The Foundations of Open

∞

Evaluating aspects of openness in software projects

A collaboration between

Waugh Partners

&

OSS Watch

V2.0

4th October 2007

Table of Contents

1 Acknowledgements

This paper would not have been possible except for the support of OSS Watch, the collaboration and input of several individuals and for the initial idea from Waugh Partners of the Foundations of Openness based on the ideas developed by Pia and Jeff Waugh since 2004.

1.1 The authors

The two core authors of this paper are Pia Waugh from Waugh Partners and Randy Metcalfe from OSS Watch, Oxford.

1.2 The contributors

Many thanks to the people who contributed their comments and modifications to the paper:

- Jeff Waugh from Waugh Partners
- John Ferlito from Vquence
- Ross Gardler from OSS Watch
- Rowan Wilson from OSS Watch
- Stuart Yeates from OSS Watch

And many thanks to:

- William Weinberg from Linux Pundit who provided excellent advice and editing
- John Ferlito from Vquence who provided good input early on in the project

Pia Waugh   in collaboration with   Randy Metcalfe

Waugh Partners   OSS Watch

1.3 The license

This document is licenced under the Creative Commons Attribution-Share Alike 2.5 Australia License.

Further details on the license are here - http://creativecommons.org/licenses/by-sa/2.5/au/

2 Introduction

This project explores the notion of openness in software projects. It contextualizes different facets of openness and considers their individual and collective usefulness. It provides a tentative evaluative schema to allow others to weigh up specific criteria that may be important to them. It acknowledges that these criteria may have different weightings for different people, e.g. governance vs open standards vs code licence. The object is *not* a blanket recommendation for openness in all facets of a project. Rather it is a tool for projects, and those who use those projects, to investigate and illuminate the choices being made as well as the implication of open and closed approaches.

We don't presume that a high rating in each aspect of openness guarantees a trustworthy, sustainable, interoperable, relevant or successful project, but rather indicates an opportunity for trustworthiness, sustainability and interoperability. The relevance of a project is most strongly linked with the market need for that functionality, and the success of a project can only ever be judged by the aims of that project.

This is the first draft of this model, and upon public dissemination of this paper, we will be making a call for participation from anyone in the technology or software space to participate in the fine tuning and improvement of this model for general use. We expect this model to be broadly useful in evaluating any software project and also for projects to self-evaluate and understand the implications in the different ways they might be open or closed. We have in this paper given explanations of the basic outcomes of open and closed approaches in the areas of the project source, standards, knowledge, governance and marketplace, as we found these areas to be key factors when considering what aspects in projects impact openness.

For this paper we have rated a few projects of interest for comparison and to illustrate the practical application of the model.

This model may with some small modifications become useful in future for evaluating hardware and other technology areas of interest. However, for the time being it is focused primarily on software. It is not meant for evaluating companies.

2.1 Openness

Openness provides a mechanism for trust. But what exactly is openness? Even the characterization of openness is rather open. To move forward, therefore, it makes sense to set down some markers for openness. Openness is important in every major aspect of life, from the process of politics to the disclosure of ingredients in canned food.

As more of our lives are dependent upon new and proliferated technologies that we use to communicate, create, vote, share and much more, we must find a way to evaluate technological openness. Openness in technology impacts the sustainability, applicability, interoperability and trustworthiness in the system:

- **Sustainability** is about the longevity and access to systems and data. All of us have experienced the pain of losing access to data, or the inability to run an old software program on a new computer. With so much of what we do being created and stored digitally, sustainability is of major importance, and not just for a few years, but well into our future so we don't lose knowledge or culture. Sustainability can be improved through the use of publicly documented data *standards*, and through a healthy *market* and *community* supporting software that helps ensure easy access to the information. Also public disclosure of the software *source* means it is always available for anyone to pick up and use/support, rather than the risk of being discontinued and thus unavailable at the software end of life, as happens to most proprietary software. Access to the *knowledge* around a project, including technical documentation is also helpful in ensuring the sustainability of a project.

- **Applicability** is how broadly the software applies to different use cases and needs. Often if there is a single point of control in software, then there is a single set of aims and use cases in mind. Individual companies that create a software product can only support so many use cases, as it would not be economically feasible for one company to try to support every single possible use case for the software, so inevitably the company caters to the lowest common denominator, or optimises for a very specific use case. If however the software allowed project participation through an open *governance* model and access to project *knowledge*, then people can modify the software to meet their own needs, and thus broaden and improve the applicability of the software. A great example of this capability is in the case of language localisations. A single company may have a strong market in 5 countries that speak 7 languages, however an individual that speaks a minority language like Mongolian can't possibly provide the market opportunity to justify the company investing to support that language, so it is not available. In the case of a participatory software project, that same person can either contribute the translations themselves or pay someone external to the software project to develop the language support making the software more broadly applicable.

- **Interoperability** is how systems and data interoperate in a single environment, across multiple versions of that environment, or across multiple environments. Interoperable systems can readily communicate, share information, and coexist to perform common or related functions. Interoperability is vital to large systems such as the Internet, where millions of computers communicate using common *standards* of communication protocols, such as TCP/IP to achieve interoperability. There are times when interoperability isn't desired, such as for highly secretive systems or for competitive reasons, however interoperability is generally a useful aim.

- **Trustworthiness** in technology entails knowing that the system will perform as documented. A good example of the importance of trustworthiness in software lies in electronic voting systems. If an electronic voting system is not open at least in terms of public source code disclosure, then how can voters trust the outcome? This has already become an issue in America where a proprietary software company, Diebold, ran electronic voting booths that were alleged to be faulty[1], which cast doubt upon the election result. If the system had been developed in the open under public community scrutiny, then the general public could check that the system did what it was supposed to and could actually trust the outcomes. Trustworthiness can be improved through openly available source code, which is a *licensing* issue.

---

1   http://www.eff.org/Activism/E-voting/20030723_eff_pr.php

2.2 The Foundations of Open

Sustainability, applicability, interoperability, and trustworthiness are facets through which openness is revealed. The following are broad areas, relevant to software projects, where these facets may reveal themselves.

- **Source** – the conditions surrounding the project source code. Usually defined within the licence terms.
- **Standards** – the data, communication and other standards used within a project, for example, APIs, protocols, & documentation norms.
- **Knowledge** – the documentation, project information, decisions made, communication archives and any other content related to the project.
- **Governance** – the structure of the organisation that defines who participates in a project and the terms of participation. Includes decision making, and any practical or policy limitations on participation.
- **Marketplace** – the ability for any organisation to build a business around a project. Includes practical, legal and technological limitations to building an open marketplace around the project.

For each aspect we have identified a number of factors that we believe impact the openness of that aspect.

2.3 Closed Systems

Generally the more closed any aspect of system is, the greater the extent to which responsibility for that aspect falls upon a single entity, and thus is introduced as a single point of control. This single point of control may offer a competitive advantage, greater control over the software development process, control over who participates in the project, or control over information dissemination. As well as these competitive advantages, a single point of control also introduces some risks. Risks include a single point of failure, vendor lock-in, and reduced transparency, interoperability and broad applicability.

In terms of sustainability, corporate participation in a project generally fosters greater project sustainability through dedicated resources and support. However, if aspects of a project are limited to one company or entity, then those risks need to be assessed.

The fact that a system possesses closed aspects may be preferred or unimportant in specific cases. For example, a company may prefer that a piece of software remains closed in all aspects (apart from user knowledge) so long as they have access to the source code for bug fixing. Or with bespoke software that delivers a company competitive advantage in the market, creators/owners probably don't want that software to be open for use or participation.

3 The Openness Evaluation Model

The Openness Evaluation Model poses a series of questions based around the *Foundations of Open* defined above. Following is an explanation of each area of evaluation followed by a proposed question set. Please note that every section has an additional questions section with areas of interest or potential for draft two of the tool. The questions are also included in Appendix 1 with a tentative set of metrics for each. These have been used to create evaluation case studies. Feedback on these questions and the areas of evaluation is encouraged. Please send comments to openness@waughpartners.com.au.

3.1 Licence

Software is most often covered by copyright. Copyrighted materials are distributed under a variety of licences. Such licences set parameters and terms for how copyrighted material can and must be treated. In general licences either augment or limit basic (copy)rights granted under the applicable copyright law, that is rights to copy, modify, (re)distribute, and in some cases, the terms of actual use. These parameters or permissions for the licensee (the recipient of licensed material) make possible more or less open behaviour. A licence may also prohibit certain activities, such as the combining of materials distributed under this licence with materials distributed under a different licence or limiting how the software can be used, unless certain specific conditions are met. Similarly a licence may grant exemptions from patent infringement or impose demands for explicit public attribution of the licensor if the software is used, modified or redistributed. Clearly, openness may be enhanced in different ways by different parameters in a licence.

In the broader open development community there are two recognized authorities on licences: the Free Software Foundation (FSF), which is the maintainer of the Free Software Definition; and the Open Source Initiative (OSI), which is the maintainer of the Open Source Definition. Software that is released under an FSF acknowledged free licence is rightly called "free software". Software that is released under an OSI-certified licence is rightly called "open source software". Many licences meet both sets of conditions.

There are also many licences that *may* meet either the criteria of the Free Software Definition or the Open Source Definition, but which have not been acknowledged or certified by either authority. While these licences *may* share similar conditions to free or open source software licences, the onus for deciding whether they do is left to the licensee, the recipient of software wishing either to use it directly or combine it with other software released under other licences. This can pose legal complications taxing to legal resources of individuals or institutions.

In addition, there are also licences that are not recognized by either of the authoritative bodies and would be unlikely to meet the criteria of either the Free Software Definition or the Open Source Definition. These licences may or may not be represented by someone as either "free" or "open", or they may straightforwardly declare themselves to be proprietary licences.

Finally, the range of licences acknowledged by the FSF as free software licences or certified by the OSI as open source licences is such that there is substantial variation of the conditions for licensees from one licence to the next. These nuanced differences allow for some exploration of the openness reflected in these licences. The following questions attempt to draw out some of these differences.

3.1.1 Licence questions

1. Is the licence either of the following:

- Recognised as a free software licence by the Free Software Foundation
- Certified as an open source licence by the Open Source Initiative?
- Both
- No

Rationale: If the licence has been recognised by either of these bodies, it is more likely to have been assessed and found to be relatively open than a new licence or one which has not been OSI or FSF approved.

2. Who has permission to run the software?
   - Anyone may run the software.
   - Anyone but for some specific purpose or but some specific group (i.e. restricted inclusively) – e.g. no commercial use.
   - Only some specified group may run the software (i.e. restricted exclusively or proprietary) – e.g. free for education only.

   Rationale: If the right to run the software is limited, that limits the recipient base of the software and thus openness is limited.

3. Who is permitted to examine the human-readable source code of the software?
   - Anyone may examine the human-readable source code of the software.
   - All but some specified group may examine the human-readable source code of the software i.e. (restricted inclusively).
   - Only some specified group may examine the human-readable source code of the software (i.e. restricted exclusively or proprietary). *For example, a company creating software released under a proprietary licence where only the employees of that company have access to the source code.*

   Rationale: Access to the source code is related to the trustworthiness, the sustainability, the ability to participate in and many other aspects of software.

4. Who is permitted to adapt or modify the source code of the software?
   - All licensees may adapt or modify the source code of the software.
   - All but some specified group may adapt or modify the source code of the software (i.e. restricted inclusively).
   - None but some specified group may adapt or modify the source code of the software (i.e. restricted exclusively or proprietary).

   Rationale: The right to modify or adapt the software makes the software more open for participation and applicability to different use cases.

5. Who is permitted to redistribute the modified or unmodified source code of the software?
   - All licensees may redistribute the modified or unmodified source code of the software.
   - All but some specified group may redistribute the modified or unmodified source code of the software (i.e. restricted inclusively).
   - None but some specified group may redistribute the modified or unmodified source code of the software (i.e. restricted exclusively or proprietary).

   Rationale: If the right to redistribute changed software is limited then the benefits from being able to see and change the source are limited to personal use.

6. Does the licence permit sub-licensing of rights?

An example of sub-licensing of rights would be if code released under one licence could be redistributed in a modified or unmodified form under another licence with different rights. It doesn't include sub-licensing where rights are not modified.

- o Yes
- o Yes, but conditionally (for example, so long as certain rights are maintained)
- o No

Rationale: Sub-licensing of rights means the licensor is not bound to the rights given to them and may choose to change the rights according to their need. This may be useful in certain situations and can provide more open use of the software beyond the original licence intent.

7. Does the licence also grant a patent licence to the licensee?
An example would be where each contributor of copyright material to the licensed code that is being distributed grants to the licensee a perpetual license to use that material without infringing any patent the contributor may hold against that contribution.

- o Yes
- o No

Rationale: Patent waivers are built into some licences and can offer marginally more protection from patent litigation than no support, however the question is not weighted heavily as a patent waiver can only be given for patents for which the project has a right or licence, and there will inevitably be many thousands of patents that exist outside of the waiver offered.

8. Is the licensee required to make modified or unmodified source code available if they redistribute the code?

- o Yes
- o Sometimes – For example, the licensee may distribute an executable under another licence, however anything under this particular licence must have source code available.
- o No

Rationale: There is a difference between the right to access the source code, and the responsibility of the project to make the source code available. This question is to answer whether the latter is required.

Additional questions for consideration:

- What are the implications for integration of the source code under this licence with source code under some other licence? This question might draw out the difference between the GPL and the LGPL.
- Are there any known patents that significantly limit the project?
- The question of copyright assignment.
- In indemnification available?
- The question of implicit or explicit grants of rights, for instance in the case of patents.

3.2 Standards

A standard is an agreed upon set of formal definitions, usually technical ones, created and employed to ensure interoperability, predictability and consistency with a system or among systems. Some

standards are private to a company or product – these are closed or at best de facto standards. When a standard is published publicly and its use is not encumbered by royalties and patents, it may be termed an *open standard* presenting opportunities for the greatest levels of interoperability and access. Often enough a neutral publicly recognised standards body is also involved in defining *open standards* which can broaden market opportunities, applicability of the standard, and promote innovation on top of the standard.

Standards are vital to ICT, particularly because so many systems need to interoperate and share information across very diverse software and hardware environments. A standard in technology terms typically applies to:

- a data format (e.g. ASCII, HTML)
- communication protocol (e.g. TCP/IP, SMTP)
- the definition of a process (e.g. BizDex)
- a storage format (e.g. CD formats like ISO-9960, DVD)
- a programming language (e.g. ANSI C(X3J11), PHP)
- a development, quality or project management process (e.g. ISO-9001, PRINCE 2)

In technology, a closed -- or unpublished and private -- standard presents substantial barriers to straightforward integration into other products, technologies, etc. Beyond the actual publishing of a standard, there are other factors like the conditions of use that may limit a standard, such as access fees, royalties or patents.

A more open -- or publicly published and easy to access -- standard can be integrated into other systems, and provide the above benefits. An open, publicly published and unencumbered standard provides the most open platform for innovation, interoperability, freer markets, long term sustainability of the standard, and the broadest applicability of the standard.

The following components of a standard relate to its potential:

- Public disclosure of the particulars of standard itself
- Costs associated with acquiring or implementing the standard
- Peer review by a recognised standards or certification body

3.2.1 Standards questions

1. Is there full public disclosure of the majority of data and communication standards used in the project?
   o Yes – this means the standard's definition and how it is implemented
   o No

   Rationale: Full public disclosure of a standard is the only practical way someone can implement the standard properly in another system. It is also the only way to future-proof data and systems and ensure there is always a mechanism to replicate and access the data or systems.

2. Does the project rely on any closed proprietary standards?
   o No
   o Yes

   Rationale: Proprietary standards can limit a project's potential and interoperability. There

are of course some projects that use proprietary standards for interoperability, however this question is about whether a project could not work without the proprietary standard depended upon.

3. Are there any costs associated with any standards used?
   o No
   o Acquisition cost but not implementation cost (such as paying to download a standard)
   o Implementation cost (such as a royalty or patent fee)

   Rationale: Costs associated with either acquiring the standard documentation or in implementing the standard are a barrier to entry that limits the potential of the standard.

4. Are the majority of standards used approved and published by any of the following standards bodies – W3C, IEEE, IETF, OASIS, or ISO?
   o Yes
   o No

   Rationale: Industry, de facto and published standards such as the Microsoft doc format can be popular, however arguably less open than a standard which has gone through a process of peer review, support and publication by a trusted and international standards organisation.

5. Does the project use standardised project or development processes such as Agile or PRINCE 2?
   o Yes
   o No

   Rationale: Open project or development processes are good standards to help a project.
6. Does the project support Unicode?
   o Yes
   o No

   Rationale: Unicode support means better opportunity for multiple language support.

   Additional questions for consideration:

   ● The question of legal access to standards used – e.g. libdvdcss
   ● Is the project or project outcome certified? eg -
     http://www.linuxdevices.com/news/NS2632432515.html
   ● Are there any standards used in the project that are unique to the project?
     o *Rationale: If a standard is only implemented once, it isn't necessarily a universal standard but rather a unique instance. More than one implementation of the standard indicates it is a more open standard.*

3.3 Knowledge

Knowledge in a project may be represented in content such as documentation on a project site or comments in source code and of course is embodied in the project source code itself. It may be found in publicly archived email discussion lists, or in a project wiki or discussion forum. The extent to which the knowledge found in these different contexts may be accessed, used, contributed to, modified, or redistributed marks the level of openness of a project.

Moreover, if some or all project knowledge is constrained in some way, for example either access to it is restricted or the possibility of contributing to it is curtailed, then this too is a indication of relative openness. Of course there exist good arguments for why certain knowledge should be less open than other knowledge, even within the same project. The following questions attempt to elicit an awareness of the complex state of the openness of knowledge.

3.3.1 Knowledge

7. Which **publicly available** communication or dissemination mechanisms does the project use?
   - documentation
     - project site documentation
     - design documents or project roadmap
     - machine readable metadata (e.g. RDF)
     - wiki(s)
   - project communication
     - version control system(s)
     - email list(s)
     - online forum(s)
     - chat: IRC/IM/Jabber/etc.
     - issue tracker

   Rationale: Multiple documentation and communication components are indicative of at least the opportunity for project knowledge to exist. There are certainly cases where too many avenues of knowledge can hurt a project.

8. Does the project discourage major project communications outside the approved channels selected above?
   - Yes.
   - No.

   Rationale: If major project communications are encouraged to be done through the main channels, then the chance to lose major decision making processes and information dissemination in private conversations is limited.

9. Is any project knowledge purposely kept private?
   - None.
   - Yes, but solely due to legal or privacy requirements.
   - Yes, above and beyond legal or privacy requirements.

   Rationale: The intent to keep knowledge private is not great for knowledge openness, however there may be specific reasons to keep the knowledge private, such as is the case for legal or privacy concerns.

10. Who is able to access *all* the (non-private) project knowledge?.
   - Anyone.
   - Participants (includes contributors and users of the software)
   - Some closed subset of the participants.

   Rationale: Apart from any knowledge defined as private, the ability for anyone to acquire

project knowledge is important to their ability to participate as well as for the sustainability of the project.

Artificial limitations to access:

11. Is there any financial or legal barrier to accessing or acquiring the knowledge of the project?
    o   No
    o   Yes

    Rationale: If there is any legal or financial barrier to accessing knowledge, then that provides a barrier to entry and participation, and thus makes the project less open.

12. Is there any technological barrier to accessing or acquiring the knowledge of the project? *For example, DRM or deliberate limitation to a specific operating system.*
    o   No
    o   Yes

    Rationale: If there is any technological barrier to accessing knowledge, then that provides a barrier to entry and participation, and thus makes the project less open.

13. Is the knowledge stored in publicly published data formats (with appropriate metadata) that will make it accessible over time?
    o   Yes
    o   No

    Rationale: If the knowledge is stored in proprietary formats then it is more likely the data won't be accessible in the long term which is a risk to the long term openness of the project.

14. Is any of the project knowledge available in more than one language?
    o   Yes: Over 10 languages
    o   Yes: 5-10 languages
    o   Yes: 2-5 languages
    o   No

    Rationale: The previous questions are determining whether there are any artificial limitations to accessing project knowledge.

15. Who is able to contribute to the project knowledge?
    o   Anyone.
    o   Only project participants that register through some mechanism.
    o   Only some closed group.

    Rationale: Understanding who can contribute to the knowledge is a good way of understanding the potential for participation in the knowledge.

16. Are there public archives of the knowledge and a documented mechanism for data recovery in the case of data loss?
    o   There are publicly available archives of all material
    o   There are some or no publicly available archives and there *is* a documented mechanism for recovery
    o   There are no publicly available archives and there is no documented mechanism for recovery

Rationale: If there is a single point of loss or failure in the knowledge base, then projects put themselves at risk of massive and possible permanent interruption.

17. How good is the *user*-specific public documentation on a scale of 0 to 5 with 5 being easy to read, access, appropriate for users and generally excellent, and 0 being non-existent.

    Rationale: The usability of knowledge is a difficult thing to measure. This could probably be captured through extensive subjective questions, however asking people to rate the quality is a good start to understanding and differentiating between base standards of good documentation and projects that really excel in this area.

18. How good is the *developer*-specific public documentation on a scale of 1 to 5 with 5 being easy to read, access, appropriate for users and generally excellent?

19. Are there documentation sources external to the project? This could be external community documentation or professional publications.
    o No
    o 1-10 sources
    o Over 10 sources

    Rationale: The more external sources of documentation, the more knowledge there is available that is likely done professionally or to cater for extra use cases.

    Additional questions for consideration:

    ● Searchability – access to third party search engines/finding aids? Important for access?

3.4 Governance

The governance of a project defines the structure, succession, codes of behaviour, transparency, accountability, who can participate and the nature and roles of project participants. Some governance models choose to be relatively closed for reasons of control and/or competitive advantage. However, an open and public governance model provides opportunity for broader participation and thus applicability of the project, increased trust in the project, and the chance to follow project progress. A project with open governance also creates a healthy environment for development and growth.

Another interesting outcome of many open governance models is the ability to *fork* a project. Forking entails the ability of participants or third parties to create a new instance of the code base and the rest of the project to develop it their own way. Sometimes forking works and sometimes it doesn't. However, freedom and the opportunity to fork provides insurance against bad leadership or a stagnant community.

Many of the components listed below may seem excessive for small projects, but are nonetheless important for future-proofing, improving participation and sustainable growth.

The following components of a governance model relate to its potential:

● Roles and nature of participants

- The structure of a project
- Project governance succession
- Participation of and dissemination of project decisions and direction
- Transparency of the project
- Predictability of outcomes

3.4.1 Governance questions

1. Is there clear leadership in the project? Leadership may be an individual or group such as a board.
   - ○ Yes
   - ○ No

   Rationale: Clear leadership means a project has a better chance of clear direction and purpose, and is more likely to avoid leadership contesting and the sort of committee based decision making which can slow a project down to a grinding halt.

2. Are the structure and policies of the project clearly and publicly documented?
   - ○ Yes – includes all the following:
     - ■ Leadership structure
     - ■ Process for decision making and other project processes
     - ■ The process for becoming a contributor and maintainer
     - ■ The licence of the software
   - ○ Partially and/or only to a limited audience
   - ○ No

   Rationale: Public documentation of a project structure and policies increase transparency and trust in a project.

3. Are there publicly accessible behavioural guidelines for the project?
   - ○ Yes
   - ○ No

   Rationale: Provides a public reference by which project members are held accountable. Encourages a good working environment that is productive and welcoming to newcomers.

4. Is there publicly accessible and easy to find documentation about how to participate in the project?
   - ○ Yes for using and contributing to the software – contributing to the software is defined as any activity which adds to the project. Includes code, bug reports, documentation, and translation
   - ○ Yes, but only for using the software – use is defined as downloading and using the software as is
   - ○ No

   Rationale: The availability of such documentation encourages use and contributions to a project, so it is important to facilitate new interest in a public fashion. If a project doesn't make this available it simply makes it more difficult to get involved in any capacity.

5. Is the project leadership elected by the project community? Leadership doesn't include

advisory groups or such where they don't have decision making or voting rights. Leadership that self-selects does not count.
- o Yes
- o Partially – the other places are reserved for individuals or sponsors
- o No

Rationale: Elected leadership indicates a more openly participatory project. It is true that many projects have only one maintainer, and thus they are handicapped by this question, however it is also true that smaller project do not require as open a governance as larger projects.

6. Who is able to generally contribute to the project development? Contributing to the software is defined as any activity which adds to the project. Includes code, patches, bug reports, documentation, and translation.
   - o Anyone
   - o Participants only or some open registration mechanism
   - o Some closed subset of the participants

   Rationale: A project may choose to limit who can contribute to a project for various reasons of control, however this limits the potential of the project.

7. Who is able to become a committer? This is a person who commits code/changes to the primary project source.
   - o Anyone the current committer/s decide on via a documented process
   - o Anyone the current committer/s decide on via an informal and completely undocumented process
   - o Only the current committer/s – no documentation on how to become a committer

   Rationale: Understanding who is able to become a committer is indicative to the openness of a project to share responsibility at the code level.

8. Is there a single point of failure or control for committing changes to the primary project source? Single point of failure/control refers to both the case of a single individual <u>and</u> the case where all committers work for the one company.
   - o No – the responsibility for committing changes to the primary project source is shared and there is no single point of failure
   - o Yes – but there is a documented succession process
   - o Yes – and there is no documented succession process

   Rationale: A single point of failure/control for committing changes to the codebase provides the opportunity for massive project disruption, whether it be through an individual not having time or in all committers working for the one company and introducing the risk of hostile takeover. If there is a single point of failure, either individual or company-wise, a documented succession plan can make the difference between seamless progress of the project or a major disruption.

9. Who is able to get practical access to and use the software?
   - o Anyone – the software is publicly and freely available
   - o Anyone but a specific group (e.g. anyone but companies wanting to commercialise)
   - o Only a specific group (e.g. only people in education, or only those forced to register for access)

Rationale: The terms of use in the licence of a project does not mean that in actual practice the software is publicly and openly available for public use. the more barriers to entry for use of the software, the less open.

10. Is the software release cycle (including snapshots and major releases):
    o Consistent and predictable
    o Inconsistent or unpredictable
    o Inconsistent and unpredictable

    Rationale: A project that is predictable and consistent is more likely to encourage regular community participation and interest than a project that is inconsistent and unpredictable.

11. Is it easy to acquire, build, configure and install the source code from scratch?
    o Yes – anyone has full access to the source code and can build and install
    o Yes – but with some technical or access limitations
    o No – Source code is hard to acquire **or** hard to build and install

    Rationale: If the codebase is unable to be openly forked, then the project can be held ransom to bad leadership or hostile takeover.

12. Is there an avenue and structure for recourse beyond the project maintainer/s?
    o Yes – there is a person or body where issues can be escalated
    o No

    Rationale: If there is no avenue for recourse, the project relies on the good will of the maintainer, and thus opens up the likelihood of forking under bad leadership.

    Additional questions for consideration:

    - Question about development methodologies? Perhaps build in some metrics based on tried and trusted development methodologies?
    - Maintainers vs committers? What is the most open process of becoming each and should both be included or just committers?

3.5 Market

An open and competitive market is important to innovation, quality services and products, and ultimately to interoperability and economic growth. Opportunities to build a business around a project can be limited by technical, philosophical and other barriers, and projects can present multiple revenue/business models to project participants and/or third parties depending on the nature of the project and the technology involved. Also having more companies participating directly or indirectly in the project increase the likelihood of its being commercially viable and thus more open to building a business on.

The following components of a project relate to its market potential:

- amount of funded development
- breadth of applicability of project
- avenues for competitive differentiation
- setup costs and barriers to entry for building a business around the project

3.5.1 Market questions

1. Are there any costs or barriers to setting up a business around the project? For example trademarks, patents, royalties, etc.
   o No
   o Yes – a set once off cost
   o Yes on a per user or percentage of revenue basis

   Rationale: The higher the costs of setup, the higher the barrier to entry for creating an open market around the project. Also a set cost is far less an overhead than ongoing costs such as royalties or patents.

2. Are there any technical barriers of entry to setting up a business around the project?
   o No
   o Yes – such as DRM, proprietary hardware or other software to make it work

   Rationale: Technical barriers to entry reduce the ease of building an open market around the project.

3. Are more than 50% of the core developers from the one company, institution or department?
   o No
   o Yes

   Rationale: If so this gives one company a potential market advantage over competitors.

4. How many contributors have some or all of the time they spend on the software paid for?
   o More than 5 people
   o 1-5 people
   o None

   Rationale: The more contributors that are able to work on a project with business support, generally the more market ready it is.

5. Is the project applicable to more than one industry?
   o Yes – for example a web server, or training tool
   o No – for example a specific analysis tool that is only used in one industry and isn't applicable beyond that

   Rationale: If the software is only applicable to one industry, the market opportunities are limited.

6. How many of the following revenue models are available to a new business looking to build a revenue stream around the project?
   o More than 5
   o 1 to 5
   o None
     ▪ Customisation
     ▪ Support and maintenance
     ▪ Hosted services
     ▪ Implementation/deployment services
     ▪ Training
     ▪ Dual licensing

- Localisation/internationalisation
- Consulting
- Proprietary components

Rationale: The more revenue models that are available the better the opportunity for building a market. The more businesses that are already involved, the more the project is already succeeding in the broader market space.

7. How many organisations offer commercial software development and code customisation services on the project?
   o More than 5
   o 2 to 5
   o 1
   o None

Rationale: The more business are already offering services around the project, the more open it most likely is for a new project to come along and build a business.

Additional questions for consideration:

- The ability to create a competitive differentiator? Good or bad thing?
- Dual licensing – implies a single point of copyright ownership and thus the issue arises about a company having the opportunity to exert control over a market by being able to relicense, or limit commercial opportunities of competitors.
- Interdependence of difference applications and the market opportunities?
- Whether specific features are only available in proprietary format and what that means in creating an even playing field.

1 Appendix – Metrics

The following is a draft set of metrics for the question set. These weightings were used for the case studies of projects that follow. We anticipate that these weightings may change following input from the wider community. We also anticipate that different weightings may be applicable on different evaluative occasions.

1.1 Licence

1. Is the licence either of the following:	
o Recognised as a free software licence by the Free Software Foundation	3
o Certified as an open source licence by the Open Source Initiative?	3
o Both	3
o No	0
2. Who has permission to run the software?	
o Anyone may run the software.	3
o Anyone but for some specific purpose or but some specific group (i.e. restricted inclusively) – e.g. no commercial use.	2
o Only some specified group may run the software (i.e. restricted exclusively or proprietary) – e.g. free for education only.	1
3. Who is permitted to examine the human-readable source code of the software?	
o Anyone may examine the human-readable source code of the software.	3
o All but some specified group may examine the human-readable source code of the software i.e. (restricted inclusively).	2
o Only some specified group may examine the human-readable source code of the software (i.e. restricted exclusively or proprietary).	1
4. Who is permitted to adapt or modify the source code of the software?	
o All licensees may adapt or modify the source code of the software.	3
o All but some specified group may adapt or modify the source code of the software (i.e. restricted inclusively).	2
o None but some specified group may adapt or modify the source code of the software (i.e. restricted exclusively or proprietary).	1
5. Who is permitted to redistribute the modified or unmodified source code of the software?	
o All licensees may redistribute the modified or unmodified source code of the software.	3
o All but some specified group may redistribute the modified or unmodified source code of the software (i.e. restricted inclusively).	2
o None but some specified group may redistribute the modified or unmodified source code of the software (i.e. restricted exclusively or proprietary).	1
6. Does the licence permit sub-licensing of rights?	
o Yes	3
o Yes, but conditionally (for example, so long as certain rights are maintained)	1
o No	0
7. Does the licence also grant a patent licence to the licensee?	
o Yes	1

o No	0

8. Is the licensee required to make modified or unmodified source code available if they redistribute the code?	
o Yes	3
o Sometimes – such as the licensee may distribute an executable under another license; however anything under this particular licence must have source code available.	1
o No	0

1.2 Standards

1. Is there full public disclosure of the majority of data and communication standards used in the project?	
o Yes – this means the standard's definition and how it is implemented	3
o No	0

2. Does the project rely on any closed proprietary standards?	
o No	3
o Yes	0

3. Are there any costs associated with any standards used?	
o No	3
o Acquisition cost but not implementation cost (such as paying to download a standard)	1
o Implementation cost (such as a royalty or patent fee)	0

4. Are the majority of standards used approved and published by any of the following standards bodies – W3C, IEEE, IETF, OASIS, or ISO?	
o Yes	3
o No	0

5. Does the project use standardised project or development processes such as Agile or PRINCE 2?	
o Yes	1
o No	0

6. Does the project support Unicode?	
o Yes	2
o No	0

1.3 Knowledge

1. Which **publicly available** communication or dissemination mechanisms does the project use?	
o documentation ▪ about the project ▪ design documents or project roadmap ▪ machine readable metadata (e.g. RDF) ▪ wiki(s)	up to 2
o project communication ▪ version control system(s)	up to 2

■ email list(s) ■ online forum(s) ■ chat: IRC/IM/Jabber/etc. ■ issue tracker	

2. Does the project discourage major project communications outside the approved channels selected above?
   o Yes
   o No
	
	1
	0

3. Is any project knowledge purposely kept private?
   o None.
   o Yes, but solely due to legal or privacy requirements.
   o Yes, above and beyond legal or privacy requirements.
	
	3
	2
	1

4. Who is able to access *all* the (non-private) project knowledge?.
   o Anyone.
   o Participants (includes contributors and users of the software)
   o Some closed subset of the participants.
	
	2
	1
	0

5. Is there any financial or legal barrier to accessing or acquiring the knowledge of the project?
   o No
   o Yes
	
	1
	0

6. Is there any technological barrier to accessing or acquiring the knowledge of the project?
   o No
   o Yes
	
	1
	0

7. Is the knowledge stored in publicly published data formats (with appropriate metadata) that will make it accessible over time?
   o Yes
   o No
	
	1
	0

8. Is any of the project knowledge available in more than one language?
   o Yes: Over 10 languages
   o Yes: 5-10 languages
   o Yes: 2-5 languages
   o No
	
	3
	2
	1
	0

9. Who is able to contribute to all the public project knowledge?
   o Anyone.
   o Only project participants that register through some mechanism.
   o Only some closed group.
	
	2
	1
	0

10. Are there public archives of the knowledge and a documented mechanism for data recovery in the case of data loss?
   o There are publicly available archives of all material
   o There are some or no publicly available archives and there *is* a documented mechanism for recovery
   o There are no publicly available archives and there is no documented mechanism for recovery
	
	3
	2
	
	1

11. How good is the *user*-specific public documentation on a scale of 0 to 5 with 5 being easy to read, access, appropriate for users and generally	0-5

excellent, and 0 being non-existent.	
12. How good is the *developer*-specific public documentation on a scale of 1 to 5 with 5 being easy to read, access, appropriate for users and generally excellent?	0-5

13. Are there documentation sources external to the project? This could be external community documentation or professional publications.
   o   No
   o   1-10 sources
   o   Over 10 sources
	
	0
	1
	2

1.4 Governance

|	

1. Is there clear leadership in the project? Leadership may be an individual or group such as a board.
   o   Yes
   o   No
	
	3
	0

2. Are the structure and policies of the project clearly and publicly documented?
   o   Yes – includes all the following:
      ■ Leadership structure
      ■ Process for decision making and other project processes
      ■ The process for becoming a contributor and maintainer
      ■ The licence of the software
   o   Partially and/or only to a limited audience
   o   No
	
	2
	
	1
	0

3. Are there publicly accessible behavioural guidelines for the project?
   o   Yes
   o   No
	
	2
	0

4. Is there publicly accessible and easy to find documentation about how to participate in the project?
   o   Yes for using and contributing to the software – contributing to the software is defined as any activity which adds to the project. Includes code, bug reports, documentation, and translation
   o   Yes, but only for using the software – use is defined as downloading and using the software as is
   o   No
	
	2
	
	1
	
	0

5. Is the project leadership elected by the project community? Leadership doesn't include advisory groups or such where they don't have decision making or voting rights. Leadership that self-selects does not count.
   o   Yes
   o   Partially – the other places are reserved for individuals or sponsors
   o   No
	
	2
	1
	0

6. Who is able to generally contribute to the project development? Contributing to the software is defined as any activity which adds to the project. Includes code, patches, bug reports, documentation, and translation.
   o   Anyone
   o   Participants only or some open registration mechanism
	
	2
	1

o Some closed subset of the participants	0
7. Who is able to become a committer? This is a person who commits code/changes to the primary project source.	
o Anyone the current committer/s decide on via a documented process	3
o Anyone the current committer/s decide on via an informal and completely undocumented process	2
o There is no documentation on how to become a committer and no openness to the process	1
8. Is there a single point of failure or control for committing changes to the primary project source? Single point of failure/control refers to both the case of a single individual <u>and</u> the case where all committers work for the one company.	
o No – the responsibility for committing changes to the primary project source is shared and there is no single point of failure	2
o Yes – but there is a documented succession process	1
o Yes – and there is no documented succession process	0
9. Who is able to get practical access to and use the software?	
o Anyone – the software is publicly and freely available	
o Anyone but a specific group (e.g. anyone but companies wanting to commercialise)	2
	1
o Only a specific group (e.g. only people in education, people who have to register for access or paying customers only)	0
10. Is the software release cycle (including snapshots and major releases):	
o Consistent and predictable	2
o Inconsistent or unpredictable	1
o Inconsistent and unpredictable	0
11. Is it easy to acquire, build, configure and install the source code from scratch?	
o Yes – anyone has full access to the source code and can build and install	2
o Yes – but with some technical or access limitations	1
o No – Source code is hard to acquire **or** hard to build and install	0
12. Is there an avenue and structure for recourse beyond the project maintainer/s?	
o Yes – there is a person or body where issues can be escalated	2
o No	0

1.5 Market

1. Are there any costs or barriers to setting up a business around the project? For example trademarks, patents, royalties, etc.	
o No	3
o Yes – a set once off cost	1
o Yes on a per user or percentage of revenue basis	0
2. Are there any technical barriers of entry to setting up a business around the project?	
o No	1
o Yes – such as DRM, proprietary hardware or other software to make it	0

work	

3. Are more than 50% of the core developers from the one company, institution or department?
   ○ No
   ○ Yes
	1
	0

4. How many contributors have some or all of the time they spend on the software paid for?
   ○ More than 5 people
   ○ 1-5 people
   ○ None
	3
	2
	1

5. Is the project applicable to more than one industry?
   ○ Yes – for example a web server, or training tool
   ○ No – for example a specific analysis tool that is only used in one industry and isn't applicable beyond that
	2
	0

6. How many of the following revenue models are available to a new business looking to build a revenue stream around the project?
   ○ More than 5
   ○ 1 to 5
   ○ None
      ■ Customisation
      ■ Support and maintenance
      ■ Hosted services
      ■ Implementation/deployment services
      ■ Training
      ■ Dual licensing
      ■ Localisation/internationalisation
      ■ Consulting
      ■ Proprietary components
	2
	1
	0

7. How many organisations offer commercial software development and code customisation services on the project?
   ○ More than 5
   ○ 2 to 5
   ○ 1
   ○ None
	3
	2
	1
	0

2 Appendix – Case Studies

2.1 Comparing Kernels

As a test of the metric, we compared three well known Open Source kernels using the questions defined in Appendix 1[2].

Comparing Kernels
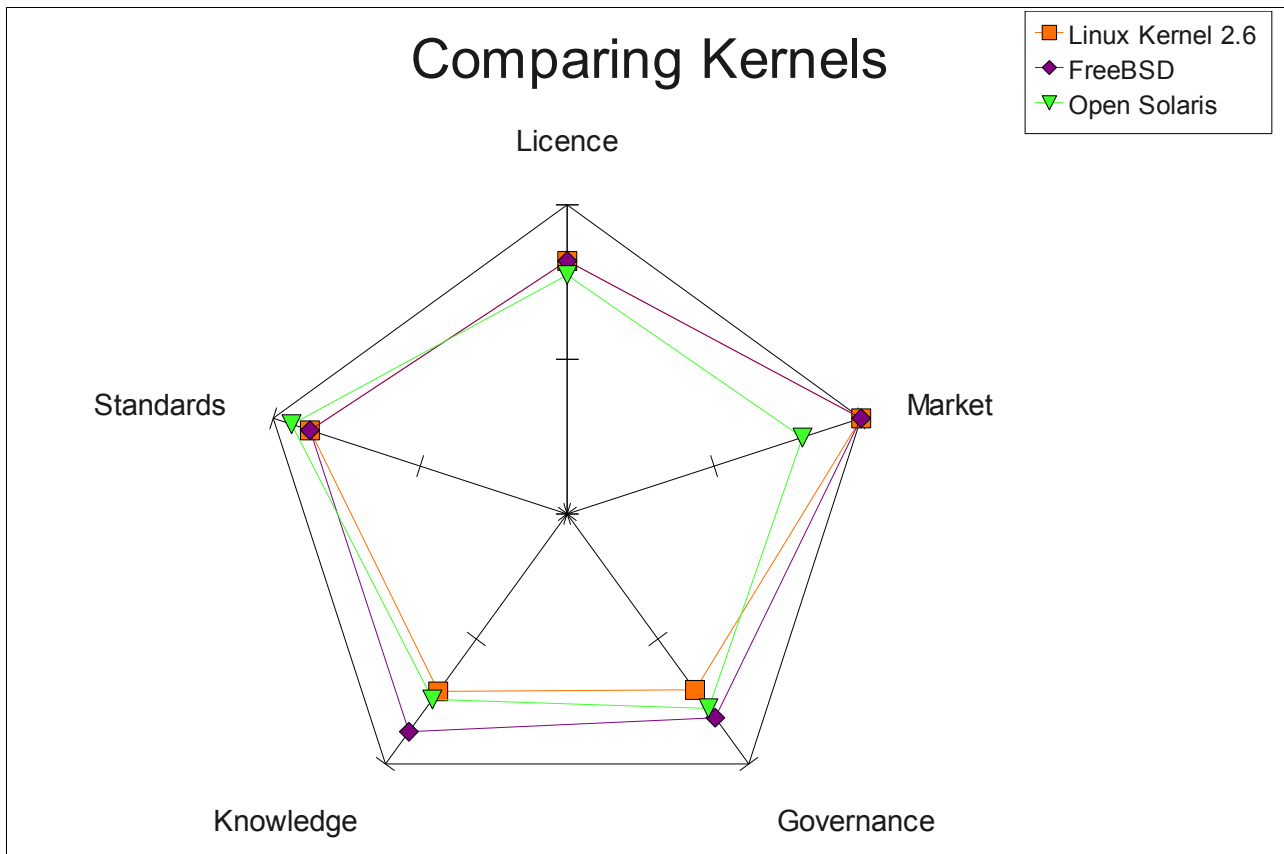
As you can see, all the Kernels examined do quite well generally. All three are very open in terms of the licences, standards and market readiness, however all three fall down just slightly on knowledge and governance.

The GPL, BSD and CDDL licences used for these three projects are very different and grant different rights. Each licence in their own way is more open than the others. The GPL does this by ensuring that modifications are made available. Although not a way of ensuring the code goes back into the project it is still better than nothing for getting positive modifications out into the open. The BSD, by allowing effective re-licensing allows a great amount of freedom on the part of the developer, and thus is open in a different way. The CDDL, by allowing relicensing (of binaries only) but makes source code be made available under the CDDL license itself. It has optional freedoms to both relicense and share code.

All three projects do well in standards and none are bound to proprietary or costly standards. Open Solaris does rate slightly higher as it clearly uses formal and documented development processes.

All three projects are fairly open in terms of knowledge, with FreeBSD doing better overall with more language support and better user documentation. Open Solaris had the best developer documentation, the Linux Kernel had the best data parity with many mirrors of the knowledge base

---

2   These comparisons are based on publicly available information. Project self-evaluation using these metrics may have access to further information that might alter the results shown here.

but had relatively difficult documentation for users or developers. Neither Open Solaris nor the Linux Kernel had the knowledge available in many languages.

All three projects are not as high rating in Governance as they could be, but for different reasons. Posted below are the actual numbers. The Linux Kernel has Governance issues because of no clear succession and the path to becoming a committer being limited (Linus is the only actual committer). Open Solaris has Governance issues due to the core committers being all within the one company with no clear process to becoming one. FreeBSD has no clear code of acceptable behaviour and the process to becoming a committer is unclear (basically you need to be recommended by an existing committer but can't directly apply through any clearly documented process).

All three did very well on market openness, with Open Solaris falling down only slightly due to the current status of the core developers all being in the one company and only one company currently offering major services around the project. As the Open Solaris community develops, this issue will disappear.

Below are the actual answers to the questions for the kernel comparison for information and comparison:

	Licence										Standards								Knowledge															Governance														Market								
Question	1	2	3	4	5	6	7	8	Totals	%	1	2	3	4	5	6	Totals	%	1	2	3	4	5	6	7	8	9	10	11	12	13	Total	%	1	2	3	4	5	6	7	8	9	10	11	12	Total	%	1	2	3	4	5	6	7	Total	%
Maximum	3	3	3	3	3	3	1	3	22		3	3	3	3	2	2	16		4	1	3	2	1	1	1	1	2	3	5	5	2	31		3	3	2	2	2	2	3	2	2	2	2	2	27		3	1	1	3	2	2	3	15	
Linux Kernel 2.6	3	3	3	3	3	0	0	3	18	0.82	3	3	3	3	0	2	14	0.88	4	1	2	2	1	1	1	0	2	3	1	2	2	22	0.71	3	2	2	2	0	2	1	1	2	1	2	0	18	0.67	3	1	1	3	2	2	3	15	1
FreeBSD	3	3	3	3	3	0	0		18	0.82	3	3	3	3	0	2	14	0.88	4	0	2	2	1	1	1	3	2	2	4	3	2	27	0.87	3	2	0	2	2	2	2	2	2	1	2	1	21	0.78	3	1	1	3	2	2	3	15	1
Open Solaris	3	3	3	3	3	1	0	1	17	0.77	3	3	3	3	1	2	15	0.94	4	1	2	2	1	1	1	0	2	2	2	4	1	23	0.74	3	2	2	2	2	2	1	0	2	1	2	1	20	0.74	3	1	0	3	2	2	1	12	0.8

2.2 Comparing Databases

Below we compare two well known Open Source databases and a well known proprietary database to show how the model applies to both proprietary and Open Source software using the questions defined in Appendix 1[3].

---

3 Again, these comparisons are based on publicly available information. Project self-evaluation using these metrics may have access to further information that might alter the results shown here.

Comparing Databases
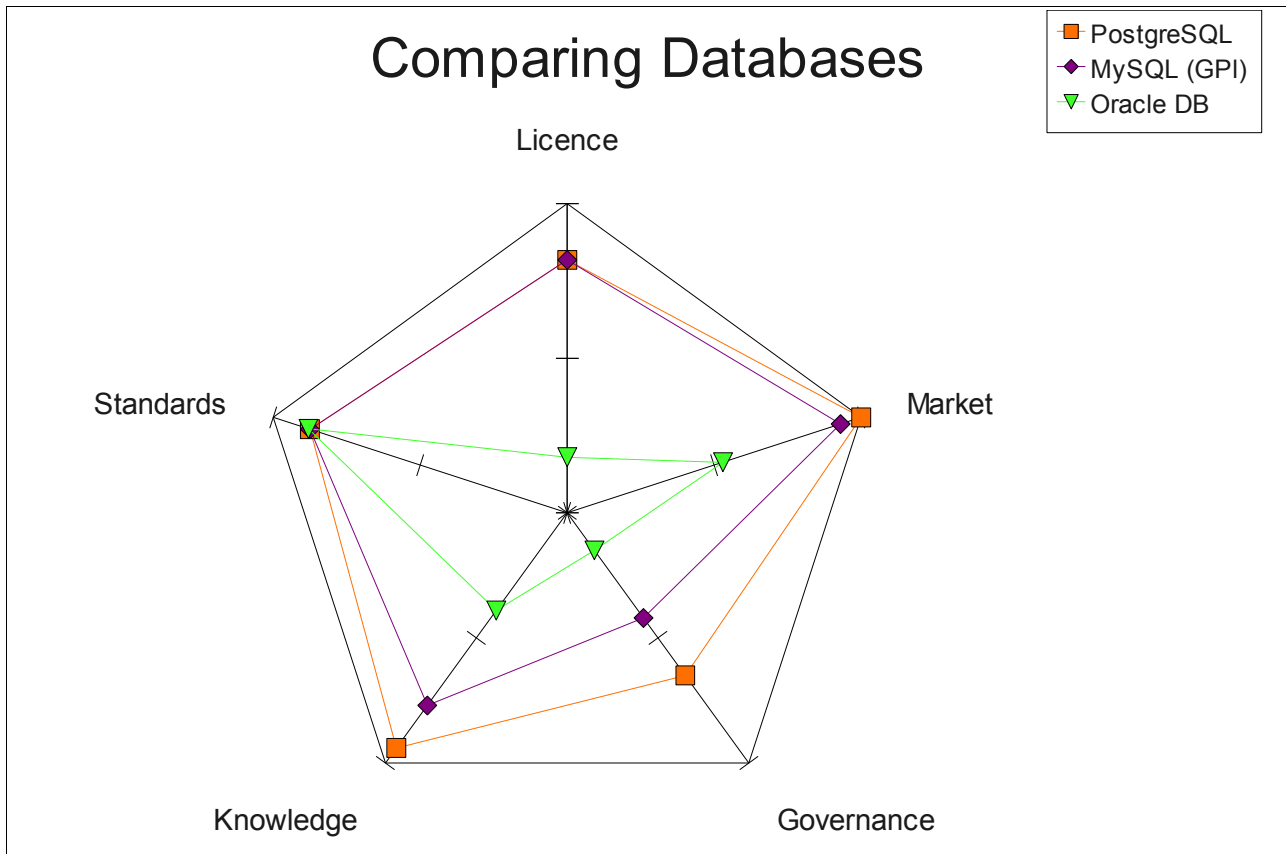
The two Open Source projects do generally better than the proprietary database, however none of the projects have particularly well documented governance models.

The GPL, BSD and a proprietary licence is used for these three projects and very different and grant different rights. The GPL does well by ensuring the modifications are made available. Although not a way of ensuring the code goes back into the project is still better than nothing for getting positive modifications out into the open. The BSD does well by allowing effective re-licensing allows a great amount of freedom on the part of the developer, and thus is open is a different way. The

proprietary licence is no open in any useful way for end user rights.

All three appeared to do well in standards.

The two Open Source projects did far better in knowledge, and PostgreSQL had better overall documentation than MySQL.

The two Open Source projects did far better in project governance than the proprietary project, however neither of the Open Source projects had particularly good governance documentation, and MySQL lost some points for having all it's committers/maintainers in the one company.

All three did relatively well in the market vector, with Oracle losing some points due to the inability for new companies to independently create business doing software modifications around their software as they don't have access or rights to the source code. MySQL and Oracle both lost some points due to projects being largely influenced by the commercial direction of those companies with more than 50% of the core developers being from those companies. This removes some commercial impetus and advantages for new companies to build a business around those projects. PostgreSQL did the best as it has the largest market opportunity according to these metrics.

Question	Licence								Totals	%	Standards						Totals	%	Knowledge													Total	%	Governance												Total	%	Market							Total	%
	1	2	3	4	5	6	7	8			1	2	3	4	5	6			1	2	3	4	5	6	7	8	9	10	11	12	13			1	2	3	4	5	6	7	8	9	10	11	12			1	2	3	4	5	6	7		
Maximum	3	3	3	3	3	3	1	3	22		3	3	3	3	2	2	16		4	1	3	2	1	1	1	1	2	3	5	5	2	31		3	2	2	2	2	2	3	2	2	2	2	2	26		3	1	1	3	2	2	3	15	
PstgreSQL	3	3	3	3	3	3	0	0	18	0.82	3	3	3	3	0	2	14	0.88	4	0	2	2	1	1	1	3	1	2	5	5	2	29	0.94	3	2	0	2	0	1	2	2	2	1	2	0	17	0.65	3	1	1	3	2	2	3	15	1
MySQL (GPL)	3	3	3	3	3	0	0	3	18	0.82	3	3	3	3	0	2	14	0.88	3	0	2	1	1	1	1	2	1	2	4	4	2	24	0.77	0	1	0	2	0	1	1	0	2	2	2	0	11	0.42	3	1	0	3	2	2	3	14	0.93
Oracle DB	0	1	1	1	1	0	0	0	4	0.18	3	3	3	3	0	2	14	0.88	2	0	1	0	1	1	1	0	0	2	2	0	2	12	0.39	0	1	0	1	0	0	1	0	0	1	0	0	4	0.15	0	1	0	3	2	1	1	8	0.53