

The UK Advisory Service on Free and Open Source Software

Sebastian Rahtz
September 24th 2003

Summary

- Why an advisory service?
- How was it created and funded?
- What have we learnt so far?
- What advice do we have for U.S. universities?
- What is the global picture?
- What next for OSS Watch?
- Some of the F/OSS issues that interest me

Background

I am:

- Information Manager for Oxford University Computing Services
- Member of the Board and Technical Council of the Text Encoding Initiative
- A long-time developer and author in the TeX world
- An XML bigot
- A user of free software since the late 80s

An open source advisory service for UK institutions

OSS Watch:

- funded by JISC for two years from 1st July 2003
- part of JISC's *Information Environment*
- working in partnership with other JISC services, eg CETIS and the Mirror Service
- serving FE and HE equally (not just research universities)
- run by Oxford University Computing Services with 2 × 50% and 1 × 25% staff (total £70,000 a year)
- visible at <http://www.oss-watch.ac.uk>

Glossing for the unfamiliar

JISC Joint Information Systems Committee.

Coordinates educational IT structures (including JANET). Directly funded by the state at the same level as research councils

JANET The Joint Academic NETwork, run by UKERNA for JISC

FE and HE Further Education and Higher Education. Most post-18-year old state-funded teaching and research institutions

Mirror service JISC-funded repository for local copies of software

CETIS JISC-funded service for studying standards in educational software

UKOLN JISC-funded service for studying the web, especially standards and metadata

JISC services

JISC funds several dozen national services, each employing between 1 and 20 people each, to meet informational or development aims. It wants to:

- Get the best value for money from JANET
- Promote good practice in networking standards
- Persuade education to use the vast stores of information available
- Fund development of software to fill gaps

JISC does not directly fund any infrastructure for computing beyond the network. Central computing provision lapsed in the late 80s.

Remember that the UK has no private universities of any stature.

Creating OSS Watch

The Open Source service is a small pilot exercise:

- Call for proposals issued in March 2003
- Service had to start on July 1st 2003, for 2-3 years
- Budget of £70,000 a year allocated
- Open for competitive bidding by any educational institution
- Service awarded on the basis of 8-page document and interview
- Five groups made it to interview stage

Why our bid was successful

- Oxford is a world-class university in teaching and research
- The services is located in a Research Technologies Service of the central computing services
- We have successful track record in the management of JISC-funded services
- We are co-located with the Oxford e-Science Centre (Grid science) and the Humbul (cataloguing humanities information services)
- Oxford has academic expertise in open source (sociologically and legally) in the Oxford Internet Institute
- We *do* open source

Our governance

We are supervised by a JISC minder, and an Advisory Committee with representatives from

1. JISC
2. UK universities
3. UK Further Education colleges
4. The UK Government (The Office of the e-Envoy)
5. Academic research
6. The commercial sector, including software vendors

What is the remit of OSS Watch?

To give assistance to the community and advise JISC on:

1. What to do with the results of publicly-funded software projects
2. To determine the exposure of the UK academic community to any problems with open source
3. To provide information to institutions considering putting open source into their information strategies

What OSS Watch does

- Offer a neutral and practical web site
 - Run at least two open meetings a year
 - Run two focus groups year, and write analyses
 - Engage in understanding institutional processes
 - Advise IT managers, project developers, and users
 - Give advice on open source at any UK/FE forum
- and make all its material available under the GNU Free Documentation License

What OSS Watch does not do

- Try to *persuade* people to adopt open source
- Run a software repository
- Help people with their Open Office problems
- Compete with *freshmeat* or *slashdot*
- Provide definitive legal advice
- Be a forum for hairy sandal-wearing geeks

OSS Watch's first few months

1. a web site framework
(<http://www.oss-watch.ac.uk>)



2. a logo
3. an online survey of users
4. presentations at meetings
5. preparation for scoping study
6. preparation for first OSS Watch conference

Staffing

Who's working on OSS Watch?

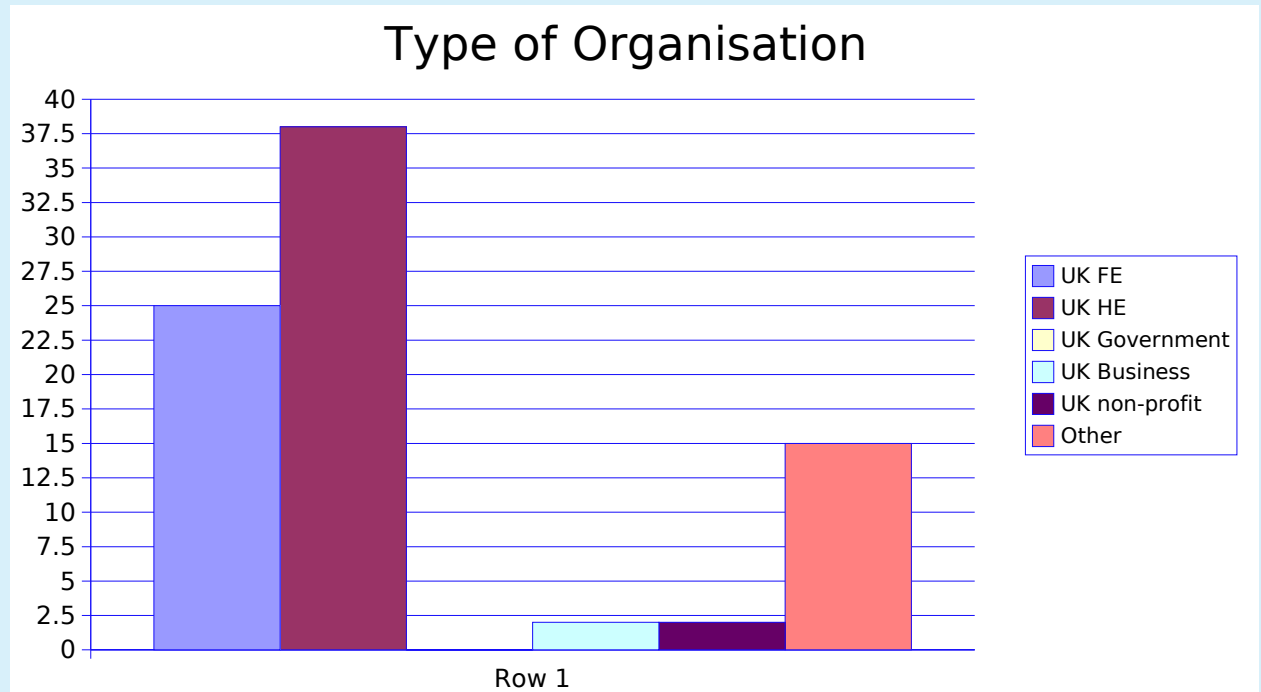
- Sebastian Rahtz (Management, 0.25%)
- Randy Metcalfe (Communications, 0.5%)
- Rowan Wilson (Development and Legal issues, 0.5%)
- David Tannenbaum (Scoping survey, September/October)
- Mike Fraser (Software survey)

First survey

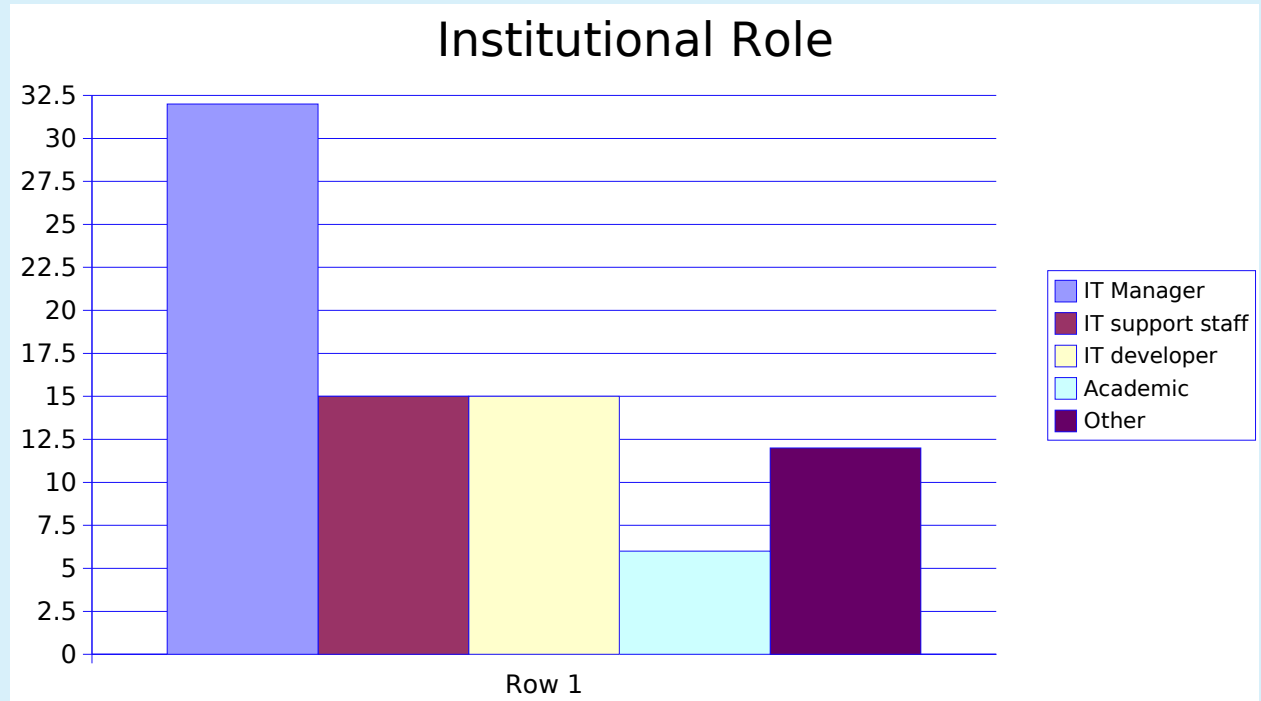
Taken by about 90 people visiting the web site over the summer. It tried to establish:

- ➡ Who was visiting us
- ➡ What they did now about F/OSS
- ➡ What areas they were looking for help with
- ➡ What they expected OSS Watch to do

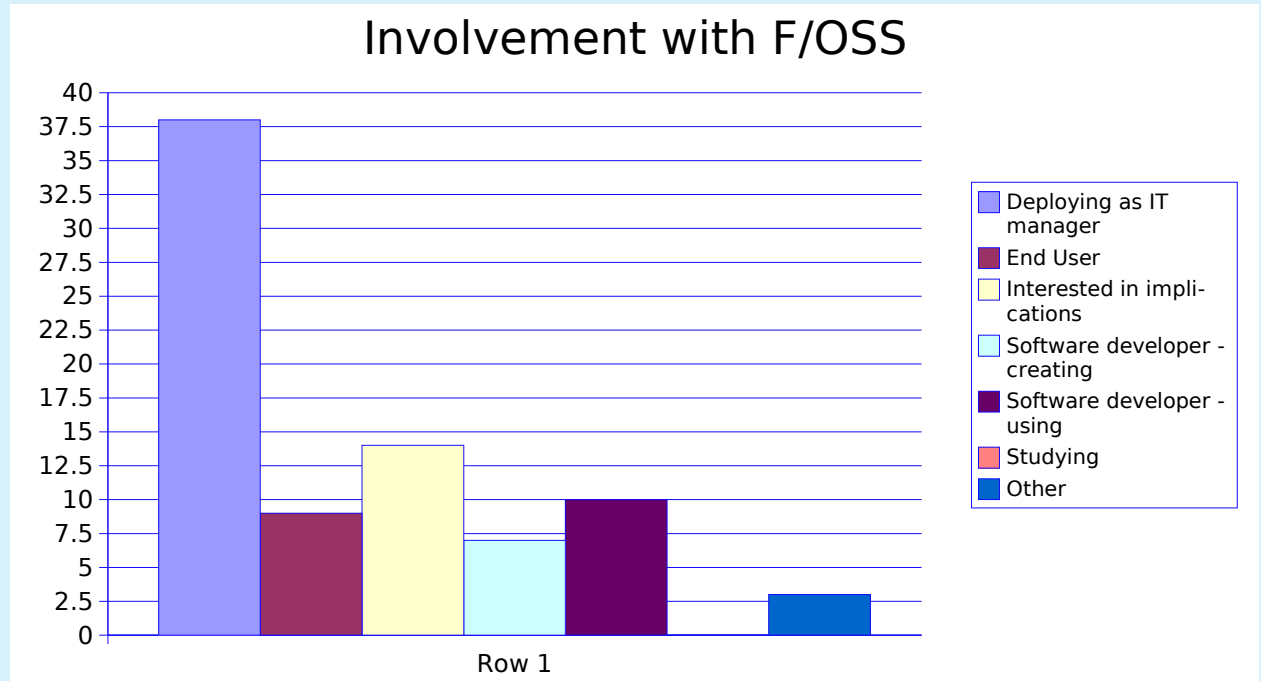
Institutional type



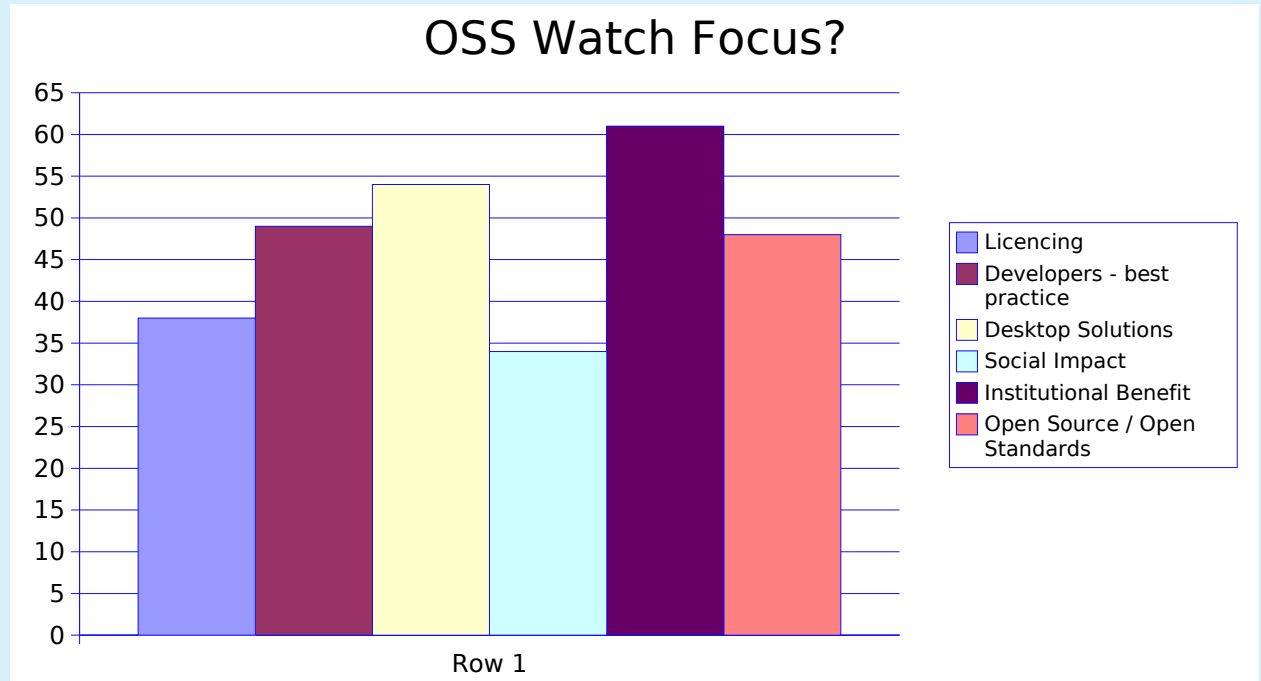
Institutional roles



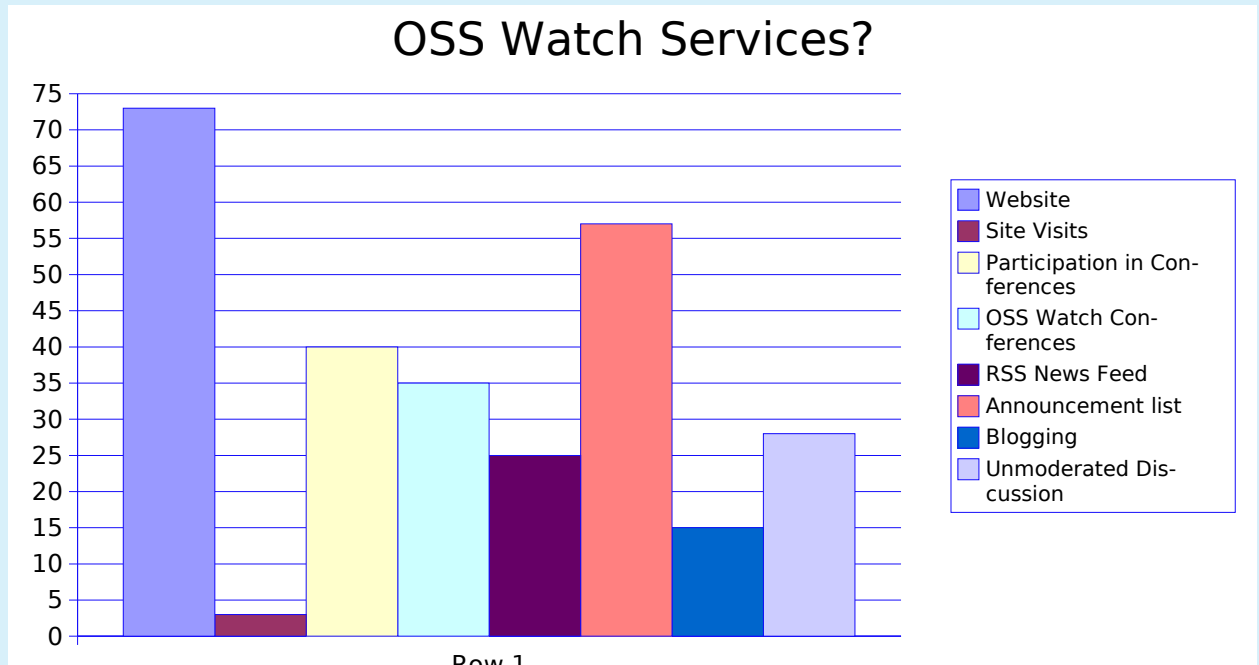
Involvement with F/OSS



OSS Watch focuses



What services are wanted?



Comments from visitors

- ➡ There seems to be a fear among many FE colleges that OSS might be hard to support without access to expensive skill-sets. IT Support skills in things like Linux must be made more mainstream before adoption increases. Not quite sure what to suggest to help with this though. Cheaper training perhaps?
- ➡ Warts and all case studies of institutes that have deployed open source software. News from people with actual experience is the best judge of a tools success or otherwise.
- ➡ I am particularly interested in reviews and comparisons of products.
- ➡ To suggest packages suitable for a large fe college's everyday needs, (services, desktop os, office, graphics, media authoring and players, **or an advisory service on Free and Open Source Software (emphasising the benefits for us all and dispelling fears and**

More comments

- ➡ Interface with mainstream software, particularly Microsoft Office
- ➡ Look into tools relating to the creation and searching of XML data.
- ➡ As an Open Source Software house, we are interested in dispelling some of the misconceptions around OSS
- ➡ Please don't permit OSS to become synonymous with Linux. We deploy OpenBSD for network services (HTTP, NFS, NTP, DHCP, syslog), and OSS applications (GRASS, R, PostgreSQL, TeX) on Windows and Solaris. Our interest in Linux itself is zero.
- ➡ FE in our region have little or no UNIX type skills to develop the potential of open source software. Some instructional guides / training courses in open source operation systems would be a good starting point with FE.

Scoping Study

A more detailed study of HE and FE institutions in order to help establish the needs of the stakeholder community. Looking at:

- needs of key stakeholders
- deployment of F/OSS at HE/FE institutions
- software development using F/OSS paradigms
- end-users view of F/OSS applications
- HE/FE goals for deploying, investigating and developing F/OSS
- interest in the longer-term HE/FE participation in the F/OSS community

Survey questions

- Personal information (role, interests, etc)
- Institutional information (size, budget, etc)
- Place of F/OSS in organisation's IT strategy
- Level of skill in F/OSS
- Degree of deployment of F/OSS
- Types of deployment
- Specific software used
- Plans for deployment
- Software development plans/practices
- Concerns about F/OSS

OSS Watch Conference

The first OSS Watch conference will be held on 11 December at the University of Oxford. Three themes:

- Deployment: what does it mean to have F/OSS in an institutional IT strategy? Can we learn lessons from other places?
- Development: what is happening in UK/HE with F/OSS? what does it mean to be an open source project?
- What can OSS Watch do to help?

Collaboration

With other JISC services in general, and:

- Bodington VLE project: how to be an open source project
- Subject Portals Project: how to be an open source project
- JISC 'Share Alike': non-software-licenses
- Advisory and Support Services Liason Team: talking to FE
- Oxford Internet Institute: research in open source history etc
- TheOpenCD: delivering a demonstrator open source CD for Windows

Upcoming projects for OSS Watch

- Develop standard talks for user groups
- Catalogue UK open source projects
- Talk to successful projects and ask them how they did it
- Write guidelines for publicly-funded software projects
- Work with **The Open CD**: hand out desktop software and study takeup
- Focus groups
- Liaising with e-Government initiatives

Drinking our own Kool Aid

We decided early on at OSS watch to actually *do* free/open source, so we

- Write our documents in XML against the Text Encoding Initiative DTD
- Author using Emacs, Open Office etc
- Deliver them on the web using Apache and the AxKit XML delivery system
- Prepare printed material using TeX
- Deliver presentations using Linux
- Store data in Postgres databases
- License documents using the GNU Free Documentation License

Is Oxford typical?

We **depend** on open source:

- Operating systems: mail server runs under Linux
- Networking: Apache web servers and numerous network systems (DNS, Exim, LDAP etc)
- Software development: the majority of web cgi applications developed at OUCS are in Perl
- Our internal helpdesk system is web-based, open-source, and written in Perl
- The open VLE system (Bodington) is our flagship project for 2003/2004
- The e-Science GRID depends on open standards and software
- We are working on a portal system using uPortal

You will recognize some of these.

The UK Advisory Service on Free and Open Source Software

What is the message to the US?

The US is much like the UK. We need **data**, and **test cases**. Do we know:

- 👉 what the cost of open source to the community is
- 👉 how to write an IT policy which supports open source
- 👉 where IPR, patents, copyright etc are going
- 👉 whether positive discrimination can work
- 👉 how to protect work done with public money
- 👉 what the ideal licensing model is

Some test cases of GPL going to court would be good!

Is anyone in the world on the ball?

Some famous cases are

- ➡ Munich, Germany, where the city council has mandated Linux
- ➡ The UK government guidelines promote open source
- ➡ India in general, and Kerala in particular, use open source a lot in schools
- ➡ The Nordic countries have an advisory service for consumers and small businesses
- ➡ The Peruvian government has had its wars with Microsoft
- ➡ The Brazilian government has endorsed open source

Motives

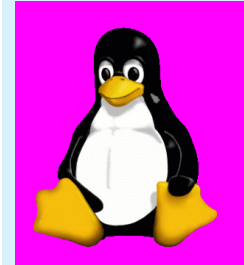
1. Worry about Microsoft in particular, and the big software vendors in general, for restrictive and monopolistic practices
 2. A need to get specialist software written
 3. Saving money in schools
 4. Promoting open standards
 5. Protecting institutional investment
 6. Wanting quality, control and security
 7. Providing apprentice training for programmers
- But which of these will stand the test of time?

Legal issues

1. Who owns the IPR? To what extent do universities control the lives of their employees?
2. Which basic license decision?
 - (a) GPL: control the future
 - (b) Apache: give it to a neutral third party
 - (c) MIT: let a thousand flowers bloom
3. Who is the best person to protect rights?

No, not SCO vs IBM, and not patents on software

The gallery



License Q and A

Are you authorised? Check whether you own the copyright. If you don't, you cannot add an OSS license

License Q and A

Are you authorised? Check whether you own the copyright. If you don't, you cannot add an OSS license

Do you care what happens to your work? If you just want to make sure your stuff remains freely available for ever, choose an MIT-type license

License Q and A

Are you authorised? Check whether you own the copyright. If you don't, you cannot add an OSS license

Do you care what happens to your work? If you just want to make sure your stuff remains freely available for ever, choose an MIT-type license

Are derivatives to be free as well? Insist on the GPL

License Q and A

Are you authorised? Check whether you own the copyright. If you don't, you cannot add an OSS license

Do you care what happens to your work? If you just want to make sure your stuff remains freely available for ever, choose an MIT-type license

Are derivatives to be free as well? Insist on the GPL

Do you simply want to make your program free? (As in beer). Don't feel you have to choose an open source license

License Q and A

Are you authorised? Check whether you own the copyright. If you don't, you cannot add an OSS license

Do you care what happens to your work? If you just want to make sure your stuff remains freely available for ever, choose an MIT-type license

Are derivatives to be free as well? Insist on the GPL

Do you simply want to make your program free? (As in beer). Don't feel you have to choose an open source license

What is Free/Open Source Software?

Software for which:

- the source code is available to the end-user;
- the source code can be modified by the end-user;
- the licensing conditions are 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.

‘Open Source is a development methodology; Free Software is a social movement.’

Virtues of free and open source software?

- 👉 has no secrets: the innards are available for anyone to inspect

Virtues of free and open source software?

- ➡ has no secrets: the innards are available for anyone to inspect
- ➡ is not privately controlled: so likely to promote open rather than proprietary formats

Virtues of free and open source software?

- ➡ has no secrets: the innards are available for anyone to inspect
- ➡ is not privately controlled: so likely to promote open rather than proprietary formats
- ➡ is typically maintained by communities rather than corporations: so bug fixes and enhancement are often frequent and free

Virtues of free and open source software?

- ☞ has no secrets: the innards are available for anyone to inspect
- ☞ is not privately controlled: so likely to promote open rather than proprietary formats
- ☞ is typically maintained by communities rather than corporations: so bug fixes and enhancement are often frequent and free
- ☞ is usually distributed free of charge (developers make their money from support, training, and specialist add-ons; not marketing)

Virtues of free and open source software?

- ➡ has no secrets: the innards are available for anyone to inspect
- ➡ is not privately controlled: so likely to promote open rather than proprietary formats
- ➡ is typically maintained by communities rather than corporations: so bug fixes and enhancement are often frequent and free
- ➡ is usually distributed free of charge (developers make their money from support, training, and specialist add-ons; not marketing)

Clearing up misunderstandings

- ➡ **Free** software uses the 'free' from 'freedom', not the one from 'free beer'. Open source software may or may not cost money

Clearing up misunderstandings

- ➡ **Free** software uses the ‘free’ from ‘freedom’, not the one from ‘free beer’. Open source software may or may not cost money
- ➡ The cost of **ownership** often bears little relation to the cost of acquiring a piece of software

Clearing up misunderstandings

- ➡ **Free** software uses the ‘free’ from ‘freedom’, not the one from ‘free beer’. Open source software may or may not cost money
- ➡ The cost of **ownership** often bears little relation to the cost of acquiring a piece of software
- ➡ ‘**Public domain**’ is something different. Open source software has a copyright holder and conditions of legal use

Clearing up misunderstandings

- ➡ **Free** software uses the ‘free’ from ‘freedom’, not the one from ‘free beer’. Open source software may or may not cost money
- ➡ The cost of **ownership** often bears little relation to the cost of acquiring a piece of software
- ➡ ‘**Public domain**’ is something different. Open source software has a copyright holder and conditions of legal use
- ➡ Open source software does not mandate **exclusivity**. You can use open source programs under Windows

Clearing up misunderstandings

- ➡ **Free** software uses the ‘free’ from ‘freedom’, not the one from ‘free beer’. Open source software may or may not cost money
- ➡ The cost of **ownership** often bears little relation to the cost of acquiring a piece of software
- ➡ ‘**Public domain**’ is something different. Open source software has a copyright holder and conditions of legal use
- ➡ Open source software does not mandate **exclusivity**. You can use open source programs under Windows
- ➡ People do not choose software solely on the basis of open source. **Interoperability and open standards** for data are equally important

Clearing up misunderstandings

- ➡ **Free** software uses the ‘free’ from ‘freedom’, not the one from ‘free beer’. Open source software may or may not cost money
- ➡ The cost of **ownership** often bears little relation to the cost of acquiring a piece of software
- ➡ ‘**Public domain**’ is something different. Open source software has a copyright holder and conditions of legal use
- ➡ Open source software does not mandate **exclusivity**. You can use open source programs under Windows
- ➡ People do not choose software solely on the basis of open source. **Interoperability and open standards** for data are equally important

F/OSS groupstyle creation characteristics (1)

Free and open source software tends to have the following positive development characteristics:

1. Programmer commitment, because the programmer is also the user
2. Rapid change, because programmers want to see results
3. Unconstrained specifications, because there is no external client
4. Collective ownership of code
5. Response to change, dictated by (perhaps unexpected) users

F/OSS groupstyle creation characteristics (2)

... and the following negative characteristics:

1. Unrealistic expectations, because there is no client controlling the specification
2. Uneven development rates, because programmers work on what they want
3. Flat managerial tree, because programmers are not working to contract, so no control or direction
4. Changing resources (no guarantee of work hours)
5. Possibly conflicting goals or aspirations
6. It does not have to succeed

a bit like a rock band?

F/OSS groupstyle creation characteristics (3)

But the following do *not* necessarily apply:

1. **Poor quality code:** likely or as unlikely as in traditional programming
2. **Slow development:** depends on who takes an interest and how hard they work
3. **It isn't as polished as commercial software:** much commercial software is riddled with problems
4. **No-one supports it:** most projects have a large group of "hangers on" who do not directly develop, but do help others

Why do people write open source?

Quite a lot is known about programmers who work on free software. They are

- ➡ Young
- ➡ Increasingly working on f/oss for a company
- ➡ Interested in learning
 - ➡ from others
 - ➡ new technology
 - ➡ at a distance

in other words **apprentices** not **hackers**

Traditional software development

Relies on sequential phases of development; it makes several dangerous assumptions:

- 👉 the existence of a set of requirements that is defined beforehand and is maintained unchanged during development;
- 👉 the assumption that all code is designed from scratch, making no allowance for the reuse of existing software components
- 👉 no repeating previous phases of the development without having to repeat the whole sequence of steps for the entirety of the system.
- 👉 development as a linear process.

Iterative and incremental development methods

Incremental development:

- proceeds from an initial subset of the requirements to more and more complete subsets, until the whole system is addressed
- partitions the desired functionality
- supports parallel development

The final product results from the total integration of the partitions.

Lighter methodologies

Over the last 3-4 years, software theorists have tried to deal with issues such as:

- ➡ Software must change more and more quickly these days
- ➡ Software keeps being delivered late
- ➡ The customer is not involved with what she is buying
- ➡ Programmers lack motivation
- ➡ We need new software all the time

The industry needs the equivalent of fast food: predictable quality delivered quickly.

A methodology: Extreme Programming

1. The Planning Game
2. Small Releases
3. System Metaphor (a *story*)
4. Simple Design
5. Continuous Testing
6. Refactoring
7. Pair Programming
8. Collective Code Ownership
9. Continuous Integration
10. 40-Hour Work Week
11. On-site Customer
12. Coding Standards

A movement: Agile

Agile is a conscious revolutionary movement:
<http://agilemanifesto.org/>. It prefers:

Individuals	over	processes and tools
Working software	over	full documentation
Customer collaboration	over	contract negotiation
Responding to change	over	following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile software engineering

Traditional engineering

passive client
minimal changes
process rules
long iterations
static process

Agile engineering

active client
encourage change
developer rules
short iterations
dynamic process

The Agile Manifesto

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The Agile Manifesto (2)

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

What does *open source* add to the mix?

Some open source projects make a virtue of:

- ☞ Space, time and culture separation of team members
- ☞ Semi-professional programmers having fun
- ☞ Publicity and (friendly) competition
- ☞ Co-operating systems (supply of components)
- ☞ Experience with virtual communities
- ☞ Promotion of open standards
- ☞ Allowance of failure

Others are traditional programming teams in big companies.

What does *free software* add to the mix?

- Free as in beer
- Concept of the public good
- Identification of software as unpatentable human heritage
- Anti-establishment ethos
- Missionary conversion of others

Open standards meets free/open source?

Which is better?

➡ Commercial software which uses an XML data format and can be accessed using web service protocols

or

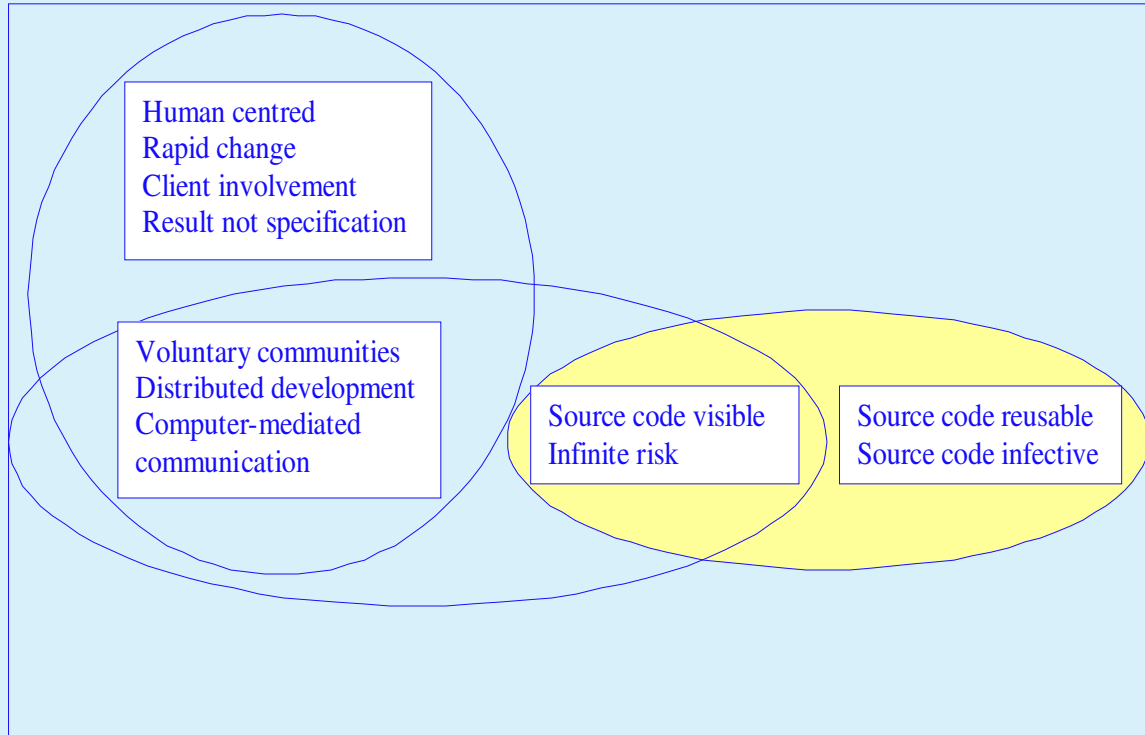
➡ An free/open source program which uses its own binary data format, its own interface, and its own programming language

Open data and open communication between different components of your IT system gives you better interchangeability of components.

Sociological perspectives on F/OSS

- An essential characteristic: *Novel use of IPR to distribute/publish software under public domain-like conditions*
- Almost essential: *Distributed mode of creating (producing) an information-good: software*
- Common but not necessary: *Extensive voluntary participation by communities of skilled and neophyte software developers*
- Likely: *Dependence of the production mode upon the non-proprietary distribution regime*
- True across the board: *Critical role of computer-mediated communications (CMC) for this production system*
- Always desirable: *Self-documenting nature of the process*

Overlapping areas of interest



Further reading

<http://www.gnu.org/philosophy> (Free Software Foundation)

<http://www.opensource.org> (Open Source)

<http://www.debian.org> (Debian Linux)

<http://www.stallman.org> (Richard Stallman)

<http://www.w3c.org> (World Wide Web Consortium)

<http://http://www.govtalk.gov.uk/> (e-GIF)

<http://www.egovos.org> (Center of Open Source & Government)

<http://www.oreilly.com/catalog/cathbazpaper/> (Eric Raymond's *The Cathedral and the Bazaar*)

What shall we do

- ☞ decide on our motives: free, open, or economic
- ☞ work out whether we need to legislate and/or band together
- ☞ find out whether mixed economies will work
- ☞ start writing our own software
- ☞ **or** stop writing our own software