized use.

One of the best-known trademarks in the software industry is that of the Java "cup and steam logo" owned by Sun Microsystems (now Oracle). Historically, Sun has controlled Java in the desktop and mobile space by mandating that distributions pass compliance requirements (technology compatibility kits – TCKs) in order to use the Java logo.

This has allowed Sun to control what is a "legitimate" Java implementation on mobile handsets, but as we know has not been sufficient to reduce Java ME fragmentation across handsets. Similarly, Google

uses the Android trademark (among other control mechanisms) to enforce handset manufacturers to pass compliance certification before they can claim their handset is powered by Android. We analyse Google's control mechanisms in detail in the next chapter.

For now, it suffices to say that trademark control is one of the clearest illustrations of the disconnect between the intent of the OSS movement (which is to minimise proprietary control of software) and successful commercial open source implementations like Android and Sun Java, which use trademarks to control how an OSS platform is distributed and used.

Trademarks are but one control mechanism used "on top" of open source licenses. Commercial open source projects employ a variety of control mechanisms to determine who gets to influence, use and modify the software – all whilst using an open source license. In the next chapter, we analyse how these control points define the **governance model** of an open source project, and how they can be far more critical than the license chosen in determining "how" the software can be used.

B. Open Source Governance

What is Governance?

Much has been written and debated regarding open source licenses – from the early days of the GPL license to the modern days of the Android open source platform. Yet we believe that there is one very important aspect of open source projects that has been neglected: the *open source governance models*.

While licenses determine rights to use, copy and modify, governance determines the rights to visibility, to influence and to create derivatives. And while licenses apply to the *source code*, governance applies to *the project or platform*. More importantly, the governance model describes the control points used in an open source project like Android, Qt or WebKit – and is a key determinant in the success or failure of a platform.

Licenses vs. Governance models

	License	**Governance**
Rights	Use, copy, modify	Visibility, influence and creation of derivatives
Use	70% of projects under 7 licenses	No agreed definition of governance
Examples	GPL, LGPL	No formal examples
Legal	Binding	Non-binding

Source:VisionMobile

The governance model used by an OSS project encapsulates all the hard questions about a project. Who decides on the project roadmap? How transparent are the decision-making processes? Can anyone follow the discussions and meetings taking place in the community? Can anyone create derivates based on that project? What compliance requirements are there, and how are these enforced?

In short, governance determines who *has influence and control over the project or platform* – beyond what is legally deemed in the open source license. In today's world of commercially-led mobile open source projects, it is not enough to understand the open source license used by a project. It is the *governance model* that makes the difference between an "open" and a "closed" project.

Analysis of governance models

We researched eight mobile open source projects: Android, MeeGo, Linux, Qt, WebKit, Mozilla, Eclipse and Symbian. We selected these projects based on breadth of coverage; we picked both successful

(Android) and unsuccessful projects (Symbian); both single-sponsor (Qt) and multi-sponsor projects (Eclipse); and both projects based on meritocracy (Linux) and on membership status (Eclipse).

Our research, carried out over a six-month period, included analysis of these popular open source projects, and conversations with community leaders, project representatives, academics and open source scholars. We acknowledge the work of West and O'Mahony[12], in particular with regard to highlighting the importance of governance, openness and transparency, but our focus has been very much on the use of governance models as a descriptor of open source control points. The table below details the key criteria that we used to assess each of these open source projects in order to identify if the governance model used is an open or closed model.

Key Governance Criteria

Access

1. Is source code freely available to all developers, at the same time?
2. Is source code available under a permissive OSI-approved license?
3. Developer support mechanisms – are project mailing lists, forums, bug-tracking databases, source code repositories, developer documentation and developer tools available to all developers?
4. Is the project roadmap available publicly?
5. Transparency of decision mechanisms – are project meeting minutes/discussions publicly available such that it is possible to understand why and how decisions are made relating to the project?

Development

6. Transparency of contributions and acceptance process – is the code contribution and acceptance process clear, with progress updates of the contribution provided (via Bugzilla or similar)?
7. Transparency of contributions to the project – can you identify from whom source code contributions originated?
8. Accessibility to become a committer – are the requirements and process to become a committer documented, and is this an equitable process (i.e., can all developers potentially become committers?). Note that a "committer" is a developer who can 'commit' code to the open source project. The terms 'maintainer' and 'reviewer' are also used as alternatives by some projects.
9. Transparency of committers – can you identify who committers to the project are?
10. Does the contribution license require a copyright assignment, a copyright license or patent grant?

Derivatives

11. Are trademarks used to control how and where the platform is used via enforcing a compliance process prior to distribution?
12. Are go-to-market channels for applications derivatives constrained by the project in terms of approval, distribution or discovery?

Community Structure

13. Is the community structure flat or hierarchical (i.e., are there tiered rights depending on membership status?)

Google Android

Android was launched amidst developer fanfare and industry scepticism in November 2007. It marked Google's entry into the mobile industry, and fundamentally questioned the business model of handset software with a zero-royalty, open source platform. It was backed by the Open Handset Alliance, a consortium of technology and mobile companies committed to supporting Android.

Android project	
Access	9/19
Development	8/18
Derivatives	3/6
Community	1/2
Open Governance Index	23%

At the same time, Google released the Android SDK and offered a $10m Developer Challenge, which immediately attracted much developer attention. Almost a year later, in October, 2008, Google released the project source code and launched the HTC G1 phone with T-Mobile in the U.S. By mid 2010, Google reported over 60 models of Android handsets launched by over 20 branded manufacturers. As of early 2011, all major handset manufacturers – except for Nokia – have launched Android-powered handsets, with Gartner predicting that Android will power over 300 million handsets sold annually. The phenomenal level of adoption is not only unprecedented by any measure, but beyond Google's wildest dreams when the very first Android handset was launched.

We next discuss Android's governance model.

Access

The Android software stack comprises Android-specific components (the platform), most of which are licensed under the Apache 2.0 license. The stack includes the Linux kernel and WebKit, which are under the GNU GPL plus LGPL licenses, respectively, and various minor components copyrighted by other owners.

The Apache 2.0 license is a permissive license that allows users to distribute modified versions of the code with no obligation to provide these changes back to the community.

Besides the public Android platform licensed under APL2, Google maintains a private code branch. The private branch is under development for the six-to-nine months prior to the release of the public code branch. The private branch is available to two arbitrarily chosen OEMs, who work closely with Google to develop the next Google-branded experience handsets.

As per standard mobile platform development practices, Google provides developers with access to mailing lists and a comprehensive suite of tools. Visibility to the roadmap is limited, as there is no Android roadmap publicly available. In fact, development of the Android private branch and the roadmap is controlled by Google, with little input from external parties or the Open Handset Alliance members.

In a major blow to openness advocates, Google announced in March, 2011, that they will not be providing public source code access to the latest version of Android, code-named Honeycomb, stating

"We have more work to do before we can deliver [Android] them to other device types including phones. Until then, we've decided not to release Honeycomb to open source"[13].

Development

Contributions to the Android codebase are encouraged by Google, although we understand that very few external contributions are actually "committed" to the Android codebase.

Google requires contributors to agree to an Individual Contributor License Grant or a Corporate Contributor License Grant. The agreements are similar in content and contain a copyright and patent license in favour of the Android Open Source Project Leads, aka Google. Source code contributions are verified and approved by "Approvers" and "Project Leads", all of whom are exclusively Google employees.

Google provides all the necessary tools and development environment for developers to contribute to the platform but clearly prioritises control and the commercial success of the platform over open governance.

There are no statistics available to show the number of non-Google contributions, nor how many are accepted by Google to the Android platform – nor can we objectively measure how long it takes for a contribution to appear in the Android code base, what determines which contributions make it into the code base, the arbitration process when there are competing contributions, or roadmap practices, as these processes are all closed.

Derivatives

Google tightly controls the Android platform and its derivatives, i.e., the make-up of the Android platform on commercial handsets. Device manufacturers must pass the Compatibility Definition Document (CDD) and Compatibility Test Suite (CTS) tests in order to be allowed use of the Android trademark, the Android Market or other important Google Mobile Services such as GMaps, Gmail and GTalk.

The CDD lists the minimum set of functionalities and technologies that an Android device must contain in order to use the Android trademark. Whilst the documentation acknowledges that components can hypothetically be replaced with alternate implementations, this practice is strongly discouraged, as passing the CTS tests will become substantially more difficult.

As such, devices that use the Android platform but have not passed the CTS – such as oPhone and Archos devices – are basically "derivative" products of Android OS, but cannot claim to be Android devices, use the Android trademarks nor access the Android Market.

Devices that pass the CTS can then "seek" approval to use the Android trademark and the Google Mobile Services, although the final criteria appear both undocumented and somewhat capricious.

As Google's Dan Morrill put it in an e-mail on Aug. 6, 2010, "We are using compatibility as a club to make them [OEMs] do things we want."

The Android Compatibility Program is the subject of a legal dispute between Skyhook Wireless and Google. In September 2010, Skyhook Wireless alleged that Google unfairly used the Compatibility Program process to force Motorola to remove Skyhook Wireless technology in favour of Google's own mapping technology, thus alleging unfair and deceptive business practices and intentional interference with contractual relations.

It is also worth noting that Google requires parties joining the Open Handset Alliance to sign up to an "anti-fragmentation" agreement, although we understand that the contents of that agreement are rather vague and reference primarily the Android Compatibility program. There are rumours that Google is now asking OEM licensees to also sign similar anti-fragmentation agreements, which has prompted complaints to the U.S. Department of Justice[14].

Community structure

When launched, the Open Handset Alliance served the purpose of a public industry endorsement for Android. Today, however, the OHA serves little purpose besides a stamp of approval for OHA members; there is no formal legal entity, no communication processes for members nor frequent member meetings.

All in all, Android is the most closed open source project, whilst also the most commercially successful mobile software platform to date.

Best practices: Android project

Best practices
Ease of source-code access via the Apache License
Ease of access to mailing lists, very good developer tools and forums
Simple code-contributions process for developers to follow
Clever targeting of developers via the Android Challenge, Summer of Code, etc.
Practices to avoid
Unilateral Android project decision-making processes, as Google determines the roadmap, feature-set and releases of Android
Closed code committer process, i.e., committers are exclusively Google personnel
Closed contributions process model
Opaque decision-making and control process around the Android Compliance Program
No project metrics around contributions, commits, contributors, top participants and bugs
No public information provided regarding meeting minutes or decisions.
No intention to move towards a more open governance model

Source: VisionMobile

Eclipse

Eclipse project	
Access	17/19
Development	15/18
Derivatives	6/6
Community	2/2
Open Governance Index	84%

The Eclipse Project was established by IBM in November, 2001, as a foundation for hosting IBM's $40M contribution in open source development tools, and supported by a consortium of over 80 software vendors[15]. Subsequently, the Eclipse Foundation (EF) was established in 2004 as an independent, not-for-profit corporation, to act as the steward of the Eclipse community. Today, all technology and source code produced through the Eclipse Foundation is made available under the Eclipse Public License (EPL).

The Eclipse Foundation hosts over 200 open source projects, most notable being the Eclipse IDE, which is today a *de facto* standard for developer tools. The Foundation provides IT infrastructure and marketing support, manages projects governance and also carries out intellectual property due diligence.

The Foundation employs around 15 staff, and is funded through annual membership contributions. In 2009, membership revenues were in the order of $2.7m, with an additional $300k arising from co-marketing agreements and donations. Current Eclipse membership includes IBM, Oracle, Nokia, Cisco, Motorola, RIM, Google, Intel, Sony Ericsson, Symbian and Adobe, amongst over 170 leading technology companies.

We next review the governance model of the Eclipse Foundation and the totality of projects it manages.

Access

Projects hosted by the Eclipse Foundation are licensed under the EPL license. The EPL requires all source code modifications and derivatives to retain the EPL license. However, the EPL is a weak copyleft license, in that EPL-licensed software can be combined with proprietary software, without passing on these same obligations. EPL also contains a reciprocal patent license. Overall, the EPL strikes a balance between sharing (reusing code with proprietary software) and contributions (contributing modifications back into the community).

The Eclipse Foundation explicitly states that it is open, transparent and meritocratic[16]:

'open' to everyone such that there are no rules to exclude any potential contributions

'transparent' such that all project discussions, minutes, project plans and roadmaps are open, public and easily accessible and

'meritocratic' such that the more you contribute, the more responsibility you earn.

All 200+ Eclipse Foundation projects use a consistent management structure. Each project has a development team, led by the PMC Lead (Project Leader). Every project comes with an extremely comprehensive information page detailing mailing lists, project leadership, committers (active, participating and inactive), bugs, releases and the project plan, among other information. This makes

the Eclipse Foundation the most "open" in terms of accessibility of information. In addition, the Eclipse Foundation provides "Project Dash" at www.eclipse.org/dash. Dash aims to provide complete transparency as to the contributions of all companies and developers participating at Eclipse.

Development

Eclipse projects are managed in accordance with the Foundation's development processes, which state how bugs, releases and roadmaps are managed, as well as detailing processes for dealing with conflicts and disagreements.

Project development is managed within each project team, while Foundation members have voting rights on the Eclipse roadmap, rather than on specific projects. Disagreements are only escalated as needed to the Eclipse Management Organisation, then the Executive Director and finally the Board. This makes Eclipse development an open, transparent and public process.

Contributions for code or documentation are evaluated using a two-step process; first, a technical evaluation at the discretion of the project committers, and second, an IP review (to check copyright provenance) when the contribution is greater than 250 lines of code. Whilst EF does not provide copyright warranty or indemnity related to the projects that it hosts, a large number of its members do, through the end products that are based on Foundation projects. All contributions are licensed to each project under the EPL license.

Committers have write access to all project resources, and are expected to influence the project's development. A committer must also complete a "Committer Agreement," which details the rights of the committer as well as his or her responsibilities in managing the code. New committers are elected through a voting process, which is based on three requirements: trust, public voting and employer neutrality. As of the end of 2010, there are nearly 1,000 active committers working on Eclipse projects.

Eclipse provides comprehensive statistics on distribution of committers by organisation; IBM is by far the largest organisation by commits, making up around 30% of all contributions to Eclipse projects, while other Eclipse members make up another 40% of contributions.

Derivatives

Each Eclipse project is free to determine its own implementation compliance and quality requirements. Note that compliance and quality requirements are not a pre-requisite to use of the Eclipse trademarks and logos; rather, members are entitled to use the Eclipse trademarks and logos provided they agree to the proper usage of Foundation-wide policies and guidelines.

Community structure

The Eclipse Foundation has developed an elaborate, comprehensive community structure. Each project team has a project leader, committers and contributors. Top-level projects are managed by project management committees (PMCs) which are overseen by the Eclipse Management Organisation.

Eclipse has a tiered membership structure where members can choose their level of voting rights, and monitoring/management tools, based on their interest in Eclipse Foundation projects.

Membership tiers: Eclipse Foundation

Membership benefits	Costs
Committer Can Commit Code to projects	Free
Associate Participate in projects and annual meetings	free to non-profits & universities otherwise US$5k per annum
Solutions For organisations incorporating Eclipse projects into products	$1,500 $20,000 per annum, scaling with company revenues
Enterprise Access to analytics on how developers use Eclipse. Access to detailed IP policies.	$125,000 per annum
Strategic Position on the Eclipse Foundation Board of Directors, providing direct influence over the strategic direction of Eclipse. Seat on the Eclipse Requirements Council driving Eclipse technology	Strategic Developers: 0.12% of revenue, minimum 8FT resources. Strategic Consumers: 0.2% of revenue.

Source: Eclipse

Best practices: Eclipse

Best Practices
Ease of source-code access via EPL
Ease of access to mailing lists, very good developer tools and forums
Use of not-for-profit foundation structure to provide a vendor-neutral and independent structure
Projects make all technical decisions, with guidance only from the Foundation
Thorough IP review provides IP copyright certainty to commercial users of Eclipse Foundation projects
Very clear and transparent single project reporting metrics
Practices to avoid
Contributions and Committers process can appear complex to outsiders and newcomers
Tiered membership such that Board seats are weighted in favour of Strategic members, although this is mitigated somewhat by the six seats elected by other members.

Source: VisionMobile

Linux kernel

Linux Kernel	
Access	14/19
Development	14/18
Derivatives	6/6
Community	2/2
Open Governance Index	71%

Since its inception in 1991, Linux has grown to become a major force in computing, powering everything from mobile phones and picture frames to Google servers and the New York Stock Exchange.

The Linux kernel comprises some 15 million lines of code developed by contributors worldwide. The kernel is at the core of 100+ software distributions (including Debian, Fedora, Red Hat, openSUSE, Ubuntu) and millions of computing devices.

The Linux kernel is supported by the Linux Foundation (LF), a non-profit foundation that sponsors kernel.org (the primary repository for the Linux kernel source code) and the work of primary Linux creator Linus Torvalds. LF was founded in 2007 by Fujitsu, Hitachi, HP, IBM, Intel, NEC, Novell, and Oracle, following the merger of the Open Source Development Labs (OSDL) and the Free Standards Group (FSG), and in October 2010 further incorporated the Consumer Electronics Linux Foundation (CELF).

The Linux Foundation hosts "workgroups" that include MeeGo, FOSSBazaar, Desktop Linux and Carrier Grade Linux, among others. Besides project hosting, LF provides legal programs, developer programs, regional programs and events to support the use of Linux globally.

We next review the governance model around the Linux kernel.

Access

Linux kernel source code is available from a number of sources, with the central 'tip of tree' repository at http://www.kernel.org/. The Kernel.org tree is led by Linus Torvalds, along with various Linux subsystem maintainers (the kernel code base is broken down into subsystems, like networking and memory management, with each subsystem having a designated maintainer).

Currently at version 2.6.x, the kernel gains a new release every 2-3 months. Each new release typically comprises over 10,000 code patches, including fixes, new features, internal API and ABI changes, and more.

Some meeting minutes and roadmaps are publicly available, but not all. It is important to note that the Linux Foundation is quite hands-off in terms of how the kernel.org project is managed; rather the LF provides support with regard to adoption, use and marketing of the kernel, as described above.

The Linux kernel mailing list (LKML) is the main electronic mailing list for Linux kernel development, where the majority of the announcements and discussions over the kernel take place. It is a very high volume list, usually receiving between 200 and 300 messages each day.

Development

Kernel development is managed via the Linux Foundation's Linux Developer Network (LDN), which works to ensure that applications created for Linux are supported across a variety of Linux distributions as well as promoting the betterment of Linux.

Code is contributed to the Linux kernel under a number of licenses, provided such licenses are compatible with the GNU GPLv2, which is the license covering the kernel distribution as a whole. Any contributions not covered by a compatible license will not be accepted into the kernel. Copyright assignments are not required (nor requested) for code contributed to the kernel. Moreover, all code merged into the mainline kernel retains its original ownership, meaning that the Linux kernel now has thousands of copyright owners.

Code contributions are reviewed and approved by the 900+ maintainers of kernel subsystems before being merged into the mainline tree. The commit process takes place during two-week-long "merge windows". At the end of this time, Linus Torvalds will declare that the window is closed and accept only patches into the kernel over the next six-to-10 weeks, after which the release becomes official.

According to a Linux Foundation study[17], over 60% of contributions to the kernel came from developers with corporate affiliations. Red Hat topped the chart with 12%, followed by Intel with 8%, IBM and Novell with 6% each, and Oracle with 3%. Further metrics regarding contributions are provided by the "Git statistics for the Kernel", including top contributors per technology domain, number of kernel developers and number of commits.

Linus Torvalds directs and controls the development of the kernel.org project in what is loosely termed a "benevolent dictatorship," since Torvalds has the final say on disputes or disagreements around kernel development. The Linux Foundation is not usually involved in the management of the kernel.org project.

Derivatives

Creation of derivatives is fundamentally impacted by the GPLv2 license, which states that all changes to the Linux kernel that are distributed must also be made publicly available. Applications can use the Linux ABI (application binary interface) and API (application programming interface) without themselves having to comply with the GPLv2, however. And, it's worth noting that Linux's available, nearly complete POSIX API allows applications to be written independently to the underlying kernel implementation. Such applications can be built and run on any POSIX-compliant system, including most commercial Unixes, Unix-like OSes like Linux and BSD, and many embedded RTOSes, such as LynxOS and QNX. Even Windows has some level of POSIX compliance, thanks to the Cygwin add-on tools and runtime.

Torvalds owns the "Linux" trademark, and monitors use of it mainly through the Linux Mark Institute and the Linux sublicense[18]. Whilst this practice is obviously a centralisation of power and authority over the development of the kernel, it appears to have worked thus far, but we would see this model as more the exception than the rule.

Distributions based on the Linux kernel and sold as products may have further trademark obligations.

Community structure

The Linux Foundation is the main organisation sponsoring and supporting kernel.org developments.

The Linux Foundation comprises three decision-making bodies: the Technical Advisory Board (helping the Foundation interact with the Linux community), the End User Council (for corporate end users) and the Vendor Advisory Council (where Foundation members discuss and collaborate).

The Linux Foundation has a number of tiered membership levels with increasing levels of influence:

Membership tiers: Linux Foundation

Membership benefits	Costs
Individual attendance at events	from $99 annually
Silver 1 seat on the Board of Directors	$5,000- 20,000 annually, scaling with number of employees
Gold up to 3 seats on the Board of Directors	$ 100,000 annually
Platinum up to 10 seats on the Board of Directors	$ 500,000 annually

Source: Linux Foundation

The Linux foundation lists Fujitsu, Hitachi, Intel, IBM, NEC, Oracle and Qualcomm as Platinum members with AMD, China Mobile, Cisco, Google, HP, Motorola, Nokia and Novell listed as Gold Members.

Best practices: Linux

Best Practices
Ease of source-code access via GPL and LGPL
Ease of access to mailing lists, developer tools, forums
Simple code-contribution process
Transparent project metrics around contributions, commits, top participants and bugs
Practices to Avoid
None

Source: VisionMobile

MeeGo

MeeGo was launched by Nokia and Intel in February, 2010, to much fanfare. At the outset, MeeGo was intended as an open source platform for powering Nokia's high-end devices and driving Intel's x86 chipset sales. Only a year later, MeeGo became a distant "plan B" for Nokia – following the Finnish OEM's refocus onto Windows Phone 7 – with MeeGo's main supporters now being Intel and LG.

MeeGo project	
Access	14/19
Development	12/18
Derivatives	5/6
Community	2/2
Open Governance Index	61%

MeeGo is a software distribution that comprises a number of mature existing open source projects. The distribution is formed from contributions by Intel (Moblin, including GTK) and Nokia (Maemo elements, but primarily Qt). MeeGo targets handsets, netbooks and in-vehicle infotainment devices. Nokia and Intel each have their own commercial platforms built out of the MeeGo distribution, with additional closed source components.

MeeGo is the only open source mobile platform that is managed and governed by the Linux Foundation – so as to ensure independence of governance and confer a sense of open source credibility to the project.

Besides providing governance, legal and marketing support, the Linux Foundation manages the MeeGo compliance program as an independent entity; however as MeeGo devices have yet to ship, it remains to be seen if compliance requirements will be used as a control mechanism.

Access

MeeGo is an open source project and governed by the Linux Foundation. Most of the access rights for the framework and the vast majority of source code (>90%) are under GPL v2 and GPL v3, as of July, 2010. MeeGo places no additional rules on top of the upstream component licenses. There is no MeeGo formal contribution license required; rather, MeeGo uses the Linux "signed off" process[19].

MeeGo has a recommended licensing policy for new components or code, i.e., "For the core OS, the licensing policy is that components must be under OSI-compatible licenses and enable linking of proprietary components or plugins. (L)GPL version 2.x is encouraged, and for the UX layer, permissive OSI licenses are encouraged".

MeeGo provides source code, developer tools, mailing lists, developer forums, etc., with no access fees or formal membership requirements. Release schedules and features are publicly available for review and roadmap information is available – albeit via Bugzilla, where new features/requests are assigned priority and given a proposed release date. Official releases are intended to be semi-annual, while daily builds are also available. It was intended that there will be a MeeGo Handset Working Group that will determine the roadmap and features to be included in new releases, etc., but in light of Nokia's move away from MeeGo, this is unlikely to occur. As reiterated by MeeGo, the roadmap will be heavily influenced by the upstream projects that comprise the MeeGo distribution, of which MeeGo is but one user/contributor. Presently these decisions are managed via discussions on the various mailing lists, and escalations are managed via the Technical Steering Group (TSG), which comprises Nokia and Intel

personnel. The TSG operates in an open and transparent manner, with meeting minutes publicly available.

Development

MeeGo code ownership is via the pre-existing upstream projects that feed into the project; i.e., Maemo is copyright Nokia, Moblin is copyright Intel and so on. These projects are then licensed under various open-source licenses. New code contributed to the project by developers follows the Linux 'signed off process,' which asks the contributor to state that the code is their own creation and legitimately free software. MeeGo actively encourages contribution to the upstream modules which comprise MeeGo, as opposed to MeeGo directly, and these upstream modules (such as Maemo, etc.) will have their own contribution process (and license), which may be different to the Linux "signed off" process.

Code contributions are committed to the project by about 20 paid committers, who are publicly listed and work for either Nokia or Intel. The process of becoming a committer is based on meritocracy and nomination, and is reported via the MeeGo community meeting minutes. As of October, 2010, the first non-Nokia/Intel committer had been appointed. Disputes regarding code contributions and acceptance are usually resolved at the developer level. Failing that, they will be escalated to the appropriate Working Groups and thereafter to the TSG.

Derivatives

MeeGo device and application compliance will be managed to ensure consistency across the MeeGo brand. MeeGo advise that there will be no costs associated with device compliance testing. The Linux Foundation will verify that organisations using the MeeGo Trademark are compliant with the MeeGo compliance specification. This process has yet to be initiated, so we cannot say how successful or transparent it is at this point. We understand that application compliance will be managed via the distribution channels, and that there may be additional compliance criteria required for applications beyond API compliance (most likely certifications at the Ovi store/Intel AppUp level).

Community structure

Regarding the MeeGo community structure, there are no admission processes, contracts, or membership fees for MeeGo. The MeeGo governance model is surprisingly transparent. On a monthly basis, Linux Foundation publishes a complete list of project statistics, including social media mentions, mailing list size, forum topics and activity, wiki/blog stats, upcoming technical group meetings, IRC activity and top participants, bugs, commits, and full visibility of top participants across mailing lists, forums, IRC, and bug lists[20].

Best practices: MeeGo

Best Practices
Ease of source-code access via open source licenses
Ease of access to mailing lists, developer tools, forums
Simple code-contribution process
Transparent project metrics re: contributions, commits, top participants, bugs, etc.
Independent management of compliance process to keep it transparent and honest
Policy of contribution to 'up-stream' projects first – fragmentation deterrent
Practices to avoid
Code commit privileges for Nokia/Intel personnel only as of November, 2010

Source: VisionMobile

Mozilla Foundation

The Mozilla Project was created in 1998 with the release of the source code for the Netscape browser under an open source license. In 2003, the Mozilla project created the Mozilla Foundation, an independent non-profit organization to manage the daily operations of the project. The Firefox 1.0 browser was released in 2004 and in less than a year, it had been downloaded over 100 million times. Today, Firefox market penetration is estimated at nearly 28% of total browser usage, as of May 2011.

Mozilla project	
Access	16/19
Development	11/18
Derivatives	6/6
Community	1/2
Open Governance Index	65%

The Mozilla Foundation owns two subsidiaries: first, the Mozilla Corporation, which employs around 400 staff, including about 200 Mozilla developers who manage the releases of the Mozilla Firefox web browser; second, Mozilla Messaging, Inc., a subsidiary that primarily develops the Mozilla Thunderbird email client.

The Mozilla Foundation is unlike any other non-profit foundation. The vast majority of its $80-million-plus revenues come from search royalties from Google, in return for making Google the default search engine within the Firefox search bar.

We next review the governance model used within the Mozilla Foundation projects.

Access

The Mozilla Foundation currently hosts 14 open source projects. This includes a set of core technologies (layout engine, networking libraries, cross-platform components) as well as applications built with those technologies (browser, mail reader, calendar, IRC client). The Mozilla Developer Network (MDN) provides free developer information, forums, FAQs and build tools for all Mozilla technologies. There are around 200 full-time Mozilla Corporation developers who are employed to develop these technologies.

Mozilla foundation provides project code under the "Mozilla tri-license," i.e., the MPL/GPL/LGPL triple license. Thus, the code can be licensed under the Mozilla Public License, version 1.1 or later (MPL); the GNU GPL, v2.0 or later (GPL) or the GNU LGPL, version 2.1 or later (LGPL). The reason that Mozilla provides the code under three different licenses is to ensure that the code is compatible with as many common open source licenses as possible.

It is worth noting that when the Mozilla Foundation moved to this tri-license approach in 2004, it took two years to track down all of the almost 450 contributors to the Mozilla code base and get them to agree to relicense their contribution using the tri-license approach. This underscores how it is imperative to get the right license strategy identified at the outset of any open source project.

Access to Mozilla project roadmaps and future code development is available via the Mozilla Foundation wiki. Project coordination meetings are held weekly and the minutes of these meetings are publicly available on the wiki. Additionally, there are project-specific build toolkits available to developers, along with various message boards, IRC channels, mailing lists, Google groups and forums.

Mozilla projects use the Mercurial source code management tool, to manage changes to the code. Bugs are tracked using the Bugzilla tool. Formal project releases are managed by personnel titled 'Release Drivers,' who are responsible for determining which patches are incorporated into which releases, and directing development efforts for the projects.

The Mozilla Foundation is currently redrafting the MPL license from v1.1 to v2.0. The update clarifies that trademark rights are not granted by the license, expressly reaffirms fair use rights, and allows contributors and distributors to add additional disclaimers of warranty and limitations of liability specific to a given jurisdiction.

Development

Contribution of code to the Mozilla Project is governed by the Mozilla Foundation Committer's Agreement v2.0, which states that all code contributed must be licensed to the project under the Mozilla tri-license. The process for contributing code is straightforward and clearly documented. All a contributor needs to do is create a code patch and forward it to the module owner responsible for reviewing and committing the code change. Patches that cross modules, change APIs or need security-related changes must pass a "super review".

The process to become a "committer" is clearly outlined. At the same time, there are no statistics on committers, their numbers or their commercial affiliations, making it difficult to ascertain the number of committers outside the Mozilla Corporation itself. Additionally, there is no public information regarding the number of contributions, where contributions come from, and which contributions are actually committed.

Derivatives

Creation of derivatives is permitted by the Mozilla tri-license. However, use of the Mozilla trademarks (such as the "Firefox" name and graphics) is permitted only if you are distributing unchanged binaries. If you change any of the source code at all, then you are not permitted to use the Mozilla trademarks.

As a result, Mozilla-derived browsers must be rebranded. Popular Mozilla-based browsers include the Flock Social Web Browser, Swiftfox, Debian Iceweasel, GNU Icecat and the Songbird Audio Player and Browser. However, these are little-used, relative to Firefox[21].

To allow distributions of the code without official Firefox branding, the Firefox source code config file contains a "branding switch". This switch allows the code to be compiled without the official logo and name, for example to produce a derivative work unencumbered by restrictions on the Firefox trademark. In the unbranded compilation, the trademarked logo and name are replaced with a freely distributable generic globe logo, and the name of the release series from which the modified version was derived. A healthy market has formed around the creation of applications, called "Add-on's", for Firefox. To create an Add-on, developers must sign the "Developer Agreement". This agreement provides a copyright license to Mozilla for use of the Add-on, and exonerates Mozilla for any infringement issues that might arise from use of the Add-on.

There are no other formal compliance requirements for use of the various Mozilla projects.

Community structure

Membership to the Mozilla Project is free, subject to signing the Committers Agreement.

Mozilla states that it is an "open source project governed as a meritocracy, a virtual organization where authority is distributed to both volunteer and employed community members as they show their abilities through contributions to the project."[22]

Governance of the πroject is managed by module owners who are assigned to technical or administrative management tasks. Module owners are responsible for code maintenance, managing conflicts relating to code contributions and determining policy with regard to licensing and trademarks.

All module owners are publicly listed, including a number who are not Mozilla employees, implying that external developers and organisations do have influence within the Mozilla Project. Additionally there are Google Groups listed for all modules that are publicly available to read and review. In the case of conflicts and disputes, these are judged by one of two Mozilla "benevolent dictators" – Brendan Eich for technical disputes and Mitchell Baker for non-technical disputes. Both have held this position since the formation of Mozilla as an 0pen source project in 1998.

Best practices: Mozilla Foundation

Best Practices
Ease of source-code access via multiple license options
Ease of access to mailing lists, developer tools, forums
Simple code-contributions process for developers to follow
Use of non-for-profit Foundation to provide a vendor neutral and independent structure
Practices to avoid
Unclear as to who committers are
No project metrics re: contributions, commits, top participants or bugs.

Source: VisionMobile

Qt

Qt is a cross-platform application framework for developing applications across desktop, embedded and mobile devices. Qt (pronounced “cute”) was created in 1991 by Trolltech ASA, and acquired by Nokia in June, 2008. More than 100 million Qt devices have shipped to date, according to Nokia.

Qt project	
Access	16/19
Development	8/18
Derivatives	6/6
Community	2/2
Open Governance Index	58%

Following two years of integration work, Nokia released the Qt SDK 1.0 in June 2010, and announced Qt as the primary development environment on MeeGo and Symbian.

The Microsoft-Nokia strategic deal announced in February, 2011 dislodged MeeGo and Symbian – and with it Qt – out of Nokia’s smartphone roadmap. Moreover, Nokia sold the commercial licensing arm of Qt to Digia, who is now licensing and supporting Qt customers outside mobile.

In a June, 2011 analyst communication, Nokia promised to “make Qt core to building applications that connect the next billion users to the Internet", thereby hinting at a role for Qt in Nokia’s mass-market phone range.

We next review the governance model of the Nokia Qt application environment.

Access

Through the years, the Qt platform has been offered under various open-source and proprietary licenses. Initially, Qt was widely used in KDE, the first fairly complete desktop software collection for Linux. However, Qt’s proprietary license worried many in the free software community, and in response, the GNOME project was founded. GNOME opted for the fledgling GTK (GIMP toolkit), which had been created as an alternative to Motif, another proprietary GUI toolkit used by Netscape and early commercial Unixes. Trolltech eventually added a GPL license option to Qt, easing fears it would somehow try to take control of KDE. However, the move arguably came too late, for today, GNOME has surpassed KDE in popularity.

Later, when Qt began targeting embedded devices, Trolltech licensed the platform under a dual licensing regime; i.e., a commercial version of Qt under a proprietary license, and an open source version under the GPL v2 (copyleft) license. The commercial license option was intended to give organisations certainty as to license obligations and restrictions, and provide clearer patent indemnities and warranties.

With the Trolltech acquisition, Nokia continued with such a licensing strategy, but also released Qt under the LGPLv2 license, in March, 2009. This weaker copyleft license allows third parties to link their software to the Qt platform without inheriting the GPLv3’s copyleft requirement to publish all source code.

The LGPL also makes it easier for companies in the mobile software sector to use Qt, given that use of copyleft licenses is somewhat rare amongst mobile handset OEMs.

Therefore this strategy does have its benefits to organisations. However, the use of such a dual and now triple licensing strategy is criticised by some members of open source communities, who argue that such a strategy is counter to the "freedoms" prescribed by the FSF. There is also a sense for some that no matter how "free" Qt becomes, it can never fully live down its proprietary software roots.

Qt source code is available via Gitorious, a community-oriented source code repository. We understand that Gitorious is updated constantly with commits typically visible to the outside world within 60-75 minutes of submission. All Qt releases are available on the Gitorious repository, with no difference between the commercial version and the open source versions.

In June, 2010, Nokia announced a move towards a more transparent governance model, designed to open up technical and product discussions to the public, and give the community access to the QA and integration process[23]. At that point, the intention was to provide more transparency in the roadmap decision-making process (and participation in it), and the ability for other organisations to create their own roadmaps.

Qt provides a comprehensive developer forum, mailing list and suite of developer tools at no cost to developers.

Development

Contributors to Qt need to sign an agreement granting copyright and patent licenses to Nokia. This is a major improvement from the previous heavy-handed contributor license, which required that copyright holders assign all copyright to Nokia.

However, the process for reviewing, approving and committing contributions to the Qt platform is still controlled by Nokia personnel only.

The contribution process on the Qt website advises that the time it takes to approve contributions depends on many factors, including the size of the contribution, its complexity and the availability of a technical review. In practice, the time taken to process submitted contributions is around two weeks.

Derivatives

Rights to create derivative products from Qt stem from the license agreement under which the developer uses Qt: i.e., GPLv3, LGPLv2 or the Qt Commercial License. Therefore, branching rights are provided, but as per the GPL and LGPL, any changes made to the code must be published.

Nokia does not appear to use trademarks as a tool to achieving compliance for Qt – unlike Android or Java ME. We believe this is because Nokia is not interested in Qt as a licensable platform, but as a means to create an application ecosystem on its own devices.

Community structure

The Qt community is loosely structured. There are no formal membership agreements nor any tiered structure. Even though all decision-making regarding the Qt project is currently managed by Nokia personnel, we note that this is intended to change in 2011.

Best practices: Qt

Best Practices
Ease of source-code access via GPL and LGPL
Ease of access to mailing lists, developer tools, forums
Simple code-contribution process
Practices to avoid
Unilateral Qt project decision-making processes via Nokia
Committer privileges restricted to Qt/Nokia personnel
Closed contributions process model
No project metrics
Little transparency regarding how decisions are made and no public information provided on this

Source: VisionMobile

Symbian

Note: The Symbian open source project is analysed here for reference purposes only. Following Nokia's partnership with Microsoft, Symbian has become a closed-source, Nokia-only platform, and is scheduled to be discontinued.

Open Governance Index	
Access	17/19
Development	10/18
Derivatives	4/6
Community	1/2
Open Governance Index	58%

Symbian Software Ltd. was established in June, 1998, as a joint venture between Ericsson, Nokia, Motorola, and Psion. Symbian quickly rose to become the best-selling smartphone platform for most of the decade. The beginning of the end came in June, 2008, when Nokia announced that they would fully acquire Symbian and release the platform under an open-source license. The Symbian Foundation was tasked to manage the merge of the Symbian OS with the S60 application platform, which was eventually released to developers in February, 2010. With Symbian an aging platform and losing to Android both in terms of market share and developer mindshare, Nokia had no option but to replace Symbian with Microsoft's Windows Phone 7, in a move whose ripples are still reverberating through the mobile industry.

The Symbian platform comprises three layers (OS, middleware and applications) and a total of 180 packages (code components). Each component was assigned to one of 12 technology domains, with each technology domain having its own roadmap.

We next review the governance model of the Symbian Foundation, for reference purposes.

Access

Up until the March, 2011, Symbian source code was available to download under the Eclipse Public License (EPL) and the Symbian Foundation License (SFL).

The Eclipse Public License is a weak copyleft license mandating that all Symbian platform source code modifications and derivatives must be licensed under the EPL, but can be combined with proprietary software without passing on these same obligations. The EPL mandates derived works to also carry the EPL License, which has limitations in terms of code reusability, and lacks compatibility with LGPL and GPL, under which most open source software is licensed.

Symbian code was also available under the Symbian Foundation License (SFL), but restricted to corporate members who paid a membership fee of $1,500. Corporate members had to sign the Membership Agreement, a copyright license for use of the Symbian platform. The SFL license also provided a FRAND (fair, reasonable and non-discriminatory) reciprocal patent license, i.e., a stronger patent protection than that provided in the Eclipse Public License to non-members.

The Symbian platform roadmap was made available publicly with the release of Symbian^3 in June, 2010.

Symbian Foundation had set up a large number of public forums (one per package), with over 25,000 registered users, but only 1,500 active users.

Development

Developers wanting to contribute to the Symbian platform had to sign a contribution agreement which provided a copyright and patent license agreement in favour of the Symbian Foundation. Package owners were then responsible for reviewing and committing these changes to the Symbian platform. Major contributions such as new platform features had to be approved by all four Symbian Councils before being integrated into the main codeline. Committers were restricted to those coming from organisation members of the Symbian Foundation.

New features to the platform were managed through the 'proposals pipeline,' and had to be approved via the four Symbian Councils, i.e., the Architecture, Features & Roadmap, UI and Release Council. Council members were usually appointed either by handset OEMs or optionally via an open election process. Council members met monthly and meeting minutes were publicly available.

The Symbian Foundation governance structure was extremely formal relative to other open source communities, and was also heavily weighed in favour of "appointed" handset OEM members, who comprised Nokia, Sony Ericsson, Samsung and Fujitsu.

Whilst the Symbian Foundation was "open" to developer contributions, it is clear that only Symbian Foundation members were able to influence the direction of the platform. More importantly, only "appointed" handset OEM members were able to wield any significant leverage over the platform.

We understand that OEM members were essentially funding operational and staff costs of the Foundation in the form of fees (running in the millions of dollars) for the right to ship devices based on the platform.

The effective lack of influence by non-member developers was what led to the creation of the Symbian Developer Cooperative (DevCo). This was set up by the Symbian Foundation in July 2010, to give individuals a say in the governance of the Symbian platform. Additionally the DevCo nominated members for seats on the four Symbian Foundation Councils.

Derivatives

Symbian Foundation had established a formal platform compliance process, the Foundation Test Suite, which was mandatory in order to use the Symbian trademark. Unfortunately, little public information was provided about this compliance process on the Foundation website. Separately, applications created to run on the Symbian platform had to be "Symbian signed," and there are some minimal costs required as part of this signing process – $200 per annum for a publisher ID, and around $150 for formal testing of an application on each version of the platform. Additionally developers then had to pass the Nokia signing criteria via the Ovi marketplace, which made for a relatively onerous process for developers.

Community structure

Membership to Symbian was restricted to established companies, thereby excluding independent developers. Members had to agree to the membership agreement and the Symbian Foundation Deed of

Adherence, which detailed membership rules, how formal membership disputes were managed and Symbian Foundation Trademark Guidelines. Membership fees were set at $1,500.

Membership provided the ability to influence the evolution of the Symbian platform, through the right to vote in annual elections, the opportunity to become a package owner, and eligibility for seats on the Symbian Foundation board and the technology Councils. It is worth noting that the community structure was very much focused on business-to-business events rather than engaging with the wider community at a developer level.

Best practices: Symbian

Best Practices
Ease of source-code access via Eclipse Public License
Ease of access to mailing lists, some developer tools, forums
Use of a not-for-profit foundation structure to provide independence to platform organisation
Transparent tactical decision-making process via open publication of meeting minutes, councils and committee members, decisions taken, etc.
Practices to avoid
Neglected to target developers, focussing on industry organisations only
Complex contributions process and structure
Tiered membership such that strategic decision-making still retained by a small group of Foundation members
Bureaucratic decision-making process, i.e., four Councils required to approve new roadmap features
Closed Code Committer process, i.e., Committers had to be Symbian Foundation members
Historical lack of good developer tools
No project metrics re: contributions, commits, contributors, top participants, bugs, etc.

Source:VisionMobile

WebKit

WebKit	
Access	15/19
Development	12/18
Derivatives	6/6
Community	2/2
Open Governance Index	68%

WebKit is an open source HTML rendering engine based on the source code of KDE's KHTML rendering engine and further developed by Apple. In addition to being used in Apple's Safari browser and on the iPhone, WebKit has been ported to Symbian, Qt, Android, Chrome OS, BlackBerry OS, Nokia Series 40 and Qualcomm's BREW platform. WebKit is now the *de facto* engine for smartphone browsers, having been shipped in more than 500 million devices to Q1, 2011, by all major smartphone vendors.

WebKit is a longstanding open source project. It originates from KHTML, a part of the KDE 2.0 open source project released in 2000. Apple then took a fork of KHTML and worked privately on its own branch of it for years, producing WebKit for the purposes of powering Safari. In 2003, Apple released the first Safari code, including the modifications to KHTML. It wasn't until 2005 that Apple launched the full WebKit open source project, but restricted reviewer and commit rights to Apple personnel only. During this early stage of the WebKit development, there was much disagreement between the KDE community and Apple related to the direction and content of WebKit, as well as the control of WebKit via restricted commit rights, which effectively sidelined the KDE developer community contribution to the project. Eventually, in late 2007, Apple responded by updating the WebKit committer and reviewer policy to allow non-Apple developers to have full commit access to the WebKit source code version control system.

To date, WebKit acknowledges contributions from a number of major commercial organisations, in particular Apple, Google, Nokia, TorchMobile (now part of RIM) and Collabora (responsible for the GTK port).

Access

The WebKit JavaScriptCore and WebCore components are available under the GNU LGPL v2.1, while the remainder of the browser engine is available under a BSD-style license. A mature open source project, WebKit has a straightforward governance structure. Source code is freely available via a public Subversion repository, with the code being refreshed nightly. Bugzilla is used for issue reporting and logging bugs. Mailing lists and forums are freely accessible, along with developer build tools.

The WebKit roadmap is a loose, unordered collection of future development requests. There is no formal process for prioritising features; rather, contributors will focus on their own priorities for development. Structural changes to WebKit are guided by the "Project Goals", a public statement of what WebKit is and is not. There are no formal releases of WebKit, either; rather, there are numerous branches that contain product-specific implementations of WebKit that are maintained by the sponsoring organisation.

Development

Code contributions do not require signing of a formal contributions license. At the same time, contributions are required to include a copyright ownership notice, while a suggested licensing text features clauses on copyright ownership and redistribution, but no warranty provisions.

Contributions to WebKit are reviewed by project committers, who may grant or deny approval. Committers have direct read-write access to the WebKit Subversion repository, enabling them to commit changes once reviewed. There is also a public list detailing over 200 Committers from Apple, Nokia, Google and other organisations.

The process to become a WebKit reviewer and committer is clearly and openly documented, and operates through meritocracy. It is based on a nomination system and the developer's contributions history and collaboration history. WebKit reviewers are appointed regularly, and come from diverse backgrounds, projects and organisations.

Derivatives

There are no official compliance requirements for WebKit-based browsers. Rather, every new implementation team relies heavily on existing implementations to understand the inner workings of WebCore and JavaScriptCore. The WebKit community employs a huge testing infrastructure called "Layout Tests", which all implementers use to self-check their derivative implementation.

Community structure

Unlike Eclipse, MeeGo and Mozilla, WebKit does not have any formal councils, community organisations or steering groups. Rather, there are several developers who are acknowledged as experts, and who influence the direction of WebKit. In their main, these developers work for Apple and Google, so naturally these organisations have much influence over the direction and roadmap of WebKit.

Best practices: WebKit

Best Practices
Ease of source-code access via LGPL and other approved Open Source licenses
Ease of access to mailing lists, developer tools, forums
Simple code-contribution process
Practices to avoid
No project metrics
Little transparency regarding how decisions are made, and no public information provided on this

Source: VisionMobile

C. The Open Governance Index

Measuring Openness

We set out this report with an ambitious goal: to measure "openness", i.e., how "open" or "closed" an open source project is in ways that are rarely discussed publicly or covered in its license. In other words, our goal has been to define and measure the governance of open source projects in a transparent and comprehensive manner – much like how open source licenses are defined and classified into "copyleft", "permissive," and so on

Unlike open source licenses, the governance model is made up of less visible terms, conditions and control points that determine access, influence, decisions and spin-offs of that project.

We researched eight mobile open source projects: Android, MeeGo, Linux, Qt, WebKit, Mozilla, Eclipse and Symbian. We talked to community leaders, project representatives, academics and open source scholars to understand "how" governance works in those projects, and to measure and identify best practices.

In this report we propose the Open Governance Index, a measure of open source project "openness". The Index comprises 13 metrics across the four areas of governance:

1. Access: availability of latest source code, developer support mechanisms, public roadmap, and transparency of decision-making

2. Development: the ability of developers to influence the content and direction of the project

3. Derivatives: the ability for developers to create and distribute derivatives of the source code

4. Community: a community structure that does not discriminate between developers

We ranked projects across each governance parameter, and on a scale of one to four on each question. The higher the score, the more open the project.

The Open Governance Index quantifies how open a project is in terms of transparency, decision-making, reuse and community structure.

Open Governance Index (% open)	
Android	23%
Qt	58%
Symbian	58%
MeeGo	61%
Mozilla	65%
WebKit	68%
Linux	71%
Eclipse	84%

"Open governance" goes hand-in-hand with "open source", as it is about ensuring that developers and users have equal freedoms not to just use, but also to modify and build on the project. In many ways, open governance is the missing piece the open source licenses do not cover. We hope our research is a step towards a fundamental change in public understanding and transparency of the use of open source.

Governance Criteria **Access**	**Ranking options**	**Android**	**Eclipse**	**Linux**	**MeeGo**	**Mozilla**	**Qt**	**Symbian**	**WebKit**
Is source code freely available to all developers, at the same time?	4. Yes 3. No – discriminates with regard to either a. developers, b. source code or c. time 2. No – discriminates with regard to two of the above 1. No – discriminates with regard to all of the above	1	4	4	4	4	4	4	4
Is source code available under a permissive OSI-approved license?	4. Yes – approved license and permissive (e.g. Apache, BSD, MIT) 3. Yes – approved license and weak copyleft (e.g. Eclipse Public License, GNU LGPL v2/v3) 2. Yes – approved license and strong copyleft (e.g. GNU GPL v2/v3) 1. No – unapproved licensed/proprietary license	4	3	2	2	3	3	3	3
Developer support mechanisms – Are project mailing lists, forums, bug-tracking databases, developer documentation and tools available to all developers?	3. Yes – developer support mechanisms open to all developers 2. No – developer support mechanisms are limited, e.g., access to bug-tracking dbase not provided by Android 1. No – there are poor developer support mechanisms	2	3	3	3	3	3	3	3
Is the project roadmap publicly available?	4. Yes – full roadmap available, with explicit call for contributions to the roadmap 3. Yes – roadmap information available but no call for contributions or similar 2. No – No formal roadmap exists, but there are committer or contributor requests to bugzilla 1. No	1	3	2	1	3	4	3	2
Transparency of decision mechanisms – Are project meeting minutes publicly available to understand decision-making in the project?	4. Yes 3. Yes – there is some information but it is hard to find and doesn't appear comprehensive 2. No – but the intent is to provide more information and make the process more open 1. No	1	4	3	4	3	2	4	3
	Total	9	17	14	14	16	16	17	15

Governance Criteria **Development**	Ranking options	Android	Eclipse	Linux	MeeGo	Mozilla	Qt	Symbian	WebKit
Transparency of contributions and acceptance process – Is the code contribution and acceptance process clear, with progress updates of the contribution provided (via Bugzilla or similar)?	4. Yes – contributions and acceptance process are clear, with progress status of contributions provided 3. Yes – contributions process and acceptance process are clear, but no progress status of contributions provided 2. No – contributions process only, with progress status of contributions provided 1. No – contributions process only, no progress status of contributions provided	1	2	2	2	2	1	1	2
Transparency of contributions to the project – can you identify from whom source code contributions are provided?	4. Yes – there are good project statistics that provide this information 3. Yes – but you must manually find and collate the information from various project sources 2. No – although you may be able to find this information by checking the copyright notices attached to each file/contribution 1. No	2	4	4	3	2	2	2	2
Accessibility to become a committer – are the requirements/process to become a committer documented and is this an equitable process, i.e., can all developers potentially become committers?	3. Yes – the process is documented and accessible to all developers 2. No – the process is vague/unclear so we do not know if it is accessible to all developers 1. No – commit access is restricted to specific users/members of the Project only	1	3	3	2	3	1	2	3
Transparency of committers – can you identify who committers to the open source project are? i.e., those developers that have the authority to 'commit' source code to the baseline	3. Yes – there are good project statistics that provide this information 2. Yes – but you must manually find and collate the information from various project sources 1. No – this information is not provided	1	3	3	3	1	1	2	3
Does the contribution license require a copyright assignment, or copyright license and/or patent license?	4. Yes – project requires a copyright assignment and patent grant 3. Yes – project requires a copyright license and patent grant 2. Yes – project requires a copyright license/'sign-off' process 1. No – no contribution license	3	3	2	2	3	3	3	2
	Total	**8**	**15**	**14**	**12**	**11**	**8**	**10**	**12**

Governance Criteria **Derivatives**	Ranking options	Android	Eclipse	Linux	MeeGo	Mozilla	Qt	Symbian	WebKit
Are trademarks used to control how and where the platform is used via enforcing a compliance process prior to distribution?	2. No – You can freely distribute to the code and use the project trademark without completing formal compliance requirements 1. Yes – code must go through a formal compliance process prior to be distributed to other parties	1	2	2	1	2	2	1	2
Are go-to-market channels for applications derivatives constrained by the project in terms of approval, distribution or discovery?	4. No 3. Yes – restricted by approval, distribution or discovery 2. Yes – restricted by two or more of approval or distribution or discovery 1. Yes – restricted by all, i.e., approval, distribution and discovery	2	4	4	4	4	4	3	4
	Totals	**3**	**6**	**6**	**5**	**6**	**6**	**4**	**6**

Governance Criteria **Community**	Ranking options	Android	Eclipse	Linux	MeeGo	Mozilla	Qt	Symbian	WebKit
Is the formal community structure flat or tall, i.e., tiered rights depending on membership status	2. No – there is no formal membership or discrimination between the rights of members and non-members from a development/access perspective 1. Yes – there are tiered rights depending on membership status	1	2	2	2	1	2	1	2
Totals across all governance criteria		**21**	**40**	**36**	**33**	**34**	**32**	**32**	**35**
Open Governance Index % (rebased on lowest score 14=0%, highest score 45=100%)		**23**	**84**	**71**	**61**	**65**	**58**	**58**	**68**

Are "open" projects more successful?

Our research suggests that platforms that are most open will be most successful in the long-term. Eclipse, Linux, WebKit and Mozilla each testify to this.

In terms of openness, Eclipse is by far the most open platform across access, development, derivatives and community attributes of governance. It is closely followed by Linux and WebKit, and then Mozilla, MeeGo, Symbian and Qt. Seven of the eight platforms reviewed fell within 30 percentage points of each other in the Open Governance Index.

Our research has identified certain attributes that successful open source projects have. These attributes are: timely access to source code, strong developer tools, process transparency, accessibility to contributing code, and accessibility to becoming a committer. Equal and fair treatment of

developers – "meritocracy" – has become the norm, and is expected by developers with regard to their involvement in open source projects.

We also note that there are common areas where most open source projects struggle to be "open". These attributes coalesce around decision-making with regard to the project roadmap and committing code to the project. In particular, we find that open source projects that originate from commercial organisations struggle most with relinquishing project control, not surprising considered the structured and hierarchical decision-making nature of most organisations.

The Android paradox

Android ranks as the most closed project we examined, with an Open Governance Index of 23%. Yet, at the same time, it is one of the most successful projects in the history of open source. Is Android proof that open governance is not needed to warrant success in an open source project?

Android's success has little to do with the open source licensing of the public codebase. Android would not have risen to its current ubiquity were it not for Google's financial muscle and famed engineering team. Android platform development has occurred without the need for external developer or commercial community involvement – as we discussed earlier, the OHA has been a stamp of approval, not a distribution channel.

> "Android is, hands down, the most successful Linux distribution ever produced"... "Android is a poster child for how one should not work in the open source community"
>
> James Bottomley,
> Director on the Board of the Linux Foundation and Chair of its Technical Advisory Board
> speaking at LinuxCon Japan 2011

Google has provided Android at "less than zero" cost, since its core business is not software or search, but driving ads to eyeballs. As is now well understood, Google's strategy has been to subsidise Android such that it can deliver cheap handsets and low-cost wireless Internet access in order to drive more eyeballs to Google's ad inventory.

More importantly, Android would not have risen were it not for the billions of dollars that OEMs and network operators poured into Android in order to compete with Apple's iconic devices. As Stephen Elop said in June, 2011, "Apple created the conditions necessary for Android".

However, there are some very good lessons for us to learn from how Google has managed the Android open source project. First, Android was released as an open source project at a point in time where it was already a very advanced, complete project. OEMs, operators and software developers could more or less immediately use it to create derivative handsets and applications. Second, Google kickstarted a developer buzz around the project with the $10 million Android Developers Challenge. Alongside financial incentives, Google provided a very strong emotional message: that of opening application development within a previously inaccessible mobile industry. Finally, Google's speed of innovation (five platform versions across 2010) outpaces any external innovation, and makes the ecosystem entirely reliant on Google.

Best Practices

Based on our research of major mobile open source projects, we have outlined the best practices for governance models. These practices are listed across the four key areas of governance: access, development, derivatives, and community.

> "Apple created the conditions necessary for Android"
>
> Stephen Elop
> CEO, Nokia
> Speaking at the Open Mobile Summit, 8 June, 2011

Access

The minimum requirement for any project to be an open source project is *source-code access* such that developers can easily read, download, change and run the code. There should be no developer discrimination, in that all source code should be available to all developers in a timely manner. Restrictions with regard to source-code should be at a minimum, and there should be no preferential access to specific developers, as this can cause friction and lead to branching of the project. All open source projects should use open source licenses that are approved by the Open Source Initiative (OSI).

The next most important requirement is *ease of access to developer tools, mailing lists, and forums,* such that developers can get up to speed on the specifics of the project, and build and run the code with minimum effort.

Development

As much as possible, a *simple code-contributions process* should operate, such that the contributions process is as free and unhindered as possible. Whilst we appreciate valid Intellectual Property (IP) concerns, such as the risk of copyright infringement, these should not complicate the contributions process any more than necessary. We also note that none of the projects reviewed in this paper mandate copyright assignment, and we believe that this is a good example of why copyright assignment is largely unnecessary. A broad copyright (and ideally patent) license for use of the work should suffice, provided the project has researched and identified the appropriate open source license under which to distribute the project. Copyright assignment is only ever needed when the project decides to change the terms under which it licenses the source code of the project, and this should be largely unnecessary, provided that the correct open source license is identified in the first place.

Given that the success of open source projects is largely based on the accrual of developer interest and support, we identify the *transparency of decision-making* and *equitable treatment of all developers* (such that they can become project committers) as being critical to long-term project success. Restriction of commit rights to specific developers or organisations is a sure way to lose developer support in the long run, as developers become frustrated with the inability to commit code themselves, especially if their contributions are continually rejected or ignored.

Developers often need to know where, how and why the project is headed, as well as wanting the opportunity to influence the project to meet their own needs: i.e., to 'scratch their own itch,' in open-source-speak. The main means by which developers can achieve this influence is by being able to commit code to the project. Therefore, it should be possible for all developers to commit code to the

project, once they have shown sufficient knowledge of the code to do so. This is where *meritocracy* comes into play, in that those that 'do' are rewarded accordingly. Additionally the project should provide *transparent project metrics* regarding where contributions come from, and who committers are.

With regard to the actual development process itself, the project should have a *policy of contribution to 'up-stream' projects first* (if the project comprises other open source projects) such that changes and benefits accrue to up-stream and down-stream projects.

Derivatives

Compliance frameworks are becoming more and more common among open source projects, in order to deter fragmentation and ensure that applications are transferable across multiple platforms or operating systems. However, the best mechanism to keep compliance requirements honest to is make *the compliance process as independent and transparent as possible* such that it cannot be manipulated by any one developer or organisation. For example, MeeGo has asked the Linux Foundation to manage its trademark compliance requirements, so that they are independent of the project.

Community

A number of projects we reviewed use a not-for-profit foundation structure to provide independence, such that the platform is not controlled by any one organisation. Alternatively, other projects have established a formal association with the Linux Foundation, and this lends strong 'open source' credibility to the project.

Another aspect of open source communities are how authority is exercised within the community. For example, we note that both Linux and Mozilla use the benevolent dictator model, where decisions regarding disputes are made by one person. Whilst this process may work, it is still centralisation of authority and decision-making, and as such does not easily allow for others to permeate this decision-making process.

Clearly, an open source license alone does not make an open project. It takes an open governance model as well. We hope this report helps more open source projects adopt the right governance model for success.

Endnotes

[1] http://upload.wikimedia.org/wikipedia/commons/1/14/Bill_Gates_Letter_to_Hobbyists.jpg extracted March 2011
[2] Estimating the Total Development Cost of a Linux Distribution, October 2008, The Linux Foundation, http://www.linuxfoundation.org/sites/main/files/publications/estimatinglinux.html, extracted April 2011
[3] http://www.limofoundation.org/images/stories/pdf/limo%20economic%20analysis.pdf, Mobile Open Source Economic Analysis, LiMo Foundation White Paper dated August 2009, extracted March 2010
[4] http://www.gnu.org/gnu/linux-and-gnu.html extracted March 2011
[5] http://www.gnu.org/philosophy/open-source-misses-the-point.html dated 01 Nov 2010
[6] http://www.opensource.org/docs/osd, The Open Source definition, extracted January 2011
[7] Open Source Institute, viewed January 2011, http://www.opensource.org/licenses/alphabetical
[8] Microsoft Announces Patent Agreement With HTC, http://www.microsoft.com/presspass/press/2010/apr10/04-27mshtcpr.mspx, April 2011
[9] Nokia Wins Apple Patent-License Deal Cash, Settles Lawsuits, http://www.bloomberg.com/news/2011-06-14/nokia-apple-payments-to-nokia-settle-all-litigation.html, June 2011
[10] Apple Patent Infringement number 2, attacks HTC, http://www.fonehome.co.uk/2011/07/13/apple-patent-infringement-no-2-attacks-htc/ July 2011
[11] In Settlement, Nokia will pay Royalties to Qualcomm, http://www.nytimes.com/2008/07/24/technology/24qualcomm.html, July 2008
[12] Joel West and Siobhán O'Mahony, "The Role of Participation Architecture in Growing Sponsored Open Source Communities," Industry & Innovation, 15, 2 (April 2008): 145-168
[13] Google keeping Honeycomb source code on ice, says it's not ready for other devices, http://www.engadget.com/2011/03/24/google-keeping-honeycomb-source-code-on-ice-says-its-not-ready/ viewed March 2011
[14] [14] Do Not Anger the Alpha Android, http://www.businessweek.com/magazine/content/11_15/b4223041200216.htm accessed April 2011
[15] http://www.linuxfordevices.com/c/a/News/IBM-makes-40M-open-source-donation-CNET/ accessed April 2011
[16] Eclipse Development Process http://www.eclipse.org/projects/dev_process/development_process_2010.php#2_1_Open_Source_Rules_of_Engagement accessed November 2010
[17] Linux Kernel Report 2010 http://www.linuxfoundation.org/docs/lf_linux_kernel_development_2010.pdf accessed November 2010
[18] Linux Trademark sublicense Agreement, http://www.linuxfoundation.org/programs/legal/trademark/sublicense-agreement, accessed March 2011
[19] MeeGo Signed-Off Process http://meego.com/about/contribution-guidelines/signed-process.
[20] MeeGo Community Health Metrics, http://wiki.meego.com/Metrics#Community_Health_Metrics, June 2011
[21] 5 Firefox Based Browsers You Probably Haven't Seen Before, http://www.techdrivein.com/2010/06/5-firefox-based-browsers-you-probably.html accessed June 2011
[22] Mozilla.org https://wiki.mozilla.org/Mozilla.org/About accessed November 2010
[23] Qt and Open Governance http://labs.qt.nokia.com/2010/06/03/qt-and-open-governance/ accessed in November 2010