NICS FOSS Masterclass

Dr Colin Turner
Head of School, School of Engineering,
University of Ulster
c.turner@ulster.ac.uk

17th May 2011

Outline

FOSS and its usage

FOSS is Free and Open Source Software.

How does it relate to you?

- Do you create FOSS?
- Do you use FOSS?
- Do you procure FOSS?

Context of this talk

On the 15th October 2008, Sir Reg Empey opened the Open Source Solution Centre in the SRC in Newry. His speech provided some useful local context.

Sir Reg Empey

Open Source Solution Centre, 15th October 2008

"... Today's event is a tangible illustration of further progress and is a timely development, given the strategic importance of Open Source software to the long term sustainability of our software sector.

It is clear that Free and Open Source Software will become fundamental to building and maintaining future market share. The FOSS model can therefore provide start-ups with a fast and efficient way to build a client base and thus gain market advantage. "

Sir Reg Empey

Open Source Solution Centre, 15th October 2008

“... Open source also adds value to the Northern Ireland software industry by providing impetus to their product and process innovation, resulting in faster delivery.”

“... The days of conventional commercial software companies holding a monopoly position over society are over.”

“... I would like to reinforce the message that Open Source solutions will in the future represent an important contribution to building and maintaining market share for the business and other sectors. The fact that all of us are gathered here today demonstrates the commitment of government to embrace this new climate of opportunity.”

Neelie Kroes

European Union Competition Commissioner, 10th June 2008

"I know a smart business decision when I see one - choosing Open Standards is a very smart business decision indeed."

"There is a democracy issue as well. No citizen or company should be forced or encouraged to use a particular company's technology to access government information. No citizen or company should be forced or encouraged to choose a closed technology over an open one, through a government having made that choice first."

Terms and their usage

It's worth noting at this early stage that we should not take it for granted that people understand the meaning of free or open source software. Even in the FOSS community, there are some divergences of opinion and usage here.

So, with apologies to those for whom this is all well understood, we shall introduce the definitions in most common usage.

Free as in Freedom

The Oxford English Dictionary (Reference Version) has 19 definitions of the word free. Here are a few:

- unrestricted, unimpeded; not restrained or fixed;
- unconstrained;
- available without charge; costing nothing.

Most people think the last of these is what “free” software pertains to, but no, it is the former two which have a higher priority that the latter, although often strong restrictions on cost exist. As the FSF (`http://www.fsf.org`) say, it is Free as in Freedom.

Terms and their usage

The Free Software Foundation (`http://www.fsf.org`) defines free software as providing you with the following freedoms with software:

Freedoms in Free Software

0. the right to run it for any purpose;
1. the right to study it and adapt to your needs; (which implies open source)
2. to redistribute it, so you can help others;
3. to release improvements, so everyone benefits.

Specifically, note, money is not mentioned, but freedom 2 and 3 normally imply freedom from cost.

Terms and their usage

Other types of software include

- Open Source:
  Most people use this term synonymously with free, but some mischievous people use this to literally mean: read access to the source.
- Custom :
  Software created “in house” which isn’t being licensed for distribution - closed source, but without ethical entanglements normally.

... and

Terms and their usage

- Proprietary :
  The “normal” model, software that is licensed, often in extraordinary ways that grants very little in return for the money. Having said that, again money might not be involved at all here, a lot of “freeware” and “shareware” software still grant none of the freedoms mentioned above.

“Crowley had been extremely impressed with the warranties offered by the computer industry, and had in fact sent a bundle Below to the department that drew up the Immortal Soul agreements, with a yellow memo form attached just saying: ‘Learn, guys’”
Pratchett & Gaiman, Good Omens

Money

Note that money is not part of the definition here.

Software we would have (and still would) cause Freeware and Shareware is not necessarily FOSS.

FOSS (libre software) can, and is used to make money.

Proprietary software can be free of charge (gratis software).

Why is Freedom important?

It might seem that this is a fairly academic distinction. After all, if the software is free in price, why should we care? Software that is free in price but not in licence is usually that way for a reason.

- Internet Explorer, alleged to be free in a simple (successful) bid to crush Netscape,
- Media Player, alleged to be free to crush the opposition.

These products have tended to help the market position of MS Windows (understandably, to be fair to Microsoft that was they they were developed), but such commercial power has been the undoing of many competitors (e.g. OS/2). Is a mono-culture a healthy thing?

Why is Freedom important?

There are other reasons:

- Mozilla rose from the ashes of Netscape, and now Mozilla Firefox is a highly popular extendable browser with open source so that it is available in almost identical form on many platforms.
- In 2004 the developer of the PWC (Linux) webcam driver withdrew his propietary driver (which was only available for the x86 platform) and his open source driver. It was judged by the kernel developers that hooks to closed source drivers could not be included for reasons of maintenance and cross platform support.

Why is Freedom important?

- Free software implies free, or at least open standards, allowing users to avoid lock-in with one vendor. Consider that many governments are now insisting on only "open" formats being used to store information in their work. Why?
- What happens when the software you rely on for your research project belongs to a company that goes bust?
- Free software is often considered more secure. This is security by design rather than security through obscurity. More pairs of eyes can see the code in free software applications, and bugs are not hidden.

How much free software exists?

This talk was prepared entirely with free software:

- on a free operating system (GNU/Linux (Debian));
- with a free editor (Emacs), using free extensions;
- compiled with a free document preparation system (LaTeX);
- with free extension packages (beamer);
- checked with a free PDF file viewer (xpdf, evince);
- occasionally I looked up resources with a free browser (Mozilla Firefox).
- and I store the files in my version control system (subversion).

And for almost all of these tasks I had several free alternatives. I could simply have used OpenOffice.

FOSS and the public sector

There are a number of ways in which the public sector could, and perhaps should, relate to FOSS.

Relationships of the public sector to FOSS

1. As a consumer of software.
   Clearly, the sector procures vast quantities of software, should some of it be FOSS?
2. As a creator of software.
   Custom software is created in large quantities, we should examine the pros and cons of making it FOSS.
3. As a consumer of ideas.
   The methodologies used in FOSS are starting to be used in industry. They promote team work and transparency.

Open Standards

A highly related idea is that of an Open Standard. This is a common standard that many companies can implement against. Consider for example TCP/IP and HTML.

Open Source implies Open Standards, since an inspection of the code allows data structures and storage to be revealed. However, it is possible (of course) to have closed source implementations of open standards.

Many of the issues surrounding "code" can also be applied to standards, knowledge and other diverse items including hardware.

Open Standards

It is often suggested that publicly funded work should make heavy uses of Open Standards.

One reason is to ensure that a citizen should not have to purchase a particular proprietary technology to access services.

This avoids the general problem of vendor lock-in which could be particularly acute with public data.

FOSS does not imply a language

FOSS can be platform and language agnostic. FOSS implementations of virtually every development language exist on virtually all major operating systems.

In fact, a huge strength of FOSS is that is can bridge many platforms. For example, Linux supports x86, amd64, ARM, MIPS to name a few.
See: `http://en.wikipedia.org/wiki/List_of_Linux_supported_architectures`

This allows Linux to inhabit markets from embedded devices, WIFI routers, phones (Android and others), Desktop computers, Servers right up to Super computers.

The Free Software Foundation

In 1984 Richard Stallman (author or the emacs editor) launched the Free Software Foundation or FSF (`http://www.fsf.org`).

- Software produced by the FSF was often branded GNU. GNU stands for Gnu's Not Unix and was an early example of one of many self-referential acronyms to come;
- The FSF intended to produce a complete free operating system — which was to be codenamed the Hurd. The intention was that this would be a free Unix like system.
- The software would be released under a licence known as the GNU General Public Licence or GPL (`http://www.gnu.org/copyleft/gpl.html`).

The Free Software Foundation

Currently, the most recent version of the GPL is v3, but v2 is still widely used. It is one of the licences used for free software, though there are many others. See `http://www.gnu.org/licenses/gpl-faq.html`.

- Not all GPL programs are GNU;
- Users are free to modify the GPL code as they wish;
- The licence ensures that any improved, or modified version of the code must also be released under the GPL (it cannot be closed);
- It is permissible to sell GPL programs;
- However, anyone with the software has the right to give it away.

GNU's not unix

- Years passed and Gnu produced a great deal of excellent and famed software, mostly under the GPL;
- Much of this software runs on Unix distributions of all makes;
- The Gnu C Compiler (GCC) becomes extremely popular (although now GCC stands for the Gnu Compiler Collection since it supports many languages);
- But the OS kernel does not appear in any meaningful way.

The rise of Linux

And then...

- In 1991, a Finnish student called Linus Torvalds began work on a new kernel to provide a freeware unix kernel for people to play with.
- It was (perhaps) based on an attempt at a free version of Minix, which was a x86 based kernel for students to explore, produced by Andrew Tanenbaum.
- Linus initially claimed that it would never be portable, but at the time of writing it is ported to many platforms and version 2.6.38.6 of the kernel is current. Linux is released under GPL v2.

This filled the missing gap in the movement.

Tensions in the Open Source / Free Software Community

Even this altruistic bunch of people have their own politics. At the front page of the GNU website (`http://www.gnu.org`) you will find a reference to why Linux Operating Systems should be known as GNU/Linux Operating Systems at (`http://www.gnu.org/gnu/linux-and-gnu.html`).

Richard Stallman on “Linux”

“If we tried to measure the GNU Project’s contribution in this way, what would we conclude? One CD-ROM vendor found that in their “Linux distribution”, GNU software was the largest single contingent, around 28% of the total source code, and this included some of the essential major components without which there could be no system. Linux itself was about 3%. So if you were going to pick a name for the system based on who wrote the programs in the system, the most appropriate single choice would be “GNU”.”

Linus Torvalds on “GNU/Linux”

“About the GNU/Linux argument; have you talked with Richard Stallman about this?”
“Linus: rms asked me if I minded the name before starting to use it, and I said “go ahead”. I didn’t think it would explode into the large discussion it resulted in, and I also thought that rms would only use it for the specific release of Linux that the FSF was working on rather than “every” Linux system. I never felt that the naming issue was all that important, but I was obviously wrong judging by how many people felt very strongly about it. So these days I just tell people to call it just plain “Linux” and nothing more.”
(`http://kde.sw.com.sg/food/linus.html`)

Free versus Open Source

Richard Stallman also draws a sharp distinction between Free and Open Source software:
"The fundamental difference between the two movements is in their values, their ways of looking at the world. For the Open Source movement, the issue of whether software should be open source is a practical question, not an ethical one. As one person put it, "Open source is a development methodology; free software is a social movement." For the Open Source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and free software is the solution."
More can be read at `http://www.gnu.org/philosophy/free-software-for-freedom.html`

Debian, Bruce Perens and the DFSG

One major GNU / Linux distribution, Debian, was for a time headed up by Bruce Perens, who worked at Pixar at the time (hence the Debian release names). He headed up the initiative to define free software for Debian: The Debian Free Software Guidelines, part of the Debian Social Contract. Essentially these are a more explicit statement of the FSF rules, and they became the Open Source Software Definition (`http://www.opensource.org/docs/osd`). Two somewhat polarised camps now exist: free software, and open source software, with subtly different goals and motivations, using different terminologies for the same concepts.

Free vs Open Source

- Free software advocates (e.g. Stallman), generally stress ethical issues, and want freedom to the the main, clear issue.
- Open Source advocates (e.g. Torvalds, Perens), generally stress the practical benefits of such software.
- Most people confuse free software with free of charge, and some mischeviously use the phrase open source when material is not free (as in freedom). Consequently, terms like FOSS (free and open source software) are used to be inclusive.

Why not use more FOSS?

It seems obvious that the public sector should be in favour of FOSS, the software is usually free of charge, and you have a lot of rights over it to customize it to need, and as the software is normally based in open standards, vendor lock in is avoided.

The OGC guidelines require open source solutions to be considered on a value for many basis alongside proprietary ones.

Despite this, uptake of FOSS by the sector is very limited, and there are common reasons given for this. So it's useful to explore and consider these.

Why not use more FOSS?

Here are some viewpoints that are typical in the sector, some are more valid that others.

Perceived problems with uptake of FOSS

- Quality "FOSS is flakey";
- Support "We need support contracts";
- Interoperability "We need to use what the real world uses";
- Gaps "There is no FOSS program to do X";
- Special Licenses "But we get all this at a huge discount anyway";
- Budgets "If we spend less, we become less important";
- Transparency and Accountability "Pardon?".

Why not use more FOSS?

The Quality Myth

FOSS products are seen as lacking in QA measures, flaky and unreliable. Of course, this is as true in the proprietary sphere, but there you simply can't do anything if the company goes out of business, loses interest etc..

This should be looked at on a case by case basis. Much of the corporate, educational and public sector infrastructure of the internet has historically run with large elements of rock solid reliable FOSS. For example the Apache web server.

Many GNU/Linux distributions as well as other major FOSS projects like Mozilla (Firefox) have elaborate QA and excellent stability.

Why not use more FOSS?

The Support Myth

Perhaps the greatest fallacy. Support networks for FOSS products are often extremely rich, and superior to those of proprietary products. For example compare Debian's "for free" support, with similar features available for MS Windows.

Also support can be paid for, from any number of different sources, unlike in the proprietary model where support is often only available from the vendor.

In other words, there is a free market to provide support and maintenance and you are quite free to fix issues in house if you choose.

There is also a belief you can sue proprietary vendors more easily in case of problems. Check the fine print, you usually cannot.

Why not use more FOSS?

The Interoperability Myth

It is often argued that compatibility with other software is a major issue. There are exceptions, but generally proprietary software is in fact the natural enemy of open standards, with vendors needing to have distinctiveness in formats and other things built into the business model. Therefore, they tend to produce interoperability within a sphere of products from the same company. This does not provide true choice.

By contrast, "open source" implies open standards, since anyone can examine the code to find out how things are done and stored. This makes it straightforward to build interoperable products.

Many FOSS solutions are based around the concept of modules that do a given job well, or in other words, there are a number of interoperable choices for each piece of functionality you want.

Why not use more FOSS?

Gaps

There are a few areas, but not many, where FOSS solutions don't exist. It's not the purpose of this talk to be unrealistic about that, but to suggest that all other things being equal the FOSS solution is the obvious choice.

Nevertheless, where the solution is in some cases to write the custom software to do the job, we should consider making the product FOSS.

Why not use more FOSS?

Budgets

It’s an uncomfortable fact to face, but in large organisations with departmental divisions, it’s not always in the interests of a given department to save money on procurement. This will be a hidden motive and is therefore hard to counter. It’s important to appropriately reward savings, and allow for transfers of how money is spent so that other resources can be obtained instead.

Perversely, it is often claimed that FOSS does not produce cost savings, since it requires training costs. That’s true of all software, but experience shows there is no greater need for FOSS training than any other.

Why not use more FOSS?

Transparency and Accountability

This seems to go over the heads of many, but a basic issue is having software that implements closed standards with public data. Who owns the data? How will you access it in 20 years time? What is the software actually doing with it?

When you commission software do you own any rights? The copyright to the code? If not, does the license mean only the original authors can change it for you?

These are concerns many simply don't have, but they should!

Creating FOSS

So why should we create FOSS in the public sector?

A better place to start would be why not? Frequently we need to implement what would normally be custom software, a special in-house solution to a problem. Are there any pros and cons to making this free software?

Creating FOSS

Problems in making custom software FOSS

- Security; a common perception is that making software open damages its security. This approach, security through obscurity is generally devalued nowadays in any case;
- Extra Effort; a project started in an open fashion from its outset should require no extra effort to have as an open source venture.

... continued

Creating FOSS

Benefits to making custom software FOSS

- Gift Culture; giving software away and making it publicly available normally acts as a powerful incentive to improve quality;
- Open in a box; even the impact of visibility of a project within its parent organisation cannot be overlooked. Other developers can provide good ideas, or learn them. A "joined up" philosophy is more likely.
- Prestige; if you are going to develop the software, and not exploit it with a proprietary license, then why not show off your good practice;

... continued

Creating FOSS

Benefits to making custom software FOSS

- Dissemination; thousands of lines of code can be more valuable than journal articles;
- Low risk business model; support can be provided on a consultancy basis, with no obligation.
- Gain expertise from others; encourage other organisations with the same problems to improve your software at little or no man power costs to you;
- Distribute expertise; if your core team leave your control, you still have expertise in the code existing elsewhere (sustainability).

In short, FOSS, is an excellent exploitation model.

Creating FOSS

But perhaps the greatest benefit to an organisation for creating FOSS is in embracing the philosophy itself.

Indeed, the philosophy can be extended far beyond software.

Can we learn from the FOSS philosophy?

Eric S. Raymond characterised FOSS development as divided into two methodologies.

The Cathedral, where development took place behind closed doors, and the finished product was released to the general world.

The Bazaar, where the whole development takes place in the public view, and was open to the community from the outset.

The Bazaar method is considered superior and has become the de-facto standard within FOSS. It is, for example, the method employed by the Linux kernel.

Linux

Linux Kernel Development report, December 2010

“Since 2005, over 6100 individual developers from over 600 different companies have contributed to the kernel. The Linux kernel, thus, has become a common resource developed on a massive scale by companies which are fierce competitors in other areas. ”

Linux

Linux Kernel Development report, December 2010

“The numbers in this edition of the paper reflect the natural development cycle of an operating system that had major pieces added/changed in the previous year. Of course the Linux kernel community is still hard at work and growing. In fact, there have been 1.5 million lines of code added to the kernel since the 2009 update. Since the last paper, additions and changes translate to an amazing 9,058 lines added, 4,495 lines removed, and 1,978 lines changed every day – weekends and holidays included. ”

Can we learn from the FOSS philosophy?

So what's good about it?

The Bazaar method has some important strengths.

- development is completely transparent to all stakeholders;
- there is therefore a higher level of accountability;
- it is harder for problems to be hidden and emerge later;
- ideas can permeate in and out of the project from a wider community.

Can we learn from the FOSS philosophy?

Increasing Openness

These ideas have influenced other initiatives:

- Wikipedia (compared with h2g2);
- Diaspora (compared with Facebook);
- Not the effect of opening communication upon government;
- Creative Commons;
- RepRap (self replicating printer);
- OpenCore;

Richard Branson, in January 2008, announced that the Virgin Galactic technical development would follow the open source philosophy.

Relationship between copyright and Licenses

- Copyright is automatic, and may attach to the author or the company;
- To be useful, it must be enforceable;
- Licenses grant rights; usually to allow use and sometimes distribution;
- Copyright holders can change the licensing, or offer dual or more licenses;
- Sometimes copyright has to be assigned away.

Licenses should be explicit

- Failure to be explicit about copyright can lead to problems;
  - It makes it difficult or impossible to change a license;
- Failure to identify an explicit license can lead to problems;
  - It may be illegal to use the code despite the author's intention;
  - It can make it difficult to incorporate code into a project.

Reminder about the GPL

Currently, the most recent version of the GPL is v3, but v2 is still widely used. It is one of the licences used for free software, though there are many others. See `http://www.gnu.org/licenses/gpl-faq.html`.

- Not all GPL programs are GNU;
- Users are free to modify the GPL code as they wish;
- The licence ensures that any improved, or modified version of the code must also be released under the GPL (it cannot be closed);
- It is permissible to sell GPL programs;
- However, anyone with the software has the right to give it away.

The Free Software Foundation - Other Licenses

- There is also the Lesser GPL or LGPL, formerly known as the library GPL. In a nut shell, it offers similar features to the GPL itself, except that code built on top of such software does not “catch” the GPL. It lacks the viral nature of the GPL.
- This license is used strategically where no benefit is obtained from using the GPL proper, for example, when there is an existing alternative to the software in the proprietary sphere that will otherwise be used.

The Free Software Foundation - Other Licenses

- Note, it is perfectly legitimate to build custom software on top of GPL code which is not distributed. It is not free, but it does not break the terms of the license. That has important consequences for web applications. An alternative license that could be used for such an application of the Affero GPL.

DRM and GPL v3

The TiVo video system was built around linux, and was used to enforce (via a subscription fee) access to a service. This (and similar practices) have become known as DRM - Digital Rights Management. The FSF and many others call this Digital Restrictions Management since in fact it is a system to deprive you of your rights, particularly freedom 0 - the right to run software for any purpose.
As part of the drafting of version 3 of the GPL `http://gplv3.fsf.org/draft` some restrictions on DRM were added, to help prevent free software being used to restrict freedom of other kinds.
This has also been divisive.

DRM and GPL v3

3. Digital Restrictions Management.
As a free software license, this License intrinsically disfavors technical attempts to restrict users' freedom to copy, modify, and share copyrighted works. Each of its provisions shall be interpreted in light of this specific declaration of the licensor's intent. Regardless of any other provision of this License, no permission is given to distribute covered works that illegally invade users' privacy, nor for modes of distribution that deny users that run covered works the full exercise of the legal rights granted by this License.
No covered work constitutes part of an effective technological protection measure: that is to say, distribution of a covered work as part of a system to generate or access certain data constitutes general permission at least for development, distribution and use, under this License, of other software capable of accessing the same data.

Linus Torvalds on DRM and GPL v3

“The difference? The hardware may only run signed kernels. The fact that the hardware is closed is a _hardware_ license issue. Not a software license issue. I’d suggest you take it up with your hardware vendor, and quite possibly just decide to not buy the hardware. Vote with your feet. Join the OpenCores groups. Make your own FPGA’s.
[..]
Notice how the current GPLv3 draft pretty clearly says that Red Hat would have to distribute their private keys so that anybody sign their own versions of the modules they recompile, in order to re-create their own versions of the signed binaries that Red Hat creates. That’s INSANE.”

The FSF on Linux, DRM and GPL v3

Richard Fontana, a representative of the Free Software Foundation (FSF) stated that GPL 3.0 is intended to stop developers using GPL code to build DRM products, services or protected content. That means not allowing developers to build code that requires a DRM key to be unlocked and certainly not allowing developers to prosecute others for unlocking their code without the use of specially provided DRM keys - or, in other words, "hacking".

"Linus Torvalds has misread it... We require disclosure of the codes if it's necessary to make the software run."

`http://www.theregister.co.uk/2006/02/15/gpl_drm_license/`

Why is DRM a potential problem?

DRM has many potential uses, but consider just one. DRM can be used to ensure that the hardware of a computer will only run crypographically signed software. This could in principal, in some situations be a good thing (e.g. digital voting machines, if you think they are a good idea anyway). However, it could easily be used to design PCs that are locked into using only one version of one operating system. This can deprive you of freedom 1, the right to study and adapt to your needs.

Software Patents?

Software patents pose another potential danger, particularly to free software. At present they can be obtained in the USA but not within Europe. What are the issues?

- patents protect big companies, and much less small companies and individuals;
  since big companies play patents off against each other
- free and open source software is by definition available for inspection, while proprietary software is not; making it easier to find, or invent, violations in the former;
- patents in software don't increase innovation, they kill it.

Software Patents?

Microsoft recently sued TomTom for infringement of a number of patents. It's worth noting that TomTom runs on top of a Linux kernel.
Microsoft's operating systems do not support any filing systems but their own, a luxury one can easily have in a monopoly position. All of these filing systems have patent entanglements, and arguably, much more advanced alternatives exist. However, in order to support the Microsoft machines, all memory sticks or differents brands and types all have to support MS filing systems.
As a result, to allow support on other OSs, support for patent entangled filing systems needs to be added, and this has allowed this law suit.

Other Significant Licenses

- The other significant family of licenses is the BSD license. Historically there have been several versions, but a recent one omits advertising clauses.
- The BSD license essentially allows third parties to do whatever they like with the code. It can be used in free software, proprietary software, and changed versions need not be distributed.
- For example, it is believed the original Microsoft TCP/IP stack was built upon BSD code.

Business Models

- The proprietary business model is based around code secrecy, and the sale of executable binaries derived from it, so a product is being sold.
- FOSS models are based around code expertise, and therefore a service is being sold. This is usually in the form of consultancy, service level agreements, hosting and so on.
- FOSS models tend to be symbiotic, since code contributed by one project (e.g. Red Hat) becomes available to others, even compitetors in some cases.
- A popular business model is using FOSS for the creation of embedded devices.

Why should we create it?

Like many universities, Ulster has produced a number of custom applications, which are not distributed under license. Two examples were:

- the PDSystem A system designed to record the outcomes of the PDP process, designed initially at a corporate level.
- OPUS A system to manage all aspects of work based placement and its administration, designed initially at a school level.

Over a number of years, as academics do, we went out on conference to discuss how we were improving practice with this software.

The recurring question was “can we have the software”.

The perception within the academic community was that they were sector leading products withni the UK and beyond.

There was interest in having it even if it was sold under a proprietary license, but we were not at that time able to answer, this was a decision that needed to be taken at institutional level.

The development team was keen to deliver the products as free software because

- we use free software to develop the applications, and build them on top of free software;
- we hoped a community of usage and development would emerge;
- we wanted to minimise the risk from being overtaken by a potentially inferior proprietary product which might become industry standard;
- we didn't want to concentrate on a full blown support operation without additional staff.

The University saw it as

- a way to demonstrate and improve its esteem nationally and beyond in these areas;
- a low risk, agile method to exploit a product that otherwise, probably would have no exploitation outside the University;
- and therefore able to produce modest consultancy, or support service income, which could be grown if need be.

The decision was taken to make the two applications free software, under the GNU General Public License v2, in September 2006.

Implementation - The Plan

Releasing code under license did not by any means happen overnight, there was some work to be done first:

- most importantly, an infrastructure had to be put in place which allowed the hosting of code, downloads, related resources, as well bug and other trackers - essentially the requirements for the bazaar style of development.
- a comprehensive security audit of the code was necessary before release, but just for vulnerabilities in the code, but for other materials like system passwords that existed in our original repositories.

These processes, occuring as they did alongside other duties took some time to complete.

Implementation - The Action

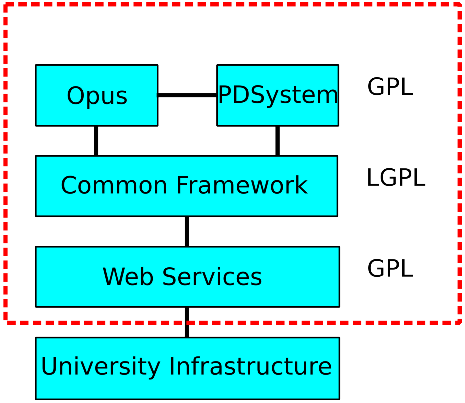
We could have hosted our code on sourceforge or any other similar entity, but we took the decision to install our own environment for hosting development.

1. it serves as a flag in the sand, to hopefully encourage more such projects within the University of Ulster and beyond;
2. it gave us total control of our first attempt at this process;
3. it made the establishment of a brand identity rather simpler;
4. we could host related activities on the same server.

So we installed `http://foss.ulster.ac.uk`, entirely with free software, notably Debian and Savane.

Implementation - The Action

- We created brand new source repositories, and moved code from our closed ones to the new ones as it was audited;
- We created mailing lists for users and developers;
- We established controls that would allow us to track the contributions of any new contributors, so it would be easy to add new team members.
- We released applications finally in February 2007.



The main packages, implemented on the LAMP stack.

Results

Benefits

- income for customized versions (TEPNI for the PDSystem);
- greatly improved, more streamlined, facilities for our own development;
- internal customers have benefited from improved code quality, and all the facilities we have implemented for external customers;
- national endorsement of our products (e.g. OPUS endorsed by ASET);
- very considerable interest from other insitutions, literally around the world, with installations in Australia and Hong Kong.

Results

Benefits

- greater impetus for improved software quality (gift culture);
- material on foss is directly useful for teaching, and allows students to build projects from scratch or on top of other technologies.

What is next?

- dual stranded consultancy; training and technical;
- continued work on improvement of existing applications by staff, project and placement students;
- improved documentation which can be produced by non technical practitioners;
- other existing products to become free;
- application for funding to undertake similar work built on the same technologies;

What you need

There are some components to being able to support this kind of endeavour

- Revision Control Systems (network enabled);
- Bug and Issue trackers (preferably open);
- Mailing lists and other fora (preferably open);

There is a learning curve for such tools, but it brings internal, and possibly external collaborative benefits. Packages like GForge provide all of this.

FOSS activity in Northern Ireland and Beyond

- open-ni network, to represent all open source interests in Northern Ireland.
  `http://www.open-ni.org`.
- foss@ulster, to host development at the University of Ulster.
  `http://foss.ulster.ac.uk`.
- Open Source Solution Centre at the SRC
  `http://www.src.ac.uk`.
- Open Source Academy Case Studies
  `http://www.opensourceacademy.gov.uk/solutions/casestudies`

Summary

1. It's a first rate solution, not second rate
   - Google uses it and produces it extensively, so does Sun, HP and many others.
   - SMEs use it to allow agile startups.
2. Consumption of FOSS
   - OGC guidelines require FOSS to be considered.
   - It provides open standards and avoids vendor lock-in.
   - FOSS procurement in the public sector is limited by many misconceptions.
   - Most of these are easily debunked.
3. Creation of FOSS
   - The public sector "may as well" create FOSS. Why not?
   - There are plenty of benefits in doing so.
4. Borrowing FOSS methodologies
   - Increasingly seen as an agile, cheap, accountable and sustainable way to conduct business.