RFI response from Russell McOrmond

Russell McOrmond
305 Southcrest Private,
Ottawa, ON
K1V 2B7
Phone: (613) 733-5836
Cell: (613) 262-1237
e-Mail: russell@flora.ca
Website: http://www.flora.ca

February 18, 2009

Solicitation Number 24062-080278/A

1. Details of respondent

I wear a number of different hats that are relevant to this RFI

- I have been a member of the Free Software community since early 1992, and a user and later support person for Linux since late 1993.
- I run a sole proprietorship business offering support for Free/Libre and Open Source Software (FLOSS), primarily as a system administrator and software author.
- I am the private-sector co-founder of the GOSLING (Getting Open Source Logic INto Government) community[1], founded in 2002.
- I have been the volunteer policy coordinator for CLUE: Canada's association for FLOSS, since early 2006

While I am a supplier of software support services, including to the Government of Canada, I am not a supplier of software the way that acquisitions branch has thus far treats software. In fact, it has not been obvious how I can offer these services through the MERX acquisition process. It is because of this that I am responding more as an informed citizen of Canada, rather than the intended audience of software suppliers.

I am currently on contract with Agriculture Canada, helping with IT services for their geospacial web and internal services within their National Land and Water Information Service (NLWIS). I was hired due to my Linux and Open Source experience. I will be including a section near the end of this submission on some of the lessons that can be learned from my experience at Agriculture Canada thus far.

2. Introduction and overview

Before answering the specific series of questions and providing feedback to appendix A and B, it is useful to put this submission into context to avoid excessive duplication of ideas within the response.

2.1. Language clarification

For the purpose of this submission I will use **Free/Libre and Open Source Software (FLOSS)** in place of either Free Software, Open Source Software, or Free and Open Source Software (FOSS). The first known usage of the term FLOSS was as part of the Free/Libre and Open Source Software: Survey and Study[2] commissioned by the European Commission which was published in 2002. Free/libre and open source software (FLOSS) is just a combination of the Free Software and Open Source Software, emphasizing the 'libre' meaning of the word free, to avoid a common confusion in the English

1 Getting Open Source Logic INto Governments (GOSLING) http://www.goslingcommunity.org (Retrieved February 2009)

2 Free/Libre and Open Source Software: Survey and Study (2002, November) , retrieved February 2009 at http://www.infonomics.nl/FLOSS/

language.

Many common misconceptions about the term Free Software comes from confusion with the meaning of the word free. While nobody would hear the terms free market or free trade and believe we are talking about something that is non-commercial or possibly anti-business (or even anti-capitalist), misinterpretations or sometimes deliberate misdirections are made about Free Software.

There is no universally recognized and agreed upon way to reference non-FLOSS software. The terms "proprietary" and "commercial" are inappropriate given all software still under copyright has one or more proprietors, and both FLOSS and non-FLOSS can be commercial in nature. For the purposes of this submission I will use the term **sole proprietor software**, focusing on software where nearly all ownership rights remain with exclusive rights holder (copyright, etc).

2.2. Meeting RFI objectives

This RFI process commenced through MERX can only be seen as a small portion of the required consultation and learning process that will ultimately meet the stated objective of creating policy. Suppliers of software related products and post-acquisition services which are the normal target audience of MERX can only provide a small part of the picture. This is true whether we are talking about software which is no charge and software where a royalty fee is charge.

With sole proprietor software you have a single proprietor which is either the exclusive rights holder (copyright, patent, etc) themselves, or who have acquired these rights from a third party in a way that allows them to resell. In this model it is most often the case that only these proprietors are able to offer key forms of software support, such as any form of support which requires modification of the software, or access to the source code.

With FLOSS, the licenses grant rights to all entities in all sectors to provide this support. This means that not only is there a free/libre market for private sector/commercial support, but that the government has the option to harness its existing workforce towards solving problems. The RFI makes a presumption that the government is a consumer as compared to a developer or distributor, which is an inappropriate assumption and restriction given both the realities of government operations, current practices within the government and other similar organizations, and the skill levels of existing government employees and contractors.

For the Government of Canada to best harness all the software options available to it, and to be able to choose the best options, it must consider not only acquisition policy but the interdependency with human resources and other policy. The federal government may also achieve goals that use software through collaboration with other levels of government, governments in other jurisdictions, or with entities in all other sectors of our economy (private, education, volunteer, health, etc). Consultation processes with these other sectors, as well as a larger portion of the private sector, should accompany this RFI process.

In my roll as co-coordinator of GOSLING I have met with many government employees and consultants working within the government. Many have an advanced understanding of the full spectrum of business methods and licensing models used within the software ecosystem. There is already considerable usage of FLOSS within the Canadian federal government, including full participation in software projects by members of the public sector (both projects originated by government branches, and projects originating elsewhere). Acquisitions branch of PWGSC would be well served to consult with fellow public sector people who may be able to bring the understanding of PWGSC further than this specific RFI will be able to.

I also want to ensure that the branch takes existing practices into consideration when working on this policy. The new policy should not become a roadblock to the appropriate use of software to make government more productive. Many common no charge licensed software can be pre-evaluated, with that evaluation being able to be adequately documented and published in a way that documents the pre-approval and the authentic source of the software.

2.3. Total Cost of Non-Ownership (TCNO)

Many studies seek to make use of a Total Cost of Ownership (TCO) for software. Given the end-user of software is not the owner of the software, but a licensee, this evaluation style is inappropriate for software. A more appropriate evaluation would be the Total Cost of Non-Ownership (TCNO)[3], which evaluates the costs (and possibly benefits) of not being the owner of the software.

An obvious example is in the area of post-acquisition support where for FLOSS there is a free market for support, while you are ultimately tied to a single supplier (and their resellers) in the case of sole proprietor software. Anyone with a basic knowledge of markets will recognize the benefits to the software user (in this case, the government) of a free market over a monopoly supplier scenario.

2.4. Special nature of Software

Software has many traits that make it inappropriate to use the same evaluation criteria as used for the acquisition of tangible goods.

2.4.1. Code is Law

This concept was best described in a book by Lawrence Lessig titled Code: and other laws of Cyberspace[4]. The idea is fairly simple in that it compares how software code, the rules which computers obey, with legal code, the rules which humans in society obey. For understanding many of the most critical questions in computer software, the best way is to look at software with a law or political science lens.

An important question surrounds the automation of government policy into software. When we translate public policy from French to English, both versions are available to the public to ensure that the translation was done correctly and we don't have problems with different citizens seeking to obey different laws. When we translate public policy from English or French to a computer language, the

3 I first read the phrase "Total Cost of non-ownership" within a paper by Brendan Scott titled "Why Free Software's Long Run TCO must be lower", 15 July 2002 http://members.optushome.com.au/brendanscott/papers/freesoftwaretco150702.html (Accessed February 2009)

4 Lessig, Lawrence (Edition 1 published in 1999 , edition 2 in 2006). **ISBN-13:** 978-0465039142 . More details at http://codev2.cc/

same required level of transparency and accountability is not yet offered.   This is a problem that must be fixed, with software that automates government policy needing to be just as transparent and accountable as the public policy it is automating. While proactive disclosure is ideal through public licensing, this software should as a minimum be available under Access to Information.

The current practice is to treat software as if it were legitimate to claim vendor proprietary rights as an excuse for the software to be non-transparent. We should be reversing this trend, requiring that a very strong justification be made in what must become the extremely rare case where the policy is not available under ATIP.

2.4.2. Software is non-rivalrous

> "If nature has made any one thing less susceptible than all others of exclusive property, it is the action of the thinking power called an idea, which an individual may exclusively possess as long as he keeps it to himself; but the moment it is divulged, it forces itself into the possession of every one, and the receiver cannot dispossess himself of it. Its peculiar character, too, is that no one possesses the less, because every other possesses the whole of it. He who receives an idea from me, receives instruction himself without lessening mine; as he who lights his taper at mine, receives light without darkening me." Thomas Jefferson[5]

Software, as a form of pure knowledge, has traits that make it unlike any tangible object that may be acquired. The chief among these are that it is non-rivalrous, and that from an economic analysis that it has a zero marginal cost to the producer. While it may cost the developer time to come up with the initial knowledge, there is no additional cost to them whether one person or the entire population of the planet have access to that knowledge.

This zero marginal cost is important to your analysis given whether software is "no charge" or a royalty fee is charged is a relatively insignificant business model choice. It is the other terms under which the software is offered that are important for the government to evaluate. While there may be vendors who for purely ideological reasons would be unwilling to make minor tweaks to their marketing to bundle "No Charge Licensed Software with their support contracts, these companies will turn out to be a rare exception.

---

5 Jefferson, Thomas. The Founders' Constitution, Volume 3, Article 1, Section 8, Clause 8, Document 12
http://press-pubs.uchicago.edu/founders/documents/a1_8_8s12.html
The University of Chicago Press

2.4.3. Software production can be organized beyond markets and firms

Following on from the non-rivalrous nature of software, there is a growing understanding of a wider spectrum of methods of production and distribution that is possible than for tangible goods. One of the best sources for understanding the additional possibilities are papers[6] and books[7] by Harvard Law School professor Yochai Benkler.

3. Answers to specific questions

The following are my answers to the specific questions asked within the RFI.

3.1.1. Q1. In the Overview, the Crown provided a definition for No Charge Licensed Software. Is this an appropriate definition?

I feel the definition is misleading. By highlighting two specific examples such as FLOSS and freeware, people giving a quick look at the definition may believe that this primarily focuses on FLOSS. Since software has a marginal cost to the producer of zero due to its intangible and non-rivalrous nature, it is a marketing strategy for the producers to offer any and all software under terms that would qualify under this definition.

As an example, vendors such as Microsoft could change the marketing language of existing contracts to instead of suggesting that these are payments for licensing to instead indicate that the payment is entirely for support, and that a "No Charge License" is available to any organization which has an appropriate support contract. There is nothing in the definition which specifies if any criteria or limitations exist for those who this "No Charge Licensed Software" is available to.

With this in mind, this definition realistically only separates appropriately licensed (legal) uses of software from unlicensed (infringing) uses of software. This is an important distinction to make, given the government should be doing everything it can to ensure that it is not making use of unlicensed (infringing) software. The policy should be clear, however, of what constitutes infringing software and what are the full spectrum of methods to obtain properly licensed software. Infringing software is available for sale, and the key policy will be ensuring the verification of the source and authenticity of the software (whether the charge for the software is $0 or otherwise).

6 Yochai Benkler, Coase's Penguin, or Linux and the Nature of the Firm, 112 Yale L.J (2002). Online at http://www.benkler.org/CoasesPenguin.html (Retrieved February 2009)

7 The Wealth of Networks: How Social Production Transforms Markets and Freedom (Yale University Press 2006) available in print as well as online at http://cyber.law.harvard.edu/wealth_of_networks/ (Retrieved February 2009)

3.1.2. Q2. What are reasonable criteria that the Crown should consider in a decision process for acquiring No Charge Licensed Software? Are there circumstances in which the acquisition of No Charge Licensed Software would not be advisable?

The sample list of criteria that appears in Appendix A appear to assume a process focused on a single vendor of software, asking whether the supplier of the software itself provides various services such as post-acquisition support or legal indemnification. As I will discuss in my comments to this appendix, this attempt to force the bundling of these services with the originator of the software is inappropriate.

There are many circumstances in which the acquisition of specific software would not be advisable, but whether the software was "No Charge" or not is not itself a relevant criteria. For instance, the Crown should be verifying the source and authenticity of the software to not only ensure that the software is not infringing, but also for security and other concerns given the author (or an unauthorized intermediary) could have introduced malware into the software.

Whatever the process, it should not become an excessively complex barrier to government productivity. Quite a bit of

3.1.3. Q3. What factors other than price should be considered as part of an evaluation guideline for No Charge Licensed Software? Are there other factors beyond those outlined in Appendix A & B that the Crown should consider?

For all licensed software the government should be evaluating both the conditions of the availability of the software (IE: is it tied to some other product or service, such as a support contract or other vendor tie-in), as well as the TCNO.

A key component of the TCNO is evaluating first whether there is a free/libre market for post-acquisition support, as well as evaluating the size and availability to the Crown of contractors and/or human resources within that marketplace.

In the case of No Charge Licensed software the government could do this evaluation once and then make a list of pre-approved software (and their authentic source) which could be acquired without additional paperwork. In the case of charged licensed software, existing standing offers are often already in place.

3.1.4. Q4. How should existing Government Furnished Equipment, Services, Service Level Agreements and internal resources be considered when evaluating the usage of No Charge Licensed Software?

It is not clear what this question is asking, and seems to mix together a number of different things, few of which are unique to "No Charge" licensed software.

3.1.4.1. Software that may be bundled with hardware

It is already common practice for the government to take hardware, wipe any software that is pre-installed, and install a site licensed version. This has caused the government to double-up existing licenses, and for licenses that cost money this is a large waste of taxpayers money. The government should be demanding hardware suppliers unbundled any software and provide bare hardware.

3.1.4.2. Availability of service level agreements, or existing agreements

It is my understanding that the government does not make full use of existing SLA's, not holding software vendors who have signed such agreements to minimum levels of service.

I remember famously in May of 2002 when the entire of the PWGSC mail system was unavailable prior-to and for the duration of our Open Source Solutions Showcase. We were told that there was an SLA with Microsoft for minimum downtime of the Microsoft Exchange service, and that the government did not pursue any action when Microsoft failed to meet the agreement.

Without the ultimate threat of being able to go to a competitor for support, and the apparent unwillingness of the government to take sole proprietor vendors to court, I am not certain of the value to the crown of these agreements.

3.1.4.3. Internal Human Resources and contractors

As already mentioned, the choice of what software to use is as much a human resources as an acquisition policy question. For many software outcomes it may be far more appropriate to have existing human resources address the need rather than going through an acquisition process. Existing public sector personnel are often able to make use of existing software already licensed to the government, make modifications to the software where modification is allowed, and more accurately and more cost effectively fill the requirement than any external supplier.

Existing departmental helpdesks should also be considered, but only in the case where such helpdesk support is needed by the user. In many cases NCLS will be used by a public sector employee who does not need any support. In this case the employee should be advised that the helpdesk can not offer support, and this lack of support should not affect the approval (and possibly pre-approval for common software) of that software.

3.1.4.4. Installed base of existing software

This question often comes down to a question of substitutability of software that otherwise fills a specific requirement. This is where vendor-neutral standards are critical, as discussed in my response to Q7.

3.1.5. Q5. How practical is No Charge Licensed Software? Are there hidden costs that need to be considered as part of the process of evaluating the alternatives available?

I do not have an answer to this question that is unique to "No Charge" compared to other licensed software. All software has costs that may not be not adequately taken into consideration by those evaluating it. If software is currently being evaluated using an acquisition process more appropriate for the acquisition of tangible assets, then missing these costs may be more likely. The price that is payed for licensing fees is often the least relevant cost when the full software life cycle is taken into consideration.

Vendor neutrality is on area already discussed, where the costs of transition away from the software to other software can be quite high. When properly evaluated, transition costs from vendor-dependent software is a greater cost to be considering than fees paid for licensing.

A wide range of software under a full spectrum of models from FLOSS, other NCLS and royalty-based sole proprietor software is already in use by the government. The practicality of this full spectrum is demonstrated by this existing use.

3.1.6. Q6. What are the general financial, technical and security risks associated with acquiring and using No Charge Licensed Software?

Financial, technical and security risks apply to all software, and are not unique to "No Charge" software. A full financial TCNO, assessment of whether the software meets a documented software need, as well as an independent security audit should be applied to all software no matter what business and licensing method is used by the supplier.

3.1.7. Q7. How do Open Standards and interoperability factor into evaluation considerations?

This criteria should be considered critical, and wherever possible software should be substitutable. When software adheres to vendor-neutral interfaces (preferably international standards) when it inter-operates with other hardware and/or software, then that component is able to be substituted by the

Crown over time. Software that uses vendor-specific interfaces, no matter how popular the software (i.e. mis-labeled "defacto standards"), should be seen as a form of anti-competitive tied-selling. This locks the Crown out of a competitive marketplace for substitutions of that specific software, as well as the marketplace for post-acquisition support. Shorter-term criteria should give way to the longer-term interests of the Crown of being able to make substitutions.

The full costs of transition from a vendor-specific interface to a vendor-neutral interface should be attributed to suppliers proposing the installation of vendor-tied software, or the continuation of an existing use of vendor-tied software.

To clarify this point: a software proprietor wishing to renew licenses after an existing contract has expired should be expected to include the costs of a transition from their vendor-tied software to recognized vendor-neutral standards. Only those vendors already using vendor-neutral standards would be exempt. This reevaluation of existing software usage should come into effect every few years, even if existing contracts have not expired, to not allow excessively long contracts to become a loophole for this policy.

3.1.8. Q8. How does the technology factor into the evaluation consideration, such as ability to maintain and evergreen?

I'm not sure this is a technology factor, but a licensing factor. Software that comes with full source code that is licensed in a way that allows the crown to make use of existing human resources and a competitive marketplace for maintenance has obvious benefits to the government. Software should not be thought of as a product that can be bought once and forgotten, but an ongoing conversation. Whether software is used internally or exposed to a network, software needs to be constantly enhanced to address security threats.

With FLOSS licensed software and methods the government is able to make use of resource multiplication where it does not need to pay the full costs of maintenance, but shares these with other organizations participating with the same software. This can all be done independently or in collaboration with the proprietors of the software code, as suits the government.

With sole-proprietor software it is only that proprietor that is able to maintain/evergreen the software, making all users dependent on their particular software release schedule, business plans, and eventually the very existence of the proprietor.

3.1.9. Q9. How does the Crown evaluate the flexibility of the licensing models for No Charge Licensed Software?

The Crown should be looking at the terms of any contract or license involved in this software acquisition carefully.

With FLOSS there is a popular subset of licenses which can be evaluated once and that knowledge then applied to all software using the same license.

In the narrow case that the government wishes to consider itself only an end-user of the software, there are no restricts at all with the use of FLOSS given the conditions within FLOSS licenses apply only to activities such as modifying and/or redistributing the software.

With non-FLOSS it is quite common for each software package, and often each version of a given software package, to have its own license agreement. Most non-FLOSS also has what is appropriately called an End User License Agreement (EULA) which unlike FLOSS licenses apply to the activities of end-users, and unlike FLOSS predominantly prohibit any ability modify and/or redistribute the software.

It should be noted that nearly all software, whether FLOSS or non-FLOSS, attempts to waive any liability or warranty. It is a common misconception that the acquisition of a non-FLOSS license provides indemnification and warranty. The reality is that these services are provided separately, and the government should be evaluating whether these separate services are available within a free market or only from a sole proprietor.

Any criteria for the acquisition of software should start with this basic knowledge of the range of licensing and availability restrictions to adequately evaluate the costs and benefits for the acquisition. I would be happy to work with the government to help create appropriate criteria.

3.1.10. Q10. What impact will No Charge Licensed Software have on Government Licensed End-User Networks (http://software.tpsgc.gc.ca/catalogue/index-e.cfm)

The current Software Information Portal appears to focus on legacy software acquisition methods. While with FLOSS a full range of support can be offered by any entity with the required expertise, the site appears to presume a sole proprietor model where only the vendor or proprietor of the software (and their resellers) can offer that support. This portal will need to be expanded to make clear that there is a full range of support options for FLOSS licensed software, including clarifying for those browsing the software options that there may already be adequate support offered by their own IT department.

This portal can be critical for documenting NCLS that has already been pre-evaluated by a department, documenting the justifications and the authentic source of the software. This would streamline the use of software such that the evaluation process does not become a productivity barrier.

4. Comments on RFI Appendix A

For nearly all the criteria I believe they apply equally to all licensed software. There appears to be a presumption that these criteria would be offered by the supplier of the software, as opposed to a possible third party provider. While there are many additional services that are tied to the vendor for sole-proprietor software, this is not the case for FLOSS.

Nearly all the checklist involves value-add products or services which in the case of FLOSS is not tied to the sole-proprietor, and can be available competitively in a free market. While the availability of suppliers, including in some cases in the form of human resources, should be part of the evaluation of the software, they can not be presumed to be supplied by the proprietors of the software.

It is important for commonly used NCLS to be pre-evaluated by the government so that duplicated effort in the evaluation process itself does not become a barrier to productivity. It is already common practice in government to use NCLS, and any new policy should enhance this experience and not deter from it. There are critical steps in the process such as verifying authentic sources and doing a TCNO analysis to determine hidden costs, but in the case of NCLS this evaluation can be done once and the evaluation (success or failure) adequately documented and published.

1.1 Software should obviously be legally licensed, and not be infringing

1.2 In nearly all cases, warranty is waived as part of software license agreements, whether FLOSS, other non-charge, or for-charge. It should not be presumed that a warranty is offered when software is purchased, given the only thing that is being purchases is a license. It is the software, not the license, which would need a warranty.

2.1 Transition costs should be tied to vendor-neutral standards. The supplier, including the incumbent supplier, or software that uses interfaces tied to a specific vendor should assume all transition costs from that vendor-dependent software to vendor-neutral standards.

3.6 Security is an item that deserves special attention. The ability to have a third party security audit done really requires open licensing in order to verify that it is truly a third party. A security audit done by the software proprietor or a business partner should be considered suspect. It is only when the software available for public peer review can critical flaws be found.

5. Comments on RFI Appendix B

Software having a license fee greater than $0 should not use a conventional procurement workflow. Software that does not qualify as "No Charge Licensed Software" should as a minimum meet all the requirements of NCLS, with an additional criteria requiring the evaluation and justification of the license fee.

5.1. Stream 1: Architectural Review and Approval

A clarification is needed around the suggestion the software should "not violate or overlap with any existing standards". The phrase "standard" has been overly-used within government to mean everything from an internationally recognized vendor-neutral standard set by a recognized standards to specific vendor choices made by departments and agencies. Vendor choices and/or popular choices (mis-named "defacto standards") should not be considered standards at all.

In the case of software standards the government should not force business models on suppliers, and ensure that any recognized standards can be implemented by a full spectrum of business methods and licensing. While there has not yet been standards which preclude implementation by sole proprietor software vendors, some of these vendors have pushed forward standards which have patent or other encumbrances which make it legally impossible to be implemented in FLOSS. The government should ensure that any recognized standard has at least two references implementations, at least one of which is FLOSS.

5.2. Stream 2: Financial Risk Assessment

I am not aware that such a financial risk assessment is currently being done for for-cost licensed software. It should be obvious that No Charge Licensed Software should not be held up against an assessment that is currently not expected of sole proprietor software.

5.3. Stream 3: Justification of No Charge Acquisition

I am not sure if this is appropriate as worded. Licensing options and software should be pre-evaluated as much as possible by the government, based on requests from departments and agencies. Since procurement officers tend to deal with the acquisition of tangible assets which have a non-zero marginal cost, something that is very different than software, they may not be the appropriate people to be doing the evaluation of software.

5.4. Stream 4: Investigation of Security Risks

It is incorrect to suggest there is a heightened security risk for downloadable No Charge Software. The correct statement is to suggest that the origins and authenticity of software must be verified, and that software only be brought into the government through adequately verified sources. This is true whether the software is FLOSS, No Charge non-FLOSS, or for-charge software.

What is needed is a process to be put in place to have the government evaluate various sources of software once, and provide documentation to all of government (and possibly citizens in general) about the level of risk.

I am unaware that a security risk assessment is done for current sole proprietor software. Conversations with many people in government have suggested that there has been a general blind trust of software vendors, and very little third party security audits of software already actively used within the government.

5.5. Stream 5: Software License Review

It has been indicated to me that in many (possibly most) cases a license review is not being done for current sole proprietor software actively used within the government. This is an appropriate review to do of all software to ensure that before software is installed on any computer that an adequate review of the rights and responsibilities attached to the government has been completed.

As suggested, in the case of FLOSS there are common licenses which are applied to many different software packages. This means that they can be evaluated and documented once, without needing to be re-evaluated for each software package which uses that license.

In the case of non-FLOSS (whether No Charge or other licensed software) it is quite common to use a custom license agreement for each software package, and often major licensing differences for different versions of the same software package. This suggests a need for the government to do a license review not only for the initial acquisition of the software, but potentially for each update (such as service packs) which modify the license agreement in any way.

6. Lessons from Agriculture Canada

Agriculture Canada is a science-based department (ie. a department with responsibilities for health, the environment, agriculture and agri-food, natural resources, fisheries and oceans, defense, etc), and as such may have more advanced usage of technology and knowledge of the scientific method. That said, I suspect that my experiences are not at all unique to this department.

While this may surprise people who do not work with technology within the Government of Canada, there are parts of government that are already advanced in the use of FLOSS and sometimes participation in FLOSS communities through both government initiated software projects as well as third party projects.

At Agriculture Canada I was hired into the IM/IT support team[8] within a project called the National Land and Water Information Service (NLWIS)[9]. NLWIS provides a series of interactive maps[10], with the most recent maps offered via what is called the Agriculture Mapping Application Framework (AgMAF). My roll is to work in the team which offers the infrastructure upon which applications are installed, and to do the version management and installation of the application in this infrastructure. The application is being developed by people within NLWIS.

This framework is a mixture of both sole proprietor software and FLOSS. The most dominant sole proprietor vendor in the Geospacial marketplace is Environmental Systems Research Institute, Inc. (ESRI)[11] , and the most prominent FLOSS group is the Open Source Geospatial Foundation (OSGeo)[12].

The data used by our interactive maps are stored in an Oracle Database which is geospacially extended using software from ESRI called ArcSDE (SDE stands for Spacial Database Extension). This geodata is then accessible concurrently to the ESRI stack of software as well as though an extension to OSGeo's Mapserver program. Other OSGeo software used include OpenLayers (Javascript on the client) as well as GDAL/OGR. We also make use of additional FLOSS components such as Tilecache[13].

The servers primarily involved in these interactive maps run RedHat Enterprise Linux. This is further enhanced by the Science Infrastructure IT team into an Enterprise Technical Applications Framework (ETAF) which enables enhanced scalability and security by allowing them to isolate software components to be easily moved between servers without breaking software dependencies.

As part of NLWIS IM/IT Support I use desktop Linux. I also connect to Citrix servers to run any software dependent on Microsoft Windows that I may need for my job. By being able to combine both Linux and Windows applications on the same desktop I am able to be far more productive than I could if I were forced to use a Windows desktop, or forced to have two computers, monitors and keyboards at my desk.

I listed the above to introduce the amount of FLOSS already in production within this department. I also wanted to describe the infrastructure in order to use two software issues as examples of the differences between out ability to solve problems associated with FLOSS components and the non-FLOSS sole proprietor components.

---

8 For those who follow government hierarchies, that is: Government of Canada, Agriculture and Agri-Food Canada, Information Systems Team, Applications Development Directorate, NLWIS IM/IT Support. http://direct.srv.gc.ca/cgi-bin/direct500/XEou%3dNIS-SSG%2cou%3dADD-DDA%2cou%3dIST-ESI%2cou%3dAGR-AGR%2co%3dGC%2cc%3dCA (Retrieved February 2009)

9 National Land and Water Information Service http://www4.agr.gc.ca/AAFC-AAC/display-afficher.do?id=1226330737632&lang=eng (Retrieved February 2009) Most current page redirected from http://atlas.agr.gc.ca

10 List of interactive maps (Retrieved February 2009) http://www4.agr.gc.ca/AAFC-AAC/display-afficher.do?id=1228838087110&lang=eng

11 Environmental Systems Research Institute, Inc. (ESRI) http://www.esri.com/ (Retrieved February 2009)

12 Open Source Geospatial Foundation http://www.osgeo.org/ (Retrieved February 2009)

13 TileCache is an implementation of a WMS-C compliant server made available under the BSD license by MetaCarta. http://tilecache.org/ (Retrieved February 2009)

6.1. Character Encoding

Symptom: Unicode encoded strings in the database were coming out blank

Details: Recent versions of Oracle and the ESRI database tools make use of a new "National" string (nvarchar in Oracle, NSTRING in ESRI's tools) data type to store strings in Unicode. Unicode is also the character encoding that is the departmental standard for Agriculture Canada.

When NSTRING data from the database was requested by Mapserver, where the value of the string should have been substituted was instead blank.

As we had source code to Mapserver I was able to use debugging tools to pinpoint the problem. There were two different problems.

- There as a minor bug in the build process for the gd graphics library[14] that mapserver depends on, which was causing failures for any attempt to transcode strings. This problem was already fixed in the latest gd library release candidate, and the fix is easy to apply to any previous version.
- ArcSDE provides mapserver with Unicode as UTF-16. UTF-16 can have different orders of the bytes (endianness[15]), and it turned out that the developers of Mapserver were using libraries which assumed the order of the local computer (in this case an Intel computer which uses little-endian) while the library we were using assumed what is called network order which is big-endian. This fix was to have Mapserver specifically indicate the endianness to the library. The fix for this problem is now part of the mapserver source code, and will be part of the next release (expected soon). We already have a patch which we use at NLWIS.

The take-away from this experience is that with the availability of the source code we were able to determine the problem and resolve it based on the business requirements of NLWIS. While we interacted with the software project, we were not dependent on them to make the fix. I was able to interact directly with the person who authored the code within mapserver that needed to be fixed.

14 GD library project is at http://www.libgd.org (Retrieved February 2009)
15 Wikipedia provides a good discussion on endianness at http://en.wikipedia.org/wiki/Endianness (Retrieved February 2009)

6.2. Compressed RASTER data in database

Symptom: When mapserver attempts to access data stored in compress RASTER format in the database, the program crashes (core dump)

Details: In order for mapserver to be able to decompress data it makes use of the zlib[16] library. Unfortunately, the ESRI provided library we need to use to talk to ArcSDE has embedded within it an incompatible version of zlib. When the ArcSDE library tried to decompress information from the database it would find the incompatible version of the library we were providing and crash.

While other programs have been known to embed zlib within their software, they do this in such a way that only their own software uses that embedded version, and they do not make the functions of the library available externally in a way that taints other software.

In this case the version of zlib embedded within the latest release of ArcSDE is version 1.1.3 which was released July 9, 1998. This version is known to have serious security vulnerabilities, and the zlib homepage recommends everyone upgrade to version 1.2.3 which was released July 18, 2005.

I verified that this was the source of the program crash by compiling our software with the 1.1.3 version of the library. All tests passed. It is not a viable solution to use this library given it has known vulnerabilities and for obvious security reasons will not be installed on production servers by the Science Infrastructure IT team.

I was able to diagnose this problem because of the availability of source code for all the FLOSS components. I am aware of the various steps needed for the ultimate solution, but we are dependent on ESRI and their priorities for any fix. From my experience with compiling the FLOSS components against multiple versions of this library, one quick temporary solution is as simple as ESRI downloading and using the most recent version of zlib. Someone with ESRI Technical Marketing has indicated that this upgrade will be done in time for version 9.4 of their suite[17], but that may be more than a year in the future and is uncertain. Many major releases and service packs to the ESRI software have been released between when they should have been aware of the zlib issue in 2005 and today.

Ultimately the correct fix is for ESRI to not taint the external environment by exporting the functions offered by any embedded version of zlib, as well as ensuring that its library is not dependent on the external version.

The types of information that I discovered would be appropriate as inputs into a security and TCNO evaluation of the relevant software.

16 Zlib is a FLOSS compression library, licensed under a BSD-style license http://www.zlib.net/ (Retrieved February 2009)
17 Publicly archived thread discussing this issue on the ESRI support forums http://forums.esri.com/Thread.asp?c=2&f=1718&t=212867 (Retrieved February 2009)