Free and Open Source Software in academia

John R Hudson*†

Part I
History of FOSS

In the beginning

Among the first modern computers outside military applications such as 'Colossus' at Bletchley Park was EDSAC, built at Cambridge University. This became the basis for the first commercial computer, LEO 1 (or L*yons* E*lectronic* O*ffice* 1), developed for the J Lyons and Co. bakery and restaurant chain (Ferry, 2003). Meanwhile at Manchester University, the Manchester Baby had evolved into the Manchester Mark 1 which became the basis for the Ferranti Mark 1. At this time computer software was provided and maintained free with the hardware as it would only run on the hardware for which it was written — a situation common until recently with in-car electronics and mobile 'phones.

Though UK software development was to become largely a closed process, in the USA it remained largely open with much of the development of the Unix system software, first developed in AT&T's labs, taking place in universities. These were required by US legislation relating to bodies in receipt of public funds to make their research publicly available. Initially, they did this by printing the code for the software in research papers but eventually the length of the code became too great to include in papers and they began publishing it separately as computer files. To comply with US legislation, they created permissive licences, that is, ones allowing users to modify and publish their changes to the code, of which the MIT (Massachusetts Institute of Technology) and BSD (Berkeley Systems Department) licences are probably the most well-known.

Because AT&T was virtually a monopoly communications company in the US, it had been barred from selling computing equipment but in 1983 it was broken up and released from this restriction. Once the ban was lifted, AT&T began charging people for Unix. This created problems for the universities whose contributions to Unix were now being sold, in apparent contravention of their legal obligations, and annoyance to many of the academics who had contributed to it. Over the next decade, there were numerous attempts to resolve this impasse including developing different versions of Unix based on the universities' work, of which BSD

*The author would welcome notification of any errors or possible misunderstandings.

†First presented in the Magrath Room of The Queen's College, Oxford at 11 am on Monday, 4 May 2015.

Unix, which is used by Yahoo and Apple, is probably the most well-known. The impasse was finally resolved in 1993 when the US computer company, Novell, bought the rights to Unix from AT&T and declared that it would not charge anyone for using Unix.[1]

TEX, LATEX and BibTEX

A significant example of free software from this era was the TEX typesetting engine written in Pascal by Donald Knuth following receipt in 1977 of the proofs for the second edition of volume 2 of *The art of computer programming* (1981). Whereas the first edition had been set by the traditional hot type process, the second, set using phototypesetting, looked awful.

The TEX engine was essentially complete in 1982 but development continued until 1989. However, passing parameters to TEX required detailed knowledge of Pascal. So Leslie Lamport, an employee of DEC, created a set of macros which allowed a less sophisticated user to use TEX. With the agreement of his employers, Leslie Lamport placed these macros in the public domain and they were taken up and developed, in particular by the American Mathematical Society (AMS) to typeset mathematics. To this Orem Patashnik (1988) contributed BIBTEX, a bibliographic database system, which has since been extended and developed to provide a wide range of styles of referencing. Leslie Lamport subsequently handed over responsibility for LATEX to Frank Mittelbach and since then much of the development of LATEX has been around extending it to support languages other than English.

LATEX became very popular among the scientific, engineering and mathematical communities in the US and continental Europe but is less well known among people from arts backgrounds and in the UK.

Free software

During the same period, programmers had been exchanging free (and not so free) software through locally based 'Bulletin Boards' but, concerned at the increasing restrictions being placed on programmers by the commercialisation of software and fearful that his work might be hijacked by his employer, the Massachusetts Institute of Technology, Richard Stallman, who worked at the Artificial Intelligence Laboratory, resigned his post in 1984 and recruited a number of like-minded programmers to create the Free Software Foundation in 1985 and articulate in the 'GNU Manifesto' the four freedoms:[2]

- *The freedom to run the program, for any purpose.*
- *The freedom to study how the program works, and change it to make it do what you wish.* Access to the source code is a precondition for this.
- *The freedom to redistribute copies so you can help your neighbor.*

[1] In 2011 Novell was taken over by Attachmate, who announced that they would continue Novell's approach to the rights to Unix. In 2014 Attachmate was taken over by the UK company, Micro Focus International, who have not indicated any intention to change these arrangements.

[2] The term is used in imitation of the four freedoms articulated by US President Franklin D. Roosevelt on January 6, 1941

- *The freedom to distribute copies of your modified versions to others.* By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

It is important to say that 'free' in this context does not mean 'without cost' but 'without restrictions on what you can do with it' or 'free as in speech, not as in beer.'

In 1989 he formalised these principles in the GPL (GNU General Public License) which, in its several versions, is used to protect a lot of FOSS. It differs from traditional forms of copyright in asserting both the right of the author to be identified as the author of the software and the rights of the user to use the software in the ways set out in the GNU Manifesto.

Stallman and his colleagues tried, unsuccessfully so far, to develop a free version of Unix but succeeded in writing a lot of useful software, sometimes collectively known as the GNU utilities, which run on most Unix-like systems and contribute to the operation of the Internet.

Open source software

The term 'open source' was not coined until 1998 (Biancuzzi, 2008) and it is a little misleading since, apart from the proprietary software developed since the 1970s by individuals and companies who were not selling any hardware and therefore wanted a fee for their work[3] and 'shareware,' that is, software distributed free but normally with a request to make a donation to the author, most software is 'open source' in the sense that anyone can read the code. The term was coined in 1998 to cover a new way of creating and supporting computer software, initiated in 1988 by Intel. They hired Michael Tiemann to write the software for a new engineering chip and then gave it away with the chip, telling customers who were themselves not sufficiently skilled to use it to contact Michael for help. In association with John Gilmore and David Henkel-Wallace, he built up Cygnus Support to provide support for a variety of free and open source software as well as to develop a number of the GNU utilities.[4]

The key feature of this model is that creating software and supporting it became separate activities with charges only being made for support. It was taken up by other companies, most notably the Swedish database company MySQL[5] which gave away its software but offered support contracts for people needing help to get the most out of it. MySQL estimated that it had a thousand users who had downloaded its software free for every one that paid for support but the feedback from this thousand enabled them to identify problems and develop the software to meet users' needs far more rapidly than if they had relied only on feedback from those who had paid for support.

The expansion of copyright

Meanwhile, as the US economy flagged, US legislators hit on the idea of promoting innovation by extending copyright to provide a greater incentive for innovation. In 1980 it became possible

[3] In 1982, at the end of the long running anti-trust litigation brought against it by the US Department of Justice, IBM began charging for all its software separately from its hardware though some people thought it 'underpriced' it to keep the competition at bay (Glass, 2005). Other hardware companies who had not already done so followed suit.

[4] In 1999 Cygnus Solutions, as it was called by then, merged with RedHat.

[5] In 2008 Sun Microsystems, a long-standing manufacturer of high performance computers, acquired MySQL for $1 billion dollars. Since Oracle acquired Sun Microsystems in 2010, MySQL is now supported by Oracle.

to copyright software in the US while the courts became more favourable to patenting software, a position formalised in law in 1990.

During the 1980s media publishers began to argue that strengthening copyright laws would benefit the US economy and by 1990 they had persuaded the US government to include in trade deals the requirement that other countries would respect US copyright laws. In 1992 breaking copyright law became a criminal, not just a civil, offence in the US and in 1994 the US was able to make respecting its copyright laws part of GATT (General Agreement on Trade and Tariffs, to be succeeded the following year by the World Trade Organization).

In 1998 the media publishers succeeded in persuading Bill Clinton to sign into law the Digital Millennium Copyright Act which imposes penalties for circumventing digital copyright systems and have continued to pursue copyright 'pirates' ever since.[6]

These changes had been justified as part of neo-liberal economic theory about creating a free market but they actually had the effect of severely curtailing the freedom of the individual which is supposed to be at the heart of such theories (Coleman, 2013).

By contrast, FOSS licences do not treat copyright as a commodity which can be given away or sold but as an attribute which never leaves the original author; in 2001 Lawrence Lessig established Creative Commons to provide a range of licences which protect the author's right to be identified as the author of material they have produced while making clear the extent to which the reader may reproduce or distribute the material.

The World Wide Web

In 1984 Tim Berners-Lee (Queens 1973) was given a CERN fellowship in the course of which he developed the idea of a hyperlinked interface for accessing the many documents that an organisation like CERN produces. By 1991 he had been able to get the World Wide Web running on a NeXT computer — which Steve Jobs had designed after he was sacked from Apple in 1985 and which became the reason why was he brought back to Apple in 1996 — and by the end of the year, with the help of supportive colleagues, he had it running on DEC VAX computers.

Over the next two years, several university and commercial projects took up the idea, not least the designers of the Mosaic web browser. In 1993 CERN finally agreed to release the source code and Tim Berners-Lee went on to found the World Wide Web Consortium (or W3C) charitable foundation to manage it.

In fact, few people inside or outside CERN saw any future for the World Wide Web and it was only the determination of a few key players, such as Marc Andreessen, one of the designers of the Mosaic web browser who went on to create the Netscape web browser, that led to its development and CERN's agreement to release the source code (Berners-Lee, 1999).

Linux

With increasing adoption of Unix on larger computers in the 1980s, Andrew Tannenbaum had created a smaller alternative, Minix, to use in teaching students the principles of Unix. In 1991 a computing student at Helsinki University, Linus Torvalds, became so impatient waiting to get onto the university computer that he used Minix as the starting point for building his

[6] Copyright 'piracy' actually accounts for a minute percentage of cybercrime (Anderson et al., 2012).

own operating system which, with the help of the systems administrator at the university, who called it 'Linux,' he put on the Internet. Within 36 hours he had people contacting him and within 18 months they had developed a kernel which worked well enough with the software which the universities and Richard Stallman and his colleagues had developed to make it a viable system for use on PCs using Intel processors.

Among the factors contributing to the success of Linux were its adoption of the GPL (GNU General Public License), the continuing disputes over Unix (see In the beginning on page 1) and the facts that it was free and would run on PCs as opposed to the mainframes needed for most varieties of Unix.

In 1996 Digital Equipment Corporation approached Linus Torvalds to adapt Linux for their new Alpha processors;[7] he wrote this up for his Masters degree, in the process making it much easier to use the Linux kernel with any computer chip. This flexibility meant that Linux began to appear on a wide range of computer chips from supercomputers (Linux runs nearly all the top 500 supercomputers) to digital televisions. Only in the personal computer market does Linux hold less than half the market share though, with sales of personal computers in decline and sales of smartphones on the increase, there are now more smartphones running FOSS in the form of Android and iOS than there are computers running Windows.[8]

The charitable foundations

As Berners-Lee (1999) explains, one of the concerns of programmers during the 1990s was to ensure that their software was not highjacked by sectional interests and the answer they came up with was the charitable foundation. Apart from the Free Software Foundation and the World Wide Web Consortium (W3C), there is Software in the Public Interest (originally set up to host the Debian distribution but now acting as the umbrella organisation for a number of projects) and ICANN, which manages the allocation of Internet domain names. Other charities involved in developing free and open source software include: the Apache Software Foundation, the KDE and Gnome Foundations, the Mozilla Foundation, which develops the Firefox browser, the Linux Foundation which manages Linux kernel development, the Document Foundation which was set up to continue the development, under the name LibreOffice,[9] of OpenOffice, the open source office suite developed under the auspices of Sun Microsystems, and the MariaDB Foundation, set up to develop MariaDB as an alternative to MySQL after it had been acquired by Oracle.

All also provide software which runs on proprietary systems such as Microsoft Windows while Microsoft has contributed to the Linux kernel to enable Windows computers to work more efficiently with Linux computers.

[7] Digital Equipment Corporation was acquired in 1998 by Compaq and they in turn were acquired in 2002 by Hewlett-Packard who continue to produce the Alpha processor.

[8] While sales of personal computers using proprietary software are in decline, in most other areas sales of computing devices are increasing; so in some areas, the number of devices running proprietary software is going up even though their share of the market is going down.

[9] The name OpenOffice is owned by Oracle as a result of its acquisition of Sun Microsystems. Oracle declined an invitation to join the Document Foundation and in 2011 offered the OpenOffice suite to the Apache Foundation.

Part II
Features of FOSS

Quality assurance through peer review and cooperation

Most programs are developed and maintained by teams of programmers who may be employees paid or permitted to develop code as part of their job or programmers who develop code in their spare time. Some are sponsored by their companies to work for one of the charitable foundations or employed directly by a foundation.

Programs are normally peer reviewed because the source code is open for examination, users are encouraged to submit bug reports and anyone can fix a bug. So contributions are normally scrutinised by other programmers and volunteers often locate bugs or make contributions which it would take the full-time programmers many hours to locate or develop (Mockus et al., 2005). Along with the source code, the entire history of the digital communications between the programmers is open for examination providing unrivalled data about the behaviour, thinking and attitudes of programmers which few researchers have so far tried to mine.

After 9/11 the US Department of Homeland Security investigated software for its potential use by terrorists and concluded that, while all new software has bugs, bugs in FOSS were fixed more quickly than those in proprietary software. In 2104 Coverity, the company which now handles this work on behalf of the Department of Homeland Security, announced that the quality of all FOSS now outpaces that of proprietary software.

That said, bugs such as Heartbleed and Shellshock do get overlooked and often receive considerable publicity because everything is open. But security bugs are normally fixed within 48 hours because companies and individuals cooperate to provide a fix which is then issued simultaneously by all those involved to avoid a period when the issue of a fix by one group alerts cybercriminals to the possibility of taking advantage of the bug elsewhere.

Personal competition and cooperation

Successful FOSS programming requires a combination of personal ambition and the ability to cooperate (Coleman, 2013); programmers can only get their code accepted if it is better than the existing code but they have to cooperate with other programmers to get it included in the program. So the most successful programmers are fiercely competitive about writing good code but good at cooperating with the other members of their team in order to get their own ideas accepted. Given the debates in the last quarter of the twentieth century about the value of competition, this makes sense of what has long been known in many team games, that those who, however skilled they may be individually, cannot cooperate with the other members of the team are unlikely to be an asset to the team.

It also helps to explain why companies with the resources to hire the best programmers in the world have not always been successful in creating good quality software. Remunerating software developers based on the number of lines of code written rather than its quality does not result in greater productivity (Brooks, 1975) and, however good a programmer may be individually, without an incentive to write good quality code and the ability to cooperate, their team will be unable to produce good quality code.

Cooperation in a market economy

Major computer companies like Intel and IBM have recognised the importance of cooperating in FOSS development because they have recognised the advantage of bringing together the best programmers to create software that everyone will need rather than each company independently hiring programmers to produce their own software to do the same job. The FOSS model of software development means they have access to programming resources to which, as individual companies, they would never have access.

Rather than selling the software everyone needs, IBM now sells specialist software that runs on top of this and has cooperated with both RedHat and SUSE, who offer Linux support to IBM's customers, to create a completely new market in which, for example, virtualised server farms run on mainframe computers. Meanwhile, RedHat and SUSE compete for enterprise customers, for whom they offer specialist services, but cooperate on the development of all manner of FOSS. In addition, newer companies such as Google and Facebook, which owe their existence to the availability of FOSS, make many contributions both in software development and in cash and kind to the various software foundations which create FOSS.

Alongside this, numerous universities contribute to FOSS development or support, for example, by hosting websites used to distribute FOSS, while Google, through its Summer of Code, sponsors university students to work with FOSS projects over the summer period. The FOSS project must provide a mentor who supports, monitors and reports on the student's work while the student has the opportunity to write code which will be used in a real application and gets a valuable addition to their CV.

Leadership

FOSS projects tend to be meritocratic, that is, those who demonstrate their competence in writing code and in cooperating with others are granted positions within the team. Their leaders, however they are chosen, normally rely on a small team to assist them in the management of the project. Each member of that team normally has responsibility for particular aspects of the development of the software, in particular, who can make direct contributions to the code and whose contributions must be vetted before they can be accepted into the code.

Because projects can only grow — and sometimes be viable — if they sustain contributions from a wide range of people outside the core team and must from time to time replace members of the core team, project leaders and those who head up different aspects of a project have to create frameworks and relationships which encourage newcomers, develop existing contributors, ensure that contributions are of a satisfactory standard and enable disputes to be resolved quickly and amicably. This is complicated by the fact that, notwithstanding the personal ambition and the cooperative skills of the contributors, their motivations are diverse (Lakhani and Wolf, 2005). So those wanting to motivate contributors have to be prepared to recognise and draw on a wide range of motivations.

Different projects and different people approach these challenges in different ways but, with none of the incentives or sanctions which a manager might deploy in relation to an employee, they have to develop a range of interpersonal motivational and management skills which are going to work with volunteers who can withdraw their labour at any time. If they also happen to be employees, these skills can become a great asset to the organisation for which they work.

Learning programming

Many of those who drove the rise of FOSS learned their computing skills on command line driven computers in which familiarity with the operating system and the ability to edit programs to change the options available to the user were essential. With the rise of graphical user interface computers, most teaching, such as the ECDL (European Computer Driving Licence) or CLAIT, focuses on being able to use such an interface.

But FOSS development requires developers familiar with languages such as C++, Python, Perl or Ruby; they do not have to know the detail of each language to start with but they need basic programming skills that they can use whatever the language in which they are developing software. MIT supports the Scratch software which allows young children to learn how to program; the Raspberry Pi offers numerous opportunities to develop projects by programming the hardware directly.[10] Google supports the Summer of Code for undergraduates and has extended this to 13–17 year olds with the Google Code-in.

All these projects rely on the availability of FOSS and all ultimately, whether sponsored by universities or not, contribute to the development of the programming skills for which there is currently a world shortage because of the success of the FOSS model of software development.

The advantages of using FOSS

- Anyone can contribute to it; so academics who need a feature not otherwise available can create it, whether it be a Beowulf cluster, a Raspberry Pi supercomputer or a new bibliographic style for BibTeX.
- The pride in contributing to FOSS arising out of the competition to write the best code and the willingness to respond to user feedback mean that FOSS is generally of higher quality than proprietary software and thus more suited to academia where artefacts such as the well documented bugs in Excel can distort results.
- Academics who contribute to FOSS have their contributions acknowledged, in most cases peer reviewed, and made available for public inspection.
- FOSS provides case examples in software development, intellectual property, quality assurance, ethics, group dynamics, leadership, competition and cooperation, cooperation in a market economy and political action which challenge conventional theories in all these areas (Coleman, 2013).
- Most FOSS uses the UTF-8 Unicode standard recommended for Internet communications; so multilingual support is built in, documents can be exchanged in any language and there are FOSS user dictionaries for languages unsupported by any proprietary software.
- Other than for specialist software such as Mathematica,[11] there is FOSS software to meet most academic needs.

[10] An unexpected runaway success, the Raspberry Pi was originally conceived as a cheap credit card sized computer to enable schools to teach programming after Eben Upton, then an admissions tutor for computing courses at Cambridge University, became dismayed at the poor programming skills of applicants to the university's computing courses.

[11] Wolfram have announced a free version of Mathematica for the Respberry Pi.

Conclusion

FOSS is not only interesting because it emerged from the universities and has become the dominant way of developing software but also because it has spawned a whole series of developments in intellectual property, in support for users, in quality assurance, in harnessing competition and cooperation, in cooperation in a market economy, in new styles of leadership, in the uses of hardware and in the new ways of learning programming.

It does not matter whether someone sees their future inside or outside academia, the lessons to be learned from a study of the rise of FOSS are applicable in most other areas of life.

References

Anderson, R., C. Barton, R. Böhme, R. Clayton, M. J. G. van. Eeten, M. Levi, T. Moore, and S. Savage (2012). Measuring the cost of cybercrime. weis2012.econinfosec.org/papers/Anderson_WEIS2012.pdf.

Berners-Lee, T. (1999). *Weaving the Web: the past, present and future of the World Wide Web by its inventor*. London: Orion Business Books.

Biancuzzi, F. (2008, 7 April). Open source decade: 10 years after the Free Software Summit. http://arstechnica.com/information-technology/2008/04/free-software-summit-10th/.

Brooks, F. P. (Ed.) (1975). *The mythical man-month: essays on software engineering*. Reading MA: Addison-Wesley.

Coleman, E. G. (2013). *Coding freedom: the ethics and aesthetics of hacking*. Oxford: Princeton University Press.

Ferry, G. (2003). *A computer called LEO: Lyons tea shops and the world's first office computer*. London: Fourth Estate.

Glass, R. L. (2005). Standing in front of the open source steamroller. In J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani (Eds.), *Perspectives on free and open source software*, Chapter 4, pp. 81–92. Cambridge MA/London: MIT Press.

Knuth, D. E. (1981). *The art of computer programming: Vol. 2: Seminumerical algorithms* (Second ed.). Reading MA: Addison-Wesley.

Lakhani, K. R. and R. G. Wolf (2005). Why hackers do what they do: understanding motivation and effort in free/open source software projects. In J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani (Eds.), *Perspectives on free and open source software*, Chapter 1, pp. 3–21. Cambridge MA/London: MIT Press.

Mockus, A., R. T. Fielding, and J. D. Herbsleb (2005). Two case studies of open source software development: Apache and Mozilla. In J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani (Eds.), *Perspectives on free and open source software*, Chapter 10, pp. 163–209. Cambridge MA/London: MIT Press.

Patashnik, O. (1988, 8 February). BibTeXing. Documentation for general BibTeX users.

A Some academic contributions

Universities and research establishments such as Fermilab continue to contribute to many free and open source software projects, sometimes by allowing staff to use work time to contribute, sometimes by developing software in-house to address a particular issue and then releasing it and sometimes by directly supporting it. Many host mirrors of popular free and open source software websites.

Over the first decade of the twenty-first century Linux became the operating system of choice for supercomputers but even before that, FOSS had spawned the Beowulf cluster, that is, a high performance, parallel processing computer build out of commodity hardware, many of which run in universities to support particular scientific needs.

In 1997, Ross Ihaka and Robert Gentleman from the University of Auckland released the R programming language.

In 2005, William Stein, a mathematician at the University of Washington, issued the first version of SAGE.

In 2006 MIT released the Scratch programming language for schools.

Sussex University hosts Snuffler, free geophysics software.

In 2012 a team at Southampton University led by Professor Cox built a supercomputer using 64 Raspberry Pi computers.

In 2013 David Spencer described how the Stanbury Hill Project, to investigate a rock art site on Bingley Moor in Yorkshire, was undertaken entirely with free and open source software.

B Some FOSS relevant in academia

Meetecho: web conferencing software

R: a free software environment for statistical computing and graphics, based on S

QGIS: A Free and Open Source Geographic Information System

SAGE: free and open source mathematical software

Lilypond: music engraving software

LibreOffice: a free office suite compatible with MS Office

Scientific Linux, a collection of FOSS distributed by Fermilab for use by those involved in similar research

Gnumeric: more accurate than Excel with mathematical and statistical functions not found in Excel

Snuffler: Free Geophysics Software

Snuffler

Tomboy: a note taking app to help you organise the ideas and information you deal with every day

LyX: the complete student's/researcher's/teacher's writing software from initial thoughts to typeset pages