Ministry of Justice Open Source Discussion Paper

11 December 2007 – version 1.0

Barry Polley, MoJ A&S, barry.polley@justice.govt.nz

Software development has changed. The old model of a software development firm's success - develop proprietary products, devote engineering resources to adding ever more features, hire a large sales force, and lock customers into annual support contracts to ensure long term profitability - is being turned on its head by Open Source Software (OSS). Now the goal of large market share is best met by releasing a product's source for free instead, and preparing enough technical documentation that potential customers can try it out. Revenue accrues to those firms by providing custom technical enhancements and by providing enterprise-class support. That support firm may or may not be the same firm that contributed the source initially, so customers aren't locked in, as they had been under the old proprietary software model. For example, support for the open source Tomcat application server has been offered to the Ministry by the established software firms BEA, Novell, Covalent, and Red Hat.

OSS has been dominant in IT infrastructure for years. GNU/Linux and various flavours of BSD UNIX have dominated the Internet, perl and PHP have been the predominant development languages of the WWW, and Apache actually has expanded its early 80% market share among web servers. All of those products are OSS. 2006 might reasonably be called the Year of Open Source, as OSS adoption grew widely beyond its traditional use in technology infrastructure to replace proprietary database, content management, and CRM products as well.

Government adoption of OSS is now extensive around the world. As proprietary and bespoke systems reach the ends of their useful lives, the opportunity now exists to replace them with OSS assemblies rather than launch new RFP processes for monolithic, closed solutions. The government of South Africa is the most recent adopter of OSS, in this case for all new software and (where feasible) for migration of current software [http://www.tectonic.co.za/view.php?id=1377].

The Ministry needs an explicit strategy to embrace the adoption and use of OSS. Our vendors are moving to OSS without our encouragement or consent; Java is the primary development language and also OSS, Weblogic is a major IT supplier currently merging OSS into their products, and the perimeter Check Point firewall runs an OSS operating system. We cannot choose to keep OSS out of the Ministry; our choices are to accept our vendors' decisions as they occur, or to adopt OSS for strategic benefit on terms of our choosing.

Definitions

Open Source Software (OSS) is software distributed to the public with its source code, allowing anyone to use the software, redistribute it, and change its functionality as desired. Frequently, OSS is called 'free' software, but most OSS retains copyright and exists to serve commercial purposes such as custom enhancements, training, and support. So all OSS is 'free to use', but only some OSS is 'free as in beer'. OSS contrasts with proprietary software, which is distributed as runnable binary code only. Proprietary software is 'closed' in a broader sense than lack of source code; feature changes, enhancements, and bugfixes all occur under the control of the source code holder rather than of users. Customers also are locked into relying upon the proprietary software vendor to support its products adequately.

*Open Standard*s are technical standards that are publicly visible and implementable by anyone with the requisite skills and resources. It is quite possible to have proprietary software support open standards; for example, PDF has been released as an open document standard by Adobe, but the Adobe Acrobat Reader is proprietary software. Some analysts will only use the term 'open standard' when there is at least one OSS implementation of that standard available, but that distinction is unnecessary from the Ministry's point of view.

Industry Standards are technical standards that are adopted by typical industry practice. As of 2007, most industry standard software is proprietary, even when OSS alternatives exist. This is especially true in end-user environments, with the Microsoft Office suite and with messaging systems (GroupWise, Exchange, et al.) An industry standard software package may be open (e.g., the Eclipse IDE for Java development) or closed (e.g., EndNote for research librarians' bibliographic work).

This document was initially composed in *industry standard* ASCII text file format, using the proprietary tool Notepad. It was then formatted using the *open source software* package OpenOffice 2.0.2, and made available digitally in the *open standards* Open Document Format (ODF) and Portable Document Format (PDF).

Objections

There are a number of common objections to enterprise adoption of Open Source Software (OSS). Most are invalid, or no longer valid in 2007, but others are worth acknowledging because OSS will not suit every Ministry software need. We see tremendous opportunity to deploy OSS widely within the Ministry to improve stability and application development responsiveness whilst reducing costs.

- **Objection: Unreliable (too new to trust, unstable, and/or unsupportable)**

The operating standards allowing the Internet to exist are open and simple. Further, the core software based on those standards is mostly open source - BIND, gcc, sendmail, et al. Even in 2007, every single implementation of TCP over IP in a commercial operating system derives from the BSD stack. Apache has enjoyed overwhelming and long-lived predominance as the webserver of choice on a wide variety of operating systems. Large systems integration firms like IBM and Hewlett-Packard rely on OSS to deliver solutions for their customers, and they pay thousands of skilled developers to work on OSS projects. 81% of the Fortune 500 admit to using OSS in production [CIO Insight, 2006], and it's highly likely that Global 1000 adoption rates are much higher in data centres and infrastructure projects. Initial installations of OSS infrastructure tools may or may not be approved beforehand; they're typically installed specifically because system administrators consider them functional and stable.

Trustworthiness and reliability are enhanced by providing access to source code. It is far more difficult to introduce Trojan horses or backdoors into OSS than into proprietary software. Simple coding errors such as poor range checking and more insidious errors such as buffer overflows are more easily spotted by dozens of external developers than by a few in-house ones. Performance improvements through refactoring as well as simple bug fixes turn around far faster with OSS than with even the most conscientious proprietary software vendor, where developers are kept as isolated as possible from user-reported faults. Finally, there is an improvement in supportability inherent in reading source code; developers can learn by reading exactly how a given product works and therefore how to change it. In turn, those changes can be scrutinised by other developers for security, performance, and stability improvements.

- **Objection: No commercial support**

Software companies such as RedHat have a solid market capitalisation and earnings based entirely on OSS. Other companies are funding OSS projects to assist their core businesses; for example, HP sells servers and storage solutions, and their contributions to Linux are cheaper than ongoing maintenance required for their proprietary HP-UX and related tools. (Linux isn't what they sell, but having Linux be robust enables HP to sell more hardware. HP-UX is no longer a profit centre, but a cost centre to be controlled.) Still other OSS funding efforts are competitive displacement; for example, IBM's contributions to web development and systems management tools are designed to take revenue back from Microsoft, which historically has had great success at IBM's expense in midsized firms by replacing AS/400 systems with Windows-based systems. OSS suppliers are raising venture capital; examples from 2007 include the IPO of network monitoring supplier GroundWork and EMC's spin-off of VMWare. OSS database supplier MySQL is expected to IPO in early 2008. Those companies expect to generate revenue and profits from services and support.

Support risks are actually greater with proprietary software than with OSS. Proprietary software firms may go out of business, casting their in-house knowledge to the wind when their developers find employment elsewhere. (Source code escrow offers some assurance against this outcome, but source code is of limited use when the community of developers working intimately with it ceases to exist.) Products may be abandoned because they are insufficiently profitable, requiring customers to live with risks arising from continued use of unmodifiable software, or to bear the costs of migrating to a replacement system. Finally, software firms may be acquired by competitors, resulting in comparable products being orphaned to reduce the vendor's support costs. By contrast, if OSS is available, there are always two support tactics of last resort: users can make required changes themselves, or they can pay someone else to make those changes. Frequently one of an OSS project's core developers can be hired directly to deliver custom features at minimal cost, though users are not dependent upon such arrangements.

- **Objection: Driven by ideology**

OSS is frequently portrayed as anti-capitalist, counter-culture, and the like. Early OSS projects reflected their academic origins - and in the case of the Internet, initial funding by the US Federal Government. Freedom to read source code supported collaboration in software development. The GNU General Public Licence (GPL), an early OSS licence, requires that anyone making changes to GPL code for redistribution make all changes available to the public on identical terms. (Under the GPL, if the software is not redistributed, then code changes do not have to be made public.) This no-secrets licence was at least in part a manifesto: political power through liberation of ideas and rejection of traditional copyright. And some adherents were motivated by frustration with large, powerful, and wealthy software companies - particularly Microsoft and Oracle. But actual software creators have chosen to use the GPL to drive more widespread and rapid adoption of their works, as with Linux years ago and Java at the end of 2006. Now there are software companies who deliver OSS as a fundamental part of their business model, deriving revenue and profits from ancillary services rather than the software itself. There remain users who prefer OSS for dogmatic reasons, but most adopters have more pragmatic concerns: cost, stability, flexibility, and maintainability.

- **Objection: Legal risks**

The potential legal exposure of adoption of OSS by Government has been raised as a concern [http://www.e.govt.nz/policy/open-source/open-source-legal]. OSS licences vary in structure, and only public domain code grants the user unlimited rights. A particular deployment may contain dozens of software components, and each component may have different licence terms. Some third-party components may be deployed contrary to licence terms, or may simply not be licenced for distribution at all.

Legal review of software licences is required as a matter of Government policy, whether open source or not. There are risks of legal action with proprietary software as well, should copyrights or patents be violated. For example, Alcatel/Lucent was awarded US$1.25B in February 2007 when a trial in the USA held Microsoft liable for infringing its patents on MP3 compression technology. (Had the MP3 compression algorithm been OSS, Microsoft would not have been held liable for damages.) OSS reduces the likelihood of copyright violation because the source is viewable, but it also complicates legal review because of the variety of licences in play. (For example, Linux is a GPL release, but Apache is a BSD release, and Apache plugins might have yet other licence terms.)

Hundreds of copyright and patent infringement cases have been filed by software companies against one another over OSS, primarily based on claims that protected intellectual property made its way without permission into an OSS product. None of these cases have involved users of the applications in question, so the significance of this legal risk to the Ministry is low. Competitors have also made explicit legal agreements not to sue one another, but to cooperate on OSS projects and avoid such lawsuits; even competitors Microsoft and Novell made such an agreement in late 2006 to provide support assurances to their customers, many of whom depend upon the continued success of both firms.

- **Objection: Poor user interfaces**

Historically, OSS user interfaces have been difficult to learn and use. The user community for infrastructure and development tools placed a high priority on functionality but a low priority on usability. The recognition given to contributors adding new functionality was higher than the recognition given to making the user interface smoother or easier to learn. Exceptions such as the Netscape web browser were initially proprietary, commercial products.

Currently, there is more importance placed on quality user interfaces because they offer competitive advantages over products with poor UI's. The Firefox browser achieved critical success and an 18% global adoption rate in 2006 by offering a quality user experience. (Its main competitor, Internet Explorer, costs the user nothing and requires no installation, making that 18% figure impressive.) The Ubuntu Linux distribution has achieved rapid adoption because it's easier for less technically skilled users to install, learn, and use than other distributions. The OSS graphics editor GIMP offers power and ease of use that are competitive with proprietary commercial alternatives, likewise for the graphical Java development tool Eclipse. Previously, OSS adoption was limited to technically sophisticated users who would tolerate a poor user experience, but now OSS can appeal to users who have no such patience.

- **Response to Objections**

In summary, the adoption of OSS can lead to a more stable, supportable, and cost-effective IT environment, and should be pursued for pragmatic reasons. OSS adoption is not a panacea, and should proceed or not based on a particular package's merits. Given two equivalent packages, one open and one proprietary, the OSS one would be the preferable choice for reasons of better supportability and lower lifecycle cost.

An OSS strategy

Private enterprises, especially for-profit software development companies, face incredible challenges in adapting to the widespread use of OSS. Yet for governments, the challenges are fewer, because maintaining proprietary competitive information is not within government's scope as it is with a vendor of proprietary software. Government implements software to accomplish its goals, not to gain competitive advantage over another organisation. Government seeks to externalise benefits to many stakeholders, rather than hoard them for shareholders. As a result of these differences, government agencies have a much easier case justifying the use of OSS. The Ministry of Justice is expected to support more initiatives over time, in a collaborative way with other agencies; the further expectation is that changes must be made ever more quickly and at lower cost. Adopting OSS wherever possible is consistent with those service expectations.

One major challenge faced by any IT organisation is the reduction of complexity. As demands on systems grow, and the number of legacy systems and customisations increases, progressively more effort is required just to maintain an existing level of functionality. OSS offers two forms of code reuse that reduce complexity - by reducing the amount of bespoke code required, and by deploying code that's already in production elsewhere. XEN virtualisation technology is OSS, and it reduces complexity of application configuration for higher uptime.

Knee-jerk prohibition of OSS is no longer feasible or cost-effective. Rather than pretend OSS is a passing fad, or ignore the growth of OSS use because it is a change from past Ministry practices, a rational adoption strategy would:

1) Encourage a selection process for internal adoption.
2) Identify specifically prohibited application types and licencing models.
3) Specify the rules by which in-house developers contribute to open source projects.

These three points are elaborated below, along with their most significant policy implications. (A set of Open Source Policies is provided as an Appendix to this document.)

Selection Process

Being free to choose an open source solution is not the same as allowing any developer to choose any OSS product for any purpose, which would lead to an unsupportable morass. There are thousands of OSS projects, but only a small percentage are suitable for large-scale deployment or of interest to anyone other than the original author. The Ministry should create an internal adoption checklist and process that functions in the same way an RFI does - identifying tools that might be acquired and deployed, rather than created as bespoke software from scratch. This effort should consist of two sequential processes: an initial screening (to eliminate unsuitable options) and a subsequent formal decision matrix (to rank suitable options). Even high-end, special-purpose tools are being made available as OSS - for example, very expensive yet capable commercial data warehouse ETL tools such as Informatica and DataStage now have OSS competition in JasperSoft and Talend. An analogous situation exists with the world's most-installed document management system, Documentum, and its new OSS rival, Alfresco.

An internal checklist for selecting an OSS product should include the following:

1) If there is a proprietary competitor, does the OSS alternative do at least one thing better?
2) Is there an active development community? (Not necessarily a large one)
3) Are there developers who are paid to contribute?
4) Is there a documentation effort that's kept current?
5) Are the licence terms clear and compatible with other licences?
6) Can support be obtained? (Must be available, affordable, with QoE/QoS specified)
7) Is the adoption rate stable or growing?

The goal is to pick a package that will have a long useful life, ideally beyond the Ministry's need for it. While OSS is inherently more maintainable than proprietary software, the benefits of OSS are compromised dramatically if additional maintenance must be performed in-house when the developer community vanishes. The migration from a stagnant OSS package would still be less complex than migrations required by end-of-life proprietary software such as Tuxedo and Powerbuilder, however, some effort should be expended to ensure such a migration is not required prematurely.

The Ministry, along with the entire NZ government, will need to revise its procurement practices to reflect the reality of OSS. Proprietary software vendors are able to justify their substantial expenditures on responding to an RFP because they have assumed those costs into their licence fee structure. OSS vendors will not have the ability to do that, as there are no upfront licence fees to offset their RFP expenses. The Ministry may well need to assign its own internal resources to identify relevant OSS packages and provide technical evaluation in lieu of an RFP response process.

Vendor relationships will need to change as well. Large consulting firms' traditional bespoke waterfall model of development will not be able to compete in any important sense (functionality, cost, or time-to-market) with OSS adoption. However, some firms may be able to add value by providing custom development to extend OSS, managing its deployment, or providing training – efforts that are smaller in scope than sought from a full-scale RFP, but very important to a project's success. Some local firms have already embraced OSS as part of their service offerings, such as Catalyst IT through its Open4Business portal [http://www.o4b.co.nz/] and consulting services. For custom changes to OSS, it's not necessary that the firm be local; broadly adopted OSS projects generate small, boutique consultancies around the globe, whose continued survival depends on the quality of their work. In some cases, core developers of a given OSS product may be available to perform custom development, and if that custom work is of general use it can be released to the community for review and ongoing maintenance.

Selection of an OSS product and associated support services should be done with care, but in the event of problems, the very nature of OSS offers flexibility. If a vendor is not providing adequate support, then keep the product but engage another vendor to support it. If an OSS product is not working as expected, or if the demands placed upon it change too dramatically, then migrate the data and adopt a different OSS product. (OSS products typically store data in more accessible formats than their proprietary counterparts.)

Application Types

Initial efforts are most likely to succeed with back-office projects whose users are technically savvy. File distribution, workflow, data mining, and document management applications used to be the province of complicated and very expensive proprietary solutions. Each of those areas now encompasses multiple, active OSS projects that can be adopted in combinations that match user needs better than any single, monolithic solution. Development toolkits such as Spring and JBoss SEAM are OSS, and enable service-oriented development and deployment consistent with the Ministry's Enterprise Architecture. There are OSS functional equivalents to major desktop applications, so future migrations should be considered. For example, the Open Document Format (ODF) has attracted widespread commerical support, making migrations from Microsoft Word to an open equivalent feasible to consider.

But there are risks of disruption that accompany such changes; for example, though the GIMP image editor is arguably functionally identical to the proprietary-and-costly Adobe Photoshop, the latter has decades of user experience on its side, and skilled users are unwilling to retrain on a new product when the existing one serves their needs so well. Similarly, the OpenOffice OSS product allows the dedicated user to edit and exchange Microsoft Office files, without paying for MS-Office, but retraining of skilled MS-Office users is necessary, and the compatibility on-screen is not yet perfect. The economics of retraining and possibly cleaning up millions of historical documents do not yet justify a move to OSS in this area. (The economic factors may change as OpenOffice improves, and as Vista's onerous hardware requirements cause departments to consider a sideways migration to Linux. In New Zealand, the IRD has already announced its consideration of moving to Linux + OpenOffice on expense grounds – not in confidence, but in public.)

Consideration should also be given to the types of licences associated with a given OSS project. The Ministry cannot feasibly prohibit the use of GPLed software (such as Java) but it can prohibit changes to any such software, so as to remain in compliance with the GPL. That policy would reduce the risks associated with custom changes, such as being orphaned from main source code ('trunk') releases, but at the very high cost of reducing the flexibility of function that OSS offers. Historically, Crown copyright has been reserved for any code written by the Ministry or its vendors; however, current SSC policy defaults to provider ownership of code with the Crown being granted a state-services-wide licence for use. BSD-style licences are thus preferred, encouraging economic development through improved commercialisation opportunities while meeting the Ministry's application needs.

In-House Development

The Ministry has stated its commitment to be an 'employer of choice' for skilled technical professionals. Adopting OSS in development and deployment is consistent with such a commitment. (Put bluntly, technical professionals want to keep their skills sharp and current; in 2007, working with open source tools and contributing to OSS projects are the way that's done.) There are very talented developers who work with older proprietary tools such as Visual Basic, ABAP, PL/SQL, and the like, but their numbers are dwindling in the current NZ seller's market for IT labour. Simple policies need to be established so new workers know what OSS tools they can use, and the extent to which they can contribute to OSS projects. (Many technical companies have followed Google's lead in allowing staff to contribute to OSS projects as part of their job responsibilities. It's unclear whether the taxpayer would approve of such policies here, and in any event the desire to give back to a code community can conflict with job performance expectations.) Contributions to a project's trunk release would reduce risk and build respect for the Ministry as a community member.

Internal development approaches change as a result of OSS adoption. It is not necessary to adopt a single OSS product when a combination of OSS products provides better coverage with less bespoke code. We can have best-of-breed functionality without dependence on a single OSS project, which is a change of paradigm for most developers used to tailoring a single body of code to improve its utility. We can also acquire skills from small, local development shops as project needs change, without having to worry about intellectual property rights as a commercial entity might.

Risks

There are a number of risks and obstacles worth noting when adopting OSS:

Software maintenance burden may increase. A thoughtful review of particular packages' codebase and community can mitigate this risk, as can adopting only OSS products with dedicated vendor support. The mindset on the part of internal technical staff needs to change too - maintenance is not necessarily limited to filing a bug report with one vendor, but entails more active investigation and problem resolution. The likelihood and speed of problem resolution will still be much improved because the old dependency on a single vendor's in-house engineering resources will no longer be a constraint.

It's not always obvious how to identify and obtain OSS. Of course, it's not always obvious how to identify proprietary software alternatives easily either, but at least the RFI process and extensive resources devoted to marketing by proprietary software vendors help. Google and Sourceforge are wonderful tools, but it's easy to overlook a good candidate project whilst wading through dozens of unsuitable ones.

Some OSS projects started as hacks and are architecturally unsound. Without some time investment, it's hard to gauge the soundness of a given project. Size of the core development team is not a reliable indicator, unfortunately.

OSS projects can lose momentum/developer interest, or hit architectural brick walls (primarily from lack of an actual design in early versions). Loosely-coupled systems might include services that have poor uptimes or simply disappear, affecting the whole system.

Industry standards may not be open (e.g., Microsoft Word file formats) so there are reverse-engineering risks with any OSS dependent upon those standards. The MoJ preference for open standards (e.g., Open Document Format, which is now supported even by Microsoft) lessens this risk.

Summary of Benefits

The benefits of OSS can be summarised as follows:

1) Cost savings (when considering full costing, not just acquisition costs)
2) Improved QA - no need to pay for suitable test instances, and functionality decisions can be deferred until late in the development process without need to consider licence consequences
3) Avoiding vendor lock-in (purchase, customisation, ancillary products, orphaning) - SOA is an essential part of this freedom, OSS is another
4) Quicker fixes, not implemented on a vendor's schedule
5) Main branch backporting (integration as opposed to building for custom features) is feasible, therefore there is less risk of being orphaned with a custom version that can't be supported properly
6) Agile deployment changes do not need renegotiation with vendors and allow new demands to be met more quickly (e.g., creation of a new tribunal)
7) Decreasing the risk of being stranded by proprietary systems (via company dissolution or product discontinuation)
8) No more 'security through obscurity' - rather, security through community review
9) Breakdown of N-1 practice (code ships when ready rather than when patchsets are issued)
10) Code reuse - both reduction of Ministry bespoke code and leverage of existing OSS code

OSS was once an extraordinary way of thinking, limited to academia and small guerilla projects in a community of hackers. Increasingly, however, it's the norm.

Appendix: Open Source Policies for MoJ

Policy 1: Open standards
Choose software that implements open standards wherever possible. OSS is more likely to implement open standards than proprietary software, facilitating interoperability and data transfer in and out of application data stores. But evaluation must be on a case-by-case basis, as some proprietary software has excellent support of open standards and might meet Ministry needs better than a comparable OSS product.

Policy 2: Prefer OSS
When evaluating software as part of a technical solution, preference is to be given to OSS alternatives over comparable proprietary solutions. OSS increases deployment flexibility, provides for more robust code, allows faster changes, and permits flexibility of support arrangements. Put generally, OSS avoids the risks associated with vendor lock-in, and greatly lowers the cost of adoption (though not to zero; see below). This policy does NOT override the Ministry's policy of reuse before buy/build; existing applications should not be eliminated simply because they do not include OSS components. (Ignoring OSS alternatives is a form of irrationality, but replacing working systems because they aren't built with OSS is equally irrational.)

Policy 3: Review licences
When considering an OSS product, the licence terms should be identified and reviewed explicitly. Licence agreements with proprietary software vendors and all service contracts are already subject to review and approval. This burden is much less with OSS (as initial download and evaluation incur no payment to anyone) but we need to collect and manage specific licence requirements for every OSS product before production deployment so any licencing constraints on use and distribution are known upfront. A simple way to comply with this policy and reduce risks associated with licence terms is to require that MoJ adopt only OSS using one of the widely-trusted licences: http://www.opensource.org/licenses/alphabetical

Policy 4: Formal OSS evaluation
When evaluating an OSS product, commit resources to formal evaluation. Because OSS costs nothing to evaluate and adopt, there is no funding stream available to pay for sales engineers' demos, independent laboratory evaluations, or generating RFP responses. Therefore adoption of OSS incurs a small upfront cost, not in licence fees but in time commitments by technical staff to deciding amongt OSS possibilities based on technical merits. This cost must be acknowledged and managed. Evaluation should occur in two phases: a short list of non-negotiatible criteria can reduce the number of options to consider, then a following formal decision process can rank the remainder. The formal process can be a simple comparison matrix based on half a dozen critical requirements; it does not need to include every relevant criterion. If no OSS offering is found to be adequate, traditional proprietary software procurement may be called for, using the best OSS solution as a benchmark.

Policy 5: Version
Adopt most current versions of any OSS product, and adopt a release lesser than 1.0 only if substantial evidence of stability exists. OSS releases tend to be frequent, driven by perceived needs in the user community rather than an arbitrary release calendar. Beta periods are typically long and widely followed, so current versions are the most stable and feature-complete; untested or untrustworthy code usually appears in its own 'edge' version, which is easily avoided. The typical N-1 policy (i.e., don't roll out a major release of software until the subsequent major release happens) is an artifact of proprietary software in a feature-competitive marketplace that placed little value on stability, and does not apply to OSS.

Policy 6: Active development
Reject adoption of any OSS product that does not have clear evidence of BOTH current development work AND community support. One substantial advantage of OSS is avoiding lock-in by vendors and possible orphaning, but this advantage is greatly diminished when customers are locked in to OSS that is essentially dead, without a community to enhance it and keep it compatible with changes in operating systems, security, etc.

Policy 7: Commercial support
Adopt OSS products that have commercial support options available. Excellent support may be available from local firms or from specialised firms operating across international borders. Consulting firms such as EDS, KPMG, et al. may bring implementation expertise and hands-on training as needed. Growing internal expertise in the support and enhancement of every OSS product adopted by the Ministry is not likely to be feasible, and would be less important that improving knowledge of the business it supports.

Policy 8: Enterprise architecture
Use the EA to guide the process of selecting an OSS product. The EA does constrain the solutions that are considered for strategic applications, and guides tactical solutions as well. OSS products should be evaluated for EA conformance exactly the same as COTS or bespoke solutions are. Product selection should give preference to technologies appearing in the Business Applications platform for support and deployment reasons (e.g., POJO under Tomcat as opposed to pure perl technologies), as such technologies will be easier to support in-house today and moving forward. OSS that is less compliant with the EA can still be adopted, but only if the solution offers compelling advantages over alternatives, and if it can be maintained as a black-box project by an external vendor.

Policy 9: Release Code Changes (Integrate, Don't Build)
Allow both internal and third-party developers to implement and release code changes back to the community. Developers are paid to deliver functionality of use to the Ministry, not to provide random enhancements for the benefit of the community at large. However, there is no reason to keep code changes secret, as the code is not confidential material. (Data, of course, need to be secured in accord with good operational practices, privacy laws, etc. and cannot be disclosed.) Going beyond simple use of OSS to contributing code back to the project decreases the risk of changes being isolated from the trunk release. It also completely bypasses possible issues with GPL compliance. And put bluntly: it's easier to ask for and receive help FROM the community when you also donate back TO the community.

Policy 10: Documentation
Adopt only OSS that have an ongoing technical documentation effort associated with them. Many of the OSS benefits outlined in this paper accrue from simply having access to source, but if that source is incomprehensible, those benefits are much reduced. Poor or out-of-date technical documentation makes fixes and enhancements difficult. Very few OSS community members are inspired by writing excellent technical documentation, so its presence is also a sign of a vibrant developer community. User documentation is less critical, especially for infrastructure products; technologists are quite used to inadequate documentation from their experiences with proprietary software (where documentation is typically an afterthought and does not generate revenue).

Open issues not addressed by these Policies
Issues not addressed explicitly above but critically important to MoJ success are:
(1) Improvements in software and solutions development practices,
(2) Operations impacts such as release processes and changes to the Gen-i relationship,
(3) Internal versioning and configuration management,
(4) R&D requirements and demands on the A&S lab,
(5) Government-wide impact assessment of Novell/Microsoft IP and reseller agreements,
(6) Broader IP issues within NZ government related to open source software, whiteboarding, digital rights management, and copyright; and
(7) Broader issues within NZ government related to economic development. More widespread adoption of OSS is a form of 'Buying Kiwi' by means of encouraging growth of NZ-based software and service providers.

Disclaimer
This document reflects months of discussions within MoJ and with its partners. It has been revised to reflect feedback from dozens of readers, and is believed to be consistent with existing MoJ policies. But this document is not the expression of any sort of legal or commercial commitment by MoJ.