Developing an open source policy

Sebastian Rahtz

OSS Watch, Oxford University Computing Services, UK

sebastian.rahtz@oucs.ox.ac.uk

Abstract

Using the UK as an example, this paper considers the relevance to an educational institution's IT strategy of the UK Government's Open Source Policy, the UK e-Government Interoperability Framework (e-GIF), and the open source policies of the UK Joint Information Systems Committee. It proposes a statement of principles suitable for any educational institution.

1 Introduction

Most educational institutions do not have a section in their overall IT Strategy which defines their attitude towards open source software. The intention of this paper is to outline the areas such a policy section should cover and what it might say. Although open source *per se* may not initially seem important, dealing with it properly covers a wide range of subjects of interest to any institution.

An institution's open source policy will affect, or be affected by,

- institutional or departmental software acquisition
- institutional, national, or international standards for data storage
- research contracts, and exploitation of inventions
- staff employment contracts
- IT training plans

All of these areas will naturally be dealt with in the institution's IT strategy, but the challenge is to ensure that decisions and policies are not dictated by any one model of software development and deployment.

We will first consider the characteristics of open source and why it is of interest to educational communities; then look at the guidance given by the UK government and one of the big IT funding bodies; and conclude with a proposal of what should go into an institutional strategy.

2 How does open source work?

Open source (`http://www.opensource.org/`) is computer software for which:

- the source code is **available** to the end-user;
- the source code can be **modified** by the end-user;
- there are no restrictions on **redistribution** or use;
- the **licensing conditions** are usually intended to *facilitate* continued re-use and wide availability of the software, in both commercial and non-commercial contexts;
- the **cost of acquisition** to the end-user is often minimal.

The virtue of open source software is that there are no secrets, and the working of the software is available for anyone to inspect, something of great importance for security audits. The fact that the software is generally not privately controlled means that it is likely to promote open rather than proprietary formats; and the model of development by communities often leads to problems being fixed quickly. The fact that open source software is almost always distributed free of charge is in some ways irrelevant, since the cost of deploying software is very often much higher than the simple cost of licence acquisition.[1] As with proprietary software, open source requires hardware to run and trained people to customise, install, upgrade, patch, test and maintain it. Many of these tasks are often indirect and a harder cost than the up-front licence fee which is often the most noticeable cost of proprietary software.

Deployment of open source software is, generally speaking, no different from that for proprietary software. It will follow the same pattern of evaluation of needs, testing, acquisition from a supplier and so on. The main differences are *additions*:

1. There can be multiple, independent, primary vendors for open source software to choose between.
2. As well as the usual range of support options (relying on a primary vendor, dealing with a large general support company, employing a specialised consultancy, and developing in-house expertise), there is the additional possiblility of bringing new feature development in-house and modifying the core source.

Open source can work as a business model for software developers for various reasons. Firstly, only a small proportion of a given piece of software is genuinely original and saleable; the infrastructure around it (storage and communication, for example) might as well consist of an implementation of open standards; if so, it makes sense to share development of this implementation with other companies in a non-competitive manner. Secondly, pyramidal consulting works: making software open makes it easier to devolve support. Lastly, funding small improvements to open source software is economically more

[1] The 'Guide to Open Source Software' issued by the Australian Government[1] deals well with the assessment of the total costs of deploying an open source solution.

efficient than engineering a new solution. Developers work on the things the business cares about and understands. Of course, the revenue margin is likely to be smaller; some estimates are 85% margin on selling licences, and 54% on selling support.

Some open source business work on a dual-licensing model, which allows them to offer a normal open source licence to most customers, and sell support at the same time, but to offer the same software under a closed licence to customers who require the facility to embed the code in their own proprietary systems.

Open source projects (probably the majority) which are not developed as part of a traditional business tend to be classified into two extreme camps:

1. A genius programmer creates something which others like. (S)he controls pace and scope of future development. The genius has a complete idea of what the finished software will look like. Changes to the design during the build process are very costly, but the resulting software very polished and uniform.
2. A group of people get together to fill a gap and take on different tasks. As time goes by, some drift away and others join in; there is always someone to take over. Design is by consensus, with people working on topics that motivate them. Design changes are relatively cheap but with no over-arching plan the finished program can be patchy.

Most projects fall in between. Most of them have a recognised leader, to whom other members defer. A gentle system of leadership challenge and deposition assures the health of the herd is kept up.

The FLOSS study led by Rishab Ghosh (`http://www.infonomics.nl/FLOSS/`) is the best known of a series of studies by economists on how open source works. These studies show that people join an open source project because (in decreasing order of importance):

- it looks technically interesting
- it solves a problem they have
- it looks like an important project
- they know other people involved

The desire to learn technical skills by joining an open project is strong. Typical reasons for staying in OSS are:

- improving skills
- ideology
- improving software
- seeking recognition

Understanding these motivations is important in the context of educational institutions, because staff and student commitment and interest is regarded as a priority. The fact that working in open source is regarded as a *training* opportunity is a real consideration.

3 The perspective of the educational institution

Open source matters to higher and further educational institutions for two main reasons: because it can be cheaper, and because it can be a better match to educational requirements and outlook. As publicly-funded agencies, educational institutions have a legislative responsibility for fiscal responsibility—they have to spend their money wisely. Open source licensing and development methodology may also mean that a piece of open source is better suited for purpose within an education institution than similar proprietary software. Some specific reasons are:

1. The ability to tailor the system completely to local needs. Both open and proprietary software are typically both customisable in a shallow sense—institutions can tailor the interface within the bounds given by the programmers. With open source, if an institution needs the software customised in ways not thought of by the programmers, they can have the program changed, either in-house or by a third party.
2. Lack of surprises. Open source licences are free and perpetual, so a licence fee increase cannot happen.
3. No incentive for theft. Not uncommonly, when proprietary software is used in an educational setting, particularly on the desktop, there is pressure on the students to have the software available at home, and this can lead to theft. With open source software students can download the software free of charge from the internet for their own use.

Apart from these immediate business potentials, how does it benefit an institution if it's staff are involved in open source? One aspect is potential deep engagement with computer-oriented specialisms. The source code to all open source software is available, so computer-related specialists (computer science, software engineering and hardware engineering) are able to engage very deeply in it. Source code and design discussions can be lifted from the very software that students use every day for use in lectures, assignments, project work and academic research. Even better, students can engage with the communities building that software, both to learn the dynamics of software engineering and to commit their own changes back to the project. On the staff side, we can see advantages in continuing professional development for those who get involved in projects, and good publicity for the institution.

As with evaluating the costs of software, the relative benefits of these will from institution to institution and unit to unit within an institution. In most scenarios open source open source use tends to encourage in-house (or at least local) skills development and spending on human recourses rather than up-front annual fees.

Most institutions are likely to have a dependency of some sort on open source software already. This may be in the form

of back-room network systems (DHCP servers, DNS, email relays); of web servers, portal frameworks, and VLE systems; of specialized departments relying on statistical or modelling software; of units doing typesetting using free systems; of staff using web browsers or email clients; or of students studying open source operating systems. While the deployment of proprietary software is often well accounted for, by the counting of licences and auditing of machines, few institutions have seriously looked at either the dependency on, or the cost of, open source.

Figure 1, Proximity between users and software shows the varying relationships between people and software. In the simplest case, the person deploys the software personally; for more complex software, there is layer of administrators and managers in between; in the most complex case there is an interaction between a content developer, the course administrators, the system administrators and the developers adding new functionality. In all but the first case, the person at the top has no reason to know or care that the software at the bottom is open source; only the intermediate layers of support staff may be concerned with the provenance of the software. In the case of the VLE, the interaction at the top layers relates to data and interoperability standards, not the software itself. Thus from one perspective the dependence on open source is at once both hidden and irrelevant, but it's existence may underly crucial parts of the institution's work.

Staff and students in an educational institution sometimes feel that they need not concern themselves with issues of copyright and licensing, and that traditional academic sharing of information will suffice. One of the successes of the open source movement has been to alert particularly computer science professionals about the real importance of properly recording intellectual property rights, and clear licensing rules.

Although 'free software' is sometimes perceived as being anti-capitalist, in fact the reverse is true, and the open source movement is bringing legal rigour to what has sometimes been a poorly-organised area of engineering. This has an important effect on the institution, as it must now provide staff working on open source with clear guidelines about their legal rights. Many academics (staff and students) currently participating in open source projects are not yet aware that they may not be permitted to release institutional intellectual property, by the terms of their contracts.

Open source projects are about more than low-level programming. The usefulness of a system depends not only on its functionality, but also on its user interface, documentation, publicity, error-reporting, and peer-group support. This means that a much wider variety of staff and students than expected may potentially be involved in open source; in some cases without really being aware of it. The casual user of an open source statistical package who maintains a set of usage tips on their home web page is both contributing to the project, and committing institutional resources to it.

Educational institutions have a lot to gain from open source, and must also accept the consequences of their current heavy commitment.

4 Public policies in the UK

4.1 e-GIF

The 'e-Government Interoperability Framework' (e-GIF)[4] is the set of technical policies and specifications mandated for interchange of information between UK Government, its citizens, business, organisations and other governments. It is part of the UK government's overall e-Government strategy, promoted by the e-Government Unit. The framework uses the web for delivery, and XML-based standards for content and metadata.

e-GIF has four main parts:

1. A list of underlying standards, essentially XML as data delivery format, and XSL as transformation language for presentation; specification of XML is more or less universal in initiatives like this. Web services are specified to follow SOAP, UDDI, and WSDL; data modelling is to use UML; data transport is to use Unicode (UTF-8);
2. A set of W3C XML Schemas for public sector information; some of these have been defined already, but there is a lot of work remaining to be done;
3. A metadata standard, close to the internationally-used Dublin Core;.
4. Procedures for keeping the framework up to date with managed change.

The standards for information access are conservative. They allow for web browsers back to and including Netscape 4, PDF, Rich Text Format, Word documents and Macromedia Flash. Perhaps surprisingly, Excel spreadsheets and Powerpoint documents are *excluded*.

Key decisions underlying this list include:

- adoption of XML, and Unicode, as the underlying data standards;
- recognition of the web as the primary delivery format;
- recognition that interoperability is best achieved by use of open standards.

The e-GIF specifications are now at version 6, and work continues on their details and implementation.

4.2 UK government open source policy

The policy 'Open Source Software: Use within UK Government'[6] was published by the e-Government Unit of the UK government's Cabinet Office in October 2004, after research (e.g.[9]), consultation with government agencies, software companies, and open source advocacy groups. The UK

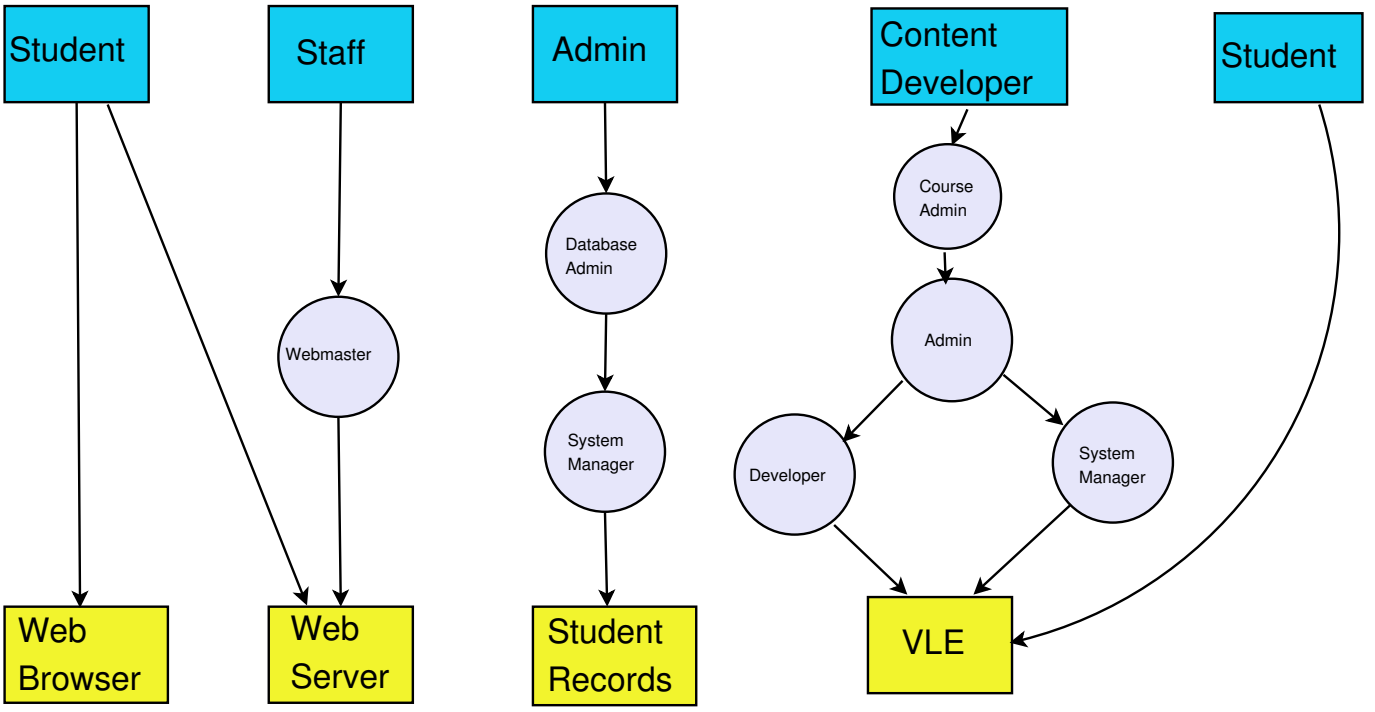

Figure 1: Proximity between users and software

Government in this context includes all publicly funded organisations, i.e. central government departments and their agencies, local government, and the education sector. The policy is part of the UK implementation of the EU plan 'eEurope 2005: An Information Society for all'[5], which mandates the governments across Europe to implement interoperability frameworks based on open standards and encouraging the use of open source software.

The policy is quite short, with three key decisions (cited verbatim from the policy) relevant to the discussion in this paper:

- *UK Government will consider OSS solutions alongside proprietary ones in IT procurements. Contracts will be awarded on a value for money basis.*
- *UK Government will only use products for interoperability that support open standards and specifications in all future IT developments.*
- *Publicly funded R&D projects which aim to produce software outputs shall specify a proposed software exploitation route at the start of the project. At the completion of the project, the software shall be exploited either commercially or within an academic community or as OSS.*

Unsurprisingly, there is an exemption for the areas of defence, national security and law enforcement. The last one is the most important for us, as it establishes the vital point that all software created with public funds must be 'exploited' in some way, and not left in limbo. The choice is between commercial exploitation, OSS, or 'within an academic community'. Unfortunately, the precise meaning of this last option is not discussed in the policy, and must cast doubt over the useability of the process, since almost any form of restrictive licensing may be covered by this clause.

The e-Government Unit is preparing guidelines on the specific issue of how to choose which licence to use if an OSS route is chosen for exploitation.

Published at the same time as the policy was a report[7] on trials of open source software in government, which concluded that 'Open Source software is a viable and credible alternative to proprietary software for infrastructure implementations'. Note here the word 'infrastructure', since the report also notes that open source implementations of enterprise-level desktop products are not yet mature enough. The report states that significant cost savings *can* be achieved by adopting open source (by reducing licensing costs and hardware upgrade requirements), but stresses that planning of migration, consideration of interoperability, training of users, and development of skills for implementation and support will be significant factors in the short term.

The trials report recommended, among other things, that public sector bodies should take the following actions (cited verbatim from the report):

- *examine carefully the technical and business case for implementation of Open Source software and the role which OSS could play in current and future projects, . . . ;*
- *review the potential for server consolidation, comparing the benefits of OSS with proprietary solutions;*

- *consider the potential costs and benefits of migration to an OSS desktop for transaction users, (potentially in conjunction with use of 'thin client' architecture solutions);*
- *identify the role of open standards in future IS/IT strategy and policy, in conformance with the e-Government Interoperability Framework (e-GIF);*
- *consider requirements for the development of skills in Open Source development, deployment and operation within the organisation, and review the availability of such skills in their outsourced IT service providers;*
- *review their current infrastructure and applications ... well in advance of any planned procurement or renewal, and determine whether current technologies and IT policies inhibit future choice; and if so consider what steps may be necessary to prevent future 'lock in';*
- *consider the benefits of incremental change by diversifying OSS use beyond the server platform to products like Email, LDAP, Web and internet browser.*

There is little in the UK government policy, or the results of its trials, to cause much surprise. They recognise open source as a valid exploitation and deployment method, and note its useful role in giving publicly-funded bodies more choice and potential cost savings; while also stressing the importance of interoperability and open standards for data.

4.3 JISC open source policy

The Joint Information Systems Committee (JISC) is tasked with funding an IT infrastructure for UK higher and further education, and its work ranges from supplying the underlying physical network (JANET), directly supporting network services in FE colleges, running advisory services (such as OSS Watch), to funding research into interoperability standards for learning objects. Within this spectrum of work, there has been an increasing recognition that viable long-term software products can emerge from research funding, and that the JISC did not have a coherent policy on how the licensing and exploitation of that should be manged. During 2004, work undertaken by the OSS Watch advisory service on proposing standards for software projects was turned into a coherent policy, which was then submitted to JISC committees for approval in February 2005.[8]

This policy (not yet published) is in three parts: a) guidelines for JISC services and projects in general; b) guidelines for writing calls for funding; and c) detailed guidelines for how to manage the software creation process. The main aim of the first two sections is to establish an environment in which open standards and interoperability are promoted, and avoiding issues regarding digital rights. The third section gives specific rules for the following areas, for which we supply a summary of the policy:

Copyright It is of paramount importance to establish who owns what, using a formal IPR Register.

Licensing All software generated by JISC projects should be released under an open source licence, unless an explicit alternative is proposed in the bid.

Trademarks Use of trademarks to establish reputation and trade on association is up to the project, not the JISC.

Patents Any patent applications associated with the project should not interfere with free distribution of software.

Dependencies Projects must record which associated software is needed to make their work run.

Archiving All documents and software code must have a preservation and archiving strategy.

Testing and quality assurance All software must have a testing framework in place, and demonstration of standards compliance.

Version control All software must be developed using version control software, and the history must be preserved by the project.

Sustainability and communities Projects should, where appropriate, encourage and support user and development communities.

Documentation Documentation must archive all forms of documentation, including mailing lists and forums.

Software development and maintenance Software should follow good engineering practice, and be demonstrable to, and testable by, peer communities.

As with the government policy, there is little here which goes beyond common sense and good practice. The most onerous statement is: 'copyright of software, documentation, design materials, manuals, user interface and source code must be released under an OSI-approved open source licence, unless the bid explicitly argues why this should not be the case and proposes an alternative licence.' This puts the burden on a project to consciously *reject* the open source route at an early stage if it wishes to pursue alternatives. However, the JISC has long maintained a condition that software it has funded must be publicly available, for any use and at no financial cost, throughout UK higher and further education; the simple route of selling licences to peer institutions is ruled out anyway.

The JISC does not make any conditions or suggestions about which open source licence to use, because this is often a contentious area.

5 A policy for educational institutions

The primary concerns for an educational institution's IT procurement strategy should be *demand* (that is to say, why do we need the system) and *value* (what will it cost us). Beyond that, the single most important consideration is the preservation of data and the interoperability of systems; this coincides with (in the UK) government policy. Thus:

1 *New software acquisitions should demonstrate conformance to open standards and interoperability with open systems.*

At each point on the procurement and deployment chain, software should be assessed on its merits. Thus:

2 *Open source and proprietary software options should be assessed using the same criteria, considering of total cost of ownership over the expected lifetime of the deployment.*

With respect to intellectical property created for the institution by the writing of software, an institutional IPR policy should acknowledge the significant role played by open source methodologies in terms of potential exploitation routes for the institution's technology transfer arm. This expands the possible range of material which can be exploited, and it is no longer necessary to regard the bulk of software IPR as 'unexploitable'. However, the exploitation route must not exclude the developer sharing in derived income. Thus:

3 *Software development by staff and students must maintain a register of intellectual property rights (IPR).*
4 *Software for which the copyright belongs to the institution must be exploited.*
5 *Open source licensing must be available as an exploitation method, and will be the default method where no alternative is proposed.*
6 *Income derived from services and training associated with an open source product must be shared with the developers using the same system as that used for patents and licensing.*
7 *The open source licence chosen should ensure that the institution is able to freely use all future versions of the software.*

The intent of the last clause is to make sure that software developed locally and currently in use is not folded into a proprietary product for which the institution may then have to pay.

Initiation of entirely original software is less common than adding to an existing product. Participation in, and contribution to, open source software projects should be a normal part of the working life of IT-related staff, and this must be reflected in employment policy and employment contracts. There must be procedures in place so that staff can do work on open source projects in good conscience, without removing the protection afforded to the institution by retention of copyriht. Thus:

8 *A register of offically-deployed open source software must be maintained for each unit.*
9 *A register of open source software for which staff may contribute code, documentation and support must be maintained for each unit. It must say whether contributions remain the property of the institution, or whether copyright has been assigned to a body maintaining the software.*
10 *Staff and students may deploy additional open source software for research or teaching, but may not contribute institutional intellectual property to it without explicit permission.*

6 Conclusions

Open source is neither a threat to, nor a panacea for, the IT needs of educational institutions. It is an opportunity to expand the range of ways they deploy and develop software, and they should take the opportunity to re-examine their IT and IPR strategies.

Acknowledgements

This paper was prepared by OSS Watch, a UK JISC service funded for 2003–2006. OSS Watch (http://www.oss-watch.ac.uk) provides unbiased advice and guidance about free and open source software for UK further and higher education. It is hosted by the Research Technologies Service at the Computing Services, University of Oxford.

Unless directly quoted, ideas and opinions expressed in this paper are those of the author and do not represent the policy or intentions of JISC or of the University of Oxford.

References

[1] *A Guide to Open Source for Australian Government Agencies*, The Australian Government Information Management Office, 2005. [http://www.sourceit.gov.au/__data/assets/pdf_file/42065/A_Guide_to_Open_Source_Software.pdf]

[2] *Open-source software - in e-government*, Danish Board of Technology, 2002. [http://www.tekno.dk/pdf/projekter/p03_opensource_paper_english.pdf]

[3] *DTI Open Source Software Factsheet*, UK Department of Trade and Industry, 2004 [http://www.dti.gov.uk/bestpractice/assets/oss.pdf]

[4] *e-Government Interoperability Framework (e-GIF)*, The Cabinet Office, 2004. [http://www.govtalk.gov.uk/interoperability/egif.asp?order=title]

[5] *eEurope 2005: An Information Society for all*, European Commission, 2004. [http://europa.eu.int/information_society/eeurope/2005/all_about/action_plan/index_en.htm]

[6] *Open Source Software: Use within UK Government*, e-Government Unit, 2004. [http://www.govtalk.gov.uk/policydocs/consult_subject_document.asp?docnum=905]

[7] *Open Source Software Trials in Government - Final Report*, e-Government Unit, 2004. [http://www.ogc.gov.uk/index.asp?docid=2190#finalreport]

[8] *Policy on Open Source Software for JISC Projects and Services*, UK Joint Information Systems Committee, February 2005 (unpublished).

[9] *Analysis of the Impact of Open Source Software*, QinetiQ, 2004. [http://www.govtalk.gov.uk/policydocs/consult_subject_document.asp?docnum=781]