FOSS 2.0

There is a published printed extended version of this resource here. [1]

Foreword

The purpose of this text is to explain what Open Source Software is, why it is important and what it might be able to do for you. Before you can understand the significance of Open Source software you need to understand licensing and its relationship with community development, technological change and its impact on society.

The texts and references could be of help to a wide audience, ranging from first-time users to whom open-source is just a strange combination of words, to users in need to advance their skills in using and working with open-source software tools, to IT people in different job positions. Not least, these texts have been compiled to help IT teachers and assessors to advance and improve their work and the work of those who they teach.

It is very important to understand the pace at which technology is developing, and to consider these texts as a basic guidance tool, with many branches and links to outside resources. It is very probable that by the time this document is completed, it will be already outdated by latest technology and software progress. This should not discourage the readers, as we are mainly focusing around the basic principles. Evolution and progress follow typical patterns and solid grounds are key to updating IT competences.

We start with an overview and then delve deeper. Below is an index list so you can either go through the information in order or jump to specific topics.

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');

What is Open Source Software and why you should be interested

Open Source Software is a commodity that can substantially lower costs associated with ICT. You probably expect to pay for licenses [2] to run software applications, to upgrade software to fix bugs and to provide enhancements. On a large school site you probably expect to pay someone to manage which software titles are installed on specific computers in accordance with the licences you have purchased. Open Source Software is different. With Open Source you do not need to pay any license fees and there is no need to manage which computers run what software. This might sound too good to be true and part of the purpose of this web site is to explain why it isn't even though it might be counter-intuitive. In addition to cost savings some people believe that Open Source is simply more ethical and a better fit with a learning culture than commercially licensed software.

Open Source Software has been around for a long time, but the internet has made the development model increasingly compelling. The major industry analysts [3] and most powerful multinational technology companies [4] have acknowledged that the rise of Open Source software is the biggest change taking place in the software industry since the personal computer. With these things in mind it is important to learn more about Open Source to be considered technologically literate. There are real practical benefits that support improvements in value for money as well as preparing for the learning future.

The Drivers of Open Source Software development

The key driver of Open Source Software development is economic. Although the modern Open Source movement can be traced back to Richard Stallman [5] and his drive for an ethical approach to the work of computer programmers, the main force now driving Open Source Software development is a complex combination of co-operation and competition between large multi-national companies.

Google [6] funds Open Source development through its summer of code [7] and other initiatives. Many other large companies do too. A particular milestone in the corporate acceptance of Open Source was in 2001 when IBM [8] claimed to be spending over a billion dollars on GNU/Linux and adopted the open source operating system for its super computers and mainframes. Sun Microsystems [9], Novell [10] and newer companies like Canonical [11] also invest substantially in Open Source development. The reason is that they believe collective development is less risky than allowing any one competitor too much power in the market place.

While the current Open Source and Free software culture was initiated by individuals like Richard Stallman, to-day, the image of the lone hacker [12] in his bedroom single-handedly changing the world is more folklore than reality. In some smaller projects lone hackers are the most significant contributors but increasingly, the development of the core productivity tools is substantially funded by contributions from the large corporations. Even Microsoft [13], arguably the company with most to lose from Open Source competition [14], recently contributed $100,000 to the Apache [15] Web Server project.

The shift from a world of proprietary software with restrictive commercial licensing to licensing that actually promotes sharing the ideas and code, extends to digital content. Click on the link to Larry Lessig [16] to hear the Stanford Law Professor talking about the way the internet threatens the established commercial interests associated with copyright, the reactions to this and why these reactions can stifle potential for creativity in learning. Lessig is not alone in believing that the philosophy and ethics of freeing software and digital information for creative use in education has benefits that could easily outweigh any of the more obvious advantages from cost savings in buying licenses.

Many people realise that the power of science in changing the world was that scientists were free to take and share ideas. Compare this with current commercial approaches that tie up ideas and knowledge in patents and copyright often preventing or tightly controlling further development. The internet has disrupted the balance of power, shifting control of digital resource production to a position where Open Source Software and open resources like Wikipedia [17], have taken on a new hegemony in the world of ICT.

Wikipedia - A Case Study in Open Collaboration

In his 2008 book, *"The Future of the Internet and How to Stop It"* [18], Jonathan Zittrain [19] of the Oxford Internet Institute [20] and Harvard Law School's Berkman Center for Internet & Society [21] cites Wikipedia' success as a case study in how open collaboration has fostered innovation on the web.

Wikipedia was launched by Jimmy Wales [22] and Larry Sanger [23], in January 2001. It is now the largest encyclopaedia ever with 10 million articles in 253 languages. [24] Finance is provided by the Wikimedia Foundation [25] with operating costs currently about £1m per year, comparable with the cost of running [26] a primary school in the UK. It is free and carries no advertising, relying largely on donations of individuals' time and intellects.

Wikipedia is an example of the power of the internet in supporting community development that massively lowers centralised costs. This is a case study in the eradication of conventional bureaucratic structures to increase productivity and lower costs through collaboration. It is evidential support for Eric S Raymond's classic work, the Cathedral and the Bazaar [27] that goes beyond software development.

Enabling participation by a massive number of contributing volunteers world-wide has completely changed the way we think about digital resource development. While the most obvious benefit of Wikipedia to schools is a free on-line research tool there are more important implications for the way education is delivered in the Twenty First Century. Learners can be involved in the process of knowledge, synthesis and information transfer to an unprecedented level. They can discuss and debate reliability and understand the need to cross-reference research. This is an environment that is created for the support of personalised learning and the promotion of social inclusion. However, as with most worthwhile change, exploiting the full potential in schools will not be easy. How do these new methods [28] compare to traditional books controlled by a single publisher? A single teacher in control of the learning of a group of children? A highly centralised national education bureaucracy? Open Source Software and open content development methods will have a wide impact on education in general but only to the extent that those working in the field understand the issues.

The Wikipedia information organisation factory mirrors on a less technical and less rigorous level the culture and production methods of Open Source Software. Open Source software applications are generally much more reliable than Wikipedia because they are more focussed and employ more rigorous quality assurance methods. Wikipedia is itself enabled through the development of Open Source Software and it is built on an Open Source Stack [29]. At the top of the stack is MediaWiki [30], a custom-made, free [31] and open source [32] wiki software [33] platform written in PHP [34]. The next stack layer is the MySQL [35] database which is also Open Source and is served from GNU/Linux [36] operating system software using Apache [15] and Squid [37], two other widely used Open Source applications. This software solutions stack is widely used in smaller projects such as SchoolforgeUK [38] as well as those of global significance and it would be quite feasible for an individual school to set up it's own MediaWiki site. The use of Open Source software throughout has a practical aspect in keeping the costs of running Wikipedia to a manageable level and the infrastructure is a lot more reliable than the content. Open source infrastructure is congruent with the culture, ethos and ethics of community projects and many in those communities consider this to be more important than saving money.

Copyright and Intellectual Property

If you want to understand Open Source Software, understanding the principles of copyright is essential. There is no absolute right for an individual or company to control their "intellectual property" [39] forever. As can be seen from the link, some people believe that the concept of intellectual property is flawed since all knowledge is based on other people's learning. Newton said if he saw further it was because he was standing on the shoulders of giants. Newton pre-dates the age of copyright and patents which are historically recent concepts. Stallman takes the extreme view that with software all code should be free and open. Lessig is more liberal but says the balance has been allowed to drift too far in favour of lengthening the duration of copyright.

Historically, the intention of granting short term monopolies was always to give enough protection to encourage creativity for the common good not primarily to maximise profit for particular interest groups. As time has gone on, strong commercial lobbying has extended the period that copyright exists and many people accept the idea that ideas and their expression can be owned just like property. Technology has enabled a backlash, some of it legal and some illegal, in the eyes of the law. These opposing views on intellectual property inevitably lead to political conflict. Licensing copyright material such that wide uptake is encouraged and using that mass audience to sustain development through alternatives to selling licenses neatly side-steps the legal issue. Google uses advertising, Canonical and Novell sell services. Many people contribute to Wikipedia because they have a specific interest or enjoy social status. If these new business models continue to strengthen and masses of unrestricted work becomes available, less and less use will be made of that which is restrictive.

Learning about technological change

Since the global analysts say [3] that Open Source Software is the biggest change taking place in the way software is produced since the early 1980s, the often claimed reason for teaching using expensive proprietary software tools - "because they are what they will use when they leave school" - is at best out of date marketing rhetoric. We don't know what will be the most used technology in 5 years time but we do know it probably won't be what is the most used now. This video [40] about the semantic web shows how importance is shifting from desktop documents to a wide range of information objects on the internet and web publishing. Migration from local installation of software on desktop machines to running applications from the internet though a web browser (cloud computing) [41] is already happening. Here [42] is what Eric Schmidt CEO of Google has to say. The current model of paying for software licences simply won't work in the world Schmidt describes.

As a more immediate example, building a personal web site within Facebook [43] or Myspace [44] is much more common than web sites created in desktop bound software like Frontpage and Dreamweaver. This change has happened in the last 5 years. How many schools teach web site development using the new methods [45]? They are actually simpler and more accessible as well as being more up to date. As more choice of applications becomes available directly from the web, the notion of installing and maintaining expensive applications on local computers will become less and less attractive.

How is Software Produced?

Software [46] is a collective term for the sets of instructions that are used to program computers. Collections of computer programs are organised into suites of software that help people carry out tasks on devices such as mobile telephones, music players, set top boxes, and of course desktop computers.

Software is written in languages that can look remarkably similar to English [47] with commands like print 'hello world' and statements like "if number >1000: print 'big'" which most people could understand. Programs written in this way have to be converted into the binary codes [48] computers use. Special programs called compilers and interpretors do this job. So for software there is the **source code** that programmers use to create programs and there is **executable** or **binary code** that the computer actually uses. To humans, even programmers, the executable code [49] is unreadable, just a mass of 0s and 1s. The source code is what is necessary to understand how the software works.

Open Source Software development is a method for producing software dependent on access to the source code by a community of developers. Crucially, in order to change the way that the software works (for example to fix a bug, add a feature, or make it work with other pieces of software), access to the source code is essential. When you buy a license for a proprietary software application (for example, MS Windows, Adobe Flash, Norton Anti-Virus etc.) you only get a licence to **use** the executable code, usually on one computer. You don't get the right to change the software and without the source code it would be near impossible to do so. The source code is likely to be held in a secure place in the company with restricted access even within the company itself.

Although the most visible examples of computers running software are desktops and laptops running Microsoft Windows, these are a small minority of the total number of computers in use today. For example, mobile 'phones [50] out-number desktop computers by at least three to one. Open source software is taking over [51] the mobile 'phone market, at least partly because traditionally 'phone operating software has come from several different sources and the functionality of a 'phone is not so dependent on specific legacy applications [52]. It makes it easier for mobile 'phone manufacturers to migrate to Open Source and there is competitive pressure to do so.

Other examples of devices that are actually computers include set top boxes, satellite navigation systems, digital cameras and portable music/video players. Software is essential to any computing device, whether it is a car satellite navigation system or multiprocessor file server.

How is Open Source Software Licensed?

Open Source Software is software for which the code that programmers use to create the software is licensed to be open and freely available for anyone to use. One way of identifying products with an Open Source license is if they carry the logo of the Open Source Initiative (OSI) [53] although not all Open Source Software carries the logo.

Access to the source code is essential if we want to see how the software works or want to change something. Most software users have no interest in looking at how the software is programmed so why is Open Source such an important issue? It is the consequences of openness and the ability of the few with the skills and knowledge to review and improve the software transparently that confers many more benefits to all users.

The opposite of Open Source is Closed Source sometimes referred to as proprietary software. Closed Source Software does not have freely open and available source code, only the compiled binary code is supplied to the user. While closed source development is perfectly legal and has spawned a massive and successful industry it has also thrown up some very significant problems.

- **Inconvenience to the user** - license keys and administration, risk of being sued for inadvertently forgetting to delete unused software, suppliers that give a poor support service because the customer's data is locked-into their software and it is difficult to transfer to competing products.

- **Lack of competition** - Software monopolies provide power to the copyright owner at the expense of the customer especially if there is a high exit price from the dominant products. The customer becomes unduly dependent on the product with all the associated risks of that situation. These include enforced upgrades, price premiums and unilateral termination of support. In general, it is accepted that lack of competition leads to higher prices and lower quality.
- **Commercial priorities drive development** - commercial entities will naturally concentrate on developments that make them money. For example, in the case of software licensing, version 2 of a piece of software which will generate income from an upgrade cycle will often take precedence over fixing bugs in version 1.
- **Lack of public scrutiny** and accountability - sometimes, when software is used for very important activities, there may be a requirement for public scrutiny of how it works (by looking at the source code) to check for fairness and accuracy. Common examples here include lottery systems, public election systems or software used in defence systems.
- **Environmental issues** A lot of closed source software is distributed in extravagant packaging. In some cases large cardboard boxes full of packing simply to provide a disc and paper licence. This is not environmentally friendly. It is much more straightforward to distribute Open Source Software over the internet because of the way it is licensed.

How is Open Source Software is different from closed source software?

Open Source Software provides a way to solve problems that are a consequence of the Closed Source approach. Users can be more directly involved in how the products are developed. Removing restrictions increases convenience to the user and costs can be fundamentally reduced. There is far broader scope for scrutiny of the source code to determine quality. Removing some of the barriers to use, increasing choice and accountability and making the whole process of software development and deployment more transparent and democratic are the characteristic benefits that drive Open Source.

It is a condition of the license that Open Source Software is free to obtain, install and use resulting in the eradication of license fees. Obtaining the source code (and executable files that run on the computer) is effectively free of charge. However, it is the production method, enabled by the licensing terms and conditions, that makes Open Source different. It is more than just getting the code without having to pay a license fee.

Why Open Source can save money

Freeing the source code from restrictive licensing can reduce development costs [54] through programming contributions from the user community and massive economies of scale enabled through a massively connected world. It is easier to build new products on existing code without having to begin everything from scratch because of licensing restrictions. The power of the internet in distributing software and enabling collaborative development has accelerated the importance of these key attributes. While Wikipedia is not an Open Source Software development project, the development methods are similar demonstrating how a very large project can be sustained from minimal centralised resources. The quality assurance issues for Wikipedia are generally more significant than with Open Source software development because the control of contributions is much more liberal. Here [55] is the Chief Technology Officer at Sun Microsystems explaining why Sun is fully embracing Open Source for all its software products.

Of course there is huge inertia in currently entrenched systems that will not disappear over night. It is not absolutely clear which applications lend themselves to the most economic Open Source support and which might remain exclusively Closed Source but the trend seems to be that Open Source applications arise most readily in mass take up applications such as operating system software, productivity tools and web services. There are Open Source equivalents now for most of the general productivity tools and the quality of the software keeps improving. Moodle [56] is a specific example of an Open Source de facto standard emerging in a market not strongly locked into proprietary products. It is now the clear market leader in the Further Education sector, where on-line learning platforms had the earliest adoption. This is largely due to lowering the barriers to take up and the cost savings over proprietary alternatives, although some also say it is easier to use and better designed. A company supporting Moodle does not have to pay for maintaining the code base on their own and they can make their products distinctive either through specific content or specific approaches to service or quality assurance.

Why is Software an unusual commodity?

Commodities fall into two broad categories:

1. Manufactured goods (cars, TVs, and furniture)
2. Intellectual work (books, music and films)

Intellectual property [57] is the ideas behind commodities. Intellectual property law provides different rules for protecting different types of commodity from indiscriminate use. This law was developed well before the digital age but is still recent in overall historical terms. Inventions can bepatented providing a monopoly for that inventor for a fixed period of time to encourage such inventors to share their ideas. Usually these inventions relate to manufactured goods but in the USA they can relate to business processes and software too. Intellectual work is subject to copyright [58] of the originator as soon as it is produced but you can't copyright an idea only the expression of the idea eg a music score, a book, computer digital resource or film.

Software is something in between the two categories. It has a functional purpose like manufactured goods but in terms of creation, manufacture and distribution it is more like intellectual work. This is important because it directly affects the way intellectual property [59] law works and has caused significant political debate. The notion of a patent enabling the sharing of ideas when the expression of those ideas is impossible to access because it is locked up in closed source code is at odds with the original intention of patent law. Software patents lead to many anomalous cases [60] such as a company suing another for patent infringement without allowing the target of the lawsuit to see the code that the patent is claimed to protect. In practical terms this means that to defend any such case you need massive resources to pay to go to court. Until you get to court you have no way of knowing whether you have accidentally infringed a patent or not, all you have is an accusation that might be a bluff. The idea of an individual programmer being able to patent a novel piece of software to her practical benefit seems very unlikely and this might explain why millions of programmers are contributing to Open Source projects.

Many people believe that the law has not kept up with technological development and that the characteristics of software make it very unsuitable for patenting. UK law does not recognise software patents. It is arguable that the success of the internet is built on the antithesis of the principles of intellectual property, particularly in relation to Open Source Software and Open Standards. Tim Berners-Lee [61] made the World-Wide Web possible by encouraging unrestricted use of HTML [62] for all. Unrestricted access is also a key behind the success of Google, You Tube and Facebook. This is contrary to the argument that creativity and invention will only be achieved in a world where patents and copyright restrict usage to those that pay royalties. Software is an unusual commodity that does not fit comfortably into the "intellectual property" boxes and this probably explains why Open Source software is becoming increasingly widespread.

Open Source software and freedom

Software is protected by copyright as soon as it is written. The copyright holder issues a licence saying who can use the software and under what conditions. There are many different types of licence but to be an Open Source licence it must allow 4 fundamental freedoms first identified by the Free Software Foundation [63].

- Freedom 0: The freedom to run the software for any purpose.
- Freedom 1: The freedom to study and modify the software.
- Freedom 2: The freedom to copy the software.
- Freedom 3: The freedom to improve the software.

For this reason you will also see Open Source Software referred to as Free Software or even FOSS or FLOSS. FOSS is Free, Open Source Software, FLOSS Free, Libre Open Source Software. Libre is included to make the distinction between being free of cost/charge and being free as in the freedoms itemised above. Open Source Software is free of cost at the point of use but it is not free of cost in terms of its development and support. Having said this, there is no evidence that in general terms Open Source Software is any more expensive to support than Closed Source. The sustained existence of Open Source software on a global scale seems to make it self-evident that the economic case for Open Source is sound.

It is misleading to refer to Open Source as "non-commercial" since there is no reason not to make money from it. It is simply that the business model based on selling software licences does not apply. The really non-intuitive aspect of Open Source Software is why it exists at all if selling licences is so important to sustaining development. The link here [64] goes some way to explaining why many large corporations are embracing the Open Source concept but competition is at the heart of the matter and the freedoms provide competitive advantage to those that embrace them. While some Open Source projects are entirely volunteer-based, increasingly the larger projects are backed by the biggest players in the industry and competition [65] between them is growing. The actual coding might be shared between many individuals even in competing organisations all over the world. The effect is that competition shifts to the service rather than in trying to establish monopolies quickly based on software code or even content. This benefits end users but not those that have a commercial interest in preserving vendor lock-in.

Vendor lock-in

When considering the cost implications of licensing particular products, the cost of lock-in [66] to that product over 10 years or more should be considered, not just the immediate cost of change. Software freedom through Open Source helps reduce the opportunities for vendors to lock customers into their products since there is no commercial incentive for the supplier to try to prevent the user moving to an alternative technology. Indeed they might make money from supporting such a move. This means that the return on investment [67] from a switch to Open Source is easily under-estimated. This is even more the case for those that put a value on the ethics and educational potential of community participation.

Open Standards [68] are an important protection against vendor lock-in and although different and broader in scope than Open Source Software, there are strong relationships [69] between the development of Open Standards and Open Source Software. A good example is ISO 26300, the Open Document standard, which originated in the OpenOffice.org [70] project and is now supported by Google, Sun, IBM, Microsoft and many others. International open standards [71] are mandatory for governments according to world trade agreements and while enforcement in software has been low key until recently it is becoming more vigorous as the cost of mass licensing linked to lock-in [72] to proprietary de facto standards comes under greater scrutiny. Again the internet has had a big influence on this. In general, Open Source software projects tend to readily adopt and promote open standards as there is no reason to lock users into a proprietary standard. As with OpenOffice.org, they also spawn new open standards.

Many Closed Source applications support open standards and an industry trend is for increasing numbers to do so. The dominant players realise that it is going to be increasingly difficult to do business with governments if they don't support open systems. Commercial strategies to prevent customers from migrating to competing software products have traditionally included the use of de facto standards for file formats [73] and communication protocols [74] but it is increasingly difficult to gain the size of customer base needed to do this with new products and the current heightened awareness of the risk to the consumer. This leaves the legacy de facto standards very much a target for change. As part of any risk analysis in software procurement, consideration should be given to the support of Open Standards whether or not the software is Open Source or Closed Source.

Google [75] has something like 700,000 servers running GNU/Linux. Think of the savings in licence fees, it's certainly tens of millions, but also consider that Google is free from dependency on any other company for their technological life blood. They can modify the software they use or extend it without asking anyone for permission and they are only liable for a small fraction of the full support and development costs of the software. Since GNU/Linux is subject to GPL licensing we all benefit because any improvements made by Google come back to the commons. Google is free from lock-in to potential competitors and has lower costs. In this case the open source mantra of "give a brick and get a house" tips the commercial balance away from lock-in and towards sharing.

On the scale of global multinational companies and national governments the costs of licensing general productivity tools and operating system software runs into hundreds of millions and may exceed the development and maintenance cost of the software itself. Once a major application is Open Source, adding to it and refining it or customising it to your own needs is far, far less expensive than starting from scratch and the bigger you are the less sense it makes to be locked into a particular commercial interest.

What does vendor lock-in mean to my school?

At a simple level, if large commercial organisations see the benefits in terms of making their businesses more efficient why would that not apply equally well to schools? There is also the wider issue of global standards and management of technological change. Back in the 1980s, the dominant computer in schools, was the BBC Micro. When the large industrial companies started standardising on the IBM PC, education followed even though there was little appropriate software. The technology was in many ways poorly matched to education with considerably more expensive software, lack of colour graphics, lack of sound and lack of standard ports for scientific measurement and control. The change was slower and more painful for those that were most locked into their legacy systems but eventually virtually everyone made the change, hardware costs fell and software was ported to the new platform.

The simple lesson to learn is that the hidden cost of lock-in is large and significant but difficult to quantify with any precision. If it can be avoided it is best to do so because trends set by the corporates globally will certainly determine the standards of the future. The pressure for change is building and at some point it will become unavoidable. As a minimum, learning about the characteristics of products to choose and those to avoid is just part of the lifelong learning supporting technological literacy that is essential to education.

One advantage we all have that was not available in the 1980s is the Internet. The open Internet removes much of the risk of vendor lock in. If an applications runs in a standards compliant web browser that is cross platform (eg Firefox or Opera) it is very unlikely that users will become dependent on any particular browser supplier or indeed any particular operating system supplier. The user will be free to choose whichever operating system gives best value without worrying whether or not applications are compatible with it. Effectively the Internet becomes the computer platform. This is a trend in any case with most e-portfolio and learning platform software web based. It is really not sensible to buy into any web application that requires a specific browser or proprietary plug-ins specific to a particular operating system. Schools should avoid vendors if they suggest their web products only work with a particular operating system or browser because the costs further down the line are a significant risk. In practice this situation is much more likely to arise with closed source than with open source products.

Types of Open Source Software

There are Open Source applications available for most widespread technological needs. The most common operating system for servers and desktop computers is the Closed Source Microsoft Windows. For telephones and mobiles Symbian is the market leader and although currently Closed Source, Nokia, the owners of the Symbian copyright, intend to make it Open Source. If so, 75% of the mobile market is likely to run Open Source operating systems by 2010. That will be the biggest volume of computers overall and of particular significance as smart phones and similar technologies increasingly encroach on what have been traditionally desktop applications. It is much more difficult to estimate the proportion of servers and desktops that employ Open Source operating systems because the statistics are compiled from shipments of new computers with installed operating systems and the main Open Source operating system in this market, GNU/Linux, is often installed after the hardware is shipped. There are also mass initiatives [76] that won't be obvious in UK schools. Nevertheless, Windows has a clear majority in the server market and more than 85% of the desktop market. For this reason the biggest take up of desktop Open Source applications is on Windows.

Open Source Operating system software

- Berkley Standard Distribution Unix (BSD) [77]
- Gnu/Linux [36](most people call it Linux for short but Linux is just the kernel, the rest of the operating system is provided by GNU)
- Solaris [78]

The above are all based on Unix [79] but the licences are different. BSD's licence [80] basically says you can do whatever you like with this software but don't sue us! This has enabled Apple to take the BSD code and turn it into Mac OSX [81] which is not Open Source. The GNU/Linux licence is the GPL (GNU Public license). [82] It supports the 4 freedoms but says that if you modify or improve the code you have to make your modifications and improvements available to others with the same freedoms that you were given. This means it can't be turned into a closed source variant. It's possible that it is this aspect of the licence that has ensured that GNU/Linux has proliferated more diversely to a bigger range of hardware than any other operating system software ever. Unlike BSD you can't just take GNU/Linux and change it from open source to closed source. Solaris [83] is Sun's operating system derived from an earlier version of BSD and is available under the Common Development and Distribution Licence (CDDL) [84].

Server and desktop operating systems

An operating system [85] is the software that acts as a consistent layer between the hardware and applications programs. There is not much difference in substance between the actual operating system software code for a server [86] or a desktop machine, it is simply the emphasis on the tasks performed. On a server, the software will be mainly sending out and receiving information from many users connected to it by a network. On a desktop, the machine will be running applications for a particular user.

Servers

Using Open Source software as the operating system for a server is least likely to cause any dissent from users because they will probably be unaware that it is there. Anyone doing a Google search or buying a book from Amazon [87] is using GNU/Linux and huge numbers of internet web servers run on a GNU/Linux platform. The reason such action is not more widespread in schools is that server management requires the most technical knowledge to implement and Windows server software licenses are relatively low cost with schools discounts. There are clear implications for training and putting "Investors in People" into practice if the take up of GNU/Linux servers is to increase in schools. If your school does have technical support staff confident to implement Open Source software it would be well worth considering strategies to ensure you keep them.

You will find GNU/Linux used for serving web pages [88], for caching web pages as proxy servers [89] and as firewalls [90] for security in many schools. File serving, [91] e-mail serving and calendar sharing [92] are also quite feasible but require more technical expertise. Thin client [93] servers can run a range of applications with no license fees either for the server software, client access or the user applications.

Desktop

It is more difficult to get acceptance of Open Source Software desktop operating systems (GNU/Linux, BSD) because this requires ordinary users to accept changes in the appearance of the tools they are using and there is uncertainty in being able to run applications that are dependent on Closed Source operating systems (Windows and to a much lesser extent Apple).

In order to make the transition easier there are many Open Source Software titles that are cross-platform. This means that there are versions of the software that run on more than one operating system eg Windows, Apple and GNU/Linux operating systems. It is far easier now to "port [94]" software between operating systems because the technologies and standards to enable this have developed to much greater power over time. If new applications are not supported across platforms it is often because of the commercial interests of the supplier rather than because it is technically difficult or expensive to do it. This is a reason why Open Source applications are far more likely to be cross-platform than Closed Source.

So if we take the Firefox [95] web browser as an example, it can be installed on a Windows computer, a GNU/Linux computer or an Apple computer. Internet Explorer in it's later versions only runs on Windows at least in part due to anti-trust [96] rulings. Since Internet Explorer comes pre-installed on Windows it gets most use because most people have simply never tried anything else. One reason for using Firefox in a school environment is that it has an integral spell check so when learners are doing e-portfolio work they are much more likely to pick up errors and learn how to spell new words. Since Firefox can be installed freely and runs perfectly well alongside Internet Explorer there is no reason not to have both and let people have a choice of the tools that they use. There are a growing number of web browsers available. Opera [97], Safari [98], Konqueror [99] to name just three. Open Internet Standards provide users with choice, increase scope for innovation and ensure that no single commercial interest gains control of the World's information.

Desktop Office productivity tools

OpenOffice.org **[100] (OOo)** provides an Open Source equivalent to MS Office. There have been over 100 million download [70]s of the Windows version of OOo and it is distributed as a matter of course by just about every GNU/Linux project. On the downside, like MS Office, Openoffice is a huge piece of code and it needs a fairly up to date computer to run well. It isn't an exact replica of MS Office but it has a lot more things in common than differences. The file filters are excellent and even very complex Word documents will open in OOo. In fact, corrupt documents that won't open in MS Word often do so in OOo so it's probably worth having at least one copy installed just for that purpose. OOo can produce pdf files [101] directly from any of its applications and version 3.0 currently in beta test can open pdf files for editing too. Later versions can open the new MS docX [102] files that MS Office 2003 and earlier can't. Since OpenOffice.org is Open Source, you can always download and install the latest version and there are no license fees to pay. In terms of inclusion, learners can freely download OOo and use it at home. It is certainly more capable than MS Works [103] and more compatible with MS Office than MS Works. The native file format of OpenOffice .odf, is an ISO standard and based on the internet XML standard. After an intense battle Microsoft now intends to fully support odf [104] in its next major release of its Office software.

There are a number of other desktop application such as AbiWord [105] word processor and Gnumeric [106] spreadsheet. These are smaller and more compact than the full OpenOffice.org suite and will work quicker and on lower specification machines.

Mind maps [107] are a visual tool representing ideas, thinking process, relationship between tasks, and everything else which can be constructed around a central theme. Mind maps can be build in advance of training and shown as a presentation, but also during a training, with new nodes and entries added as discussion unfolds, showing the trainees how the whole process works. It is particularly useful in training sitiations related to brainstorming, planning, and project management.

Free Mind [108] is among the leading open-source solutions for mind mapping. It is a cross-platform, free, Java-based application, released under GPL license. FreeMind features folding branches, saves and exports to XML, HTML, XHTML, PNG, JPEG, SVG and OpenDocument formats. Developers have created plugins for FreeMind files to be used in some Wiki and CMS, such as Drupal, Moodle, MediaWiki, and other.

Visual Understanding Environment [109] (VUE) is an open-source project based at Tufts University [110] (USA). At its core, VUE is a concept and content mapping application, developed to support teaching, learning and research and for anyone who needs to organize, contextualize, and access digital information. Using a simple set of tools and a basic visual grammar consisting of nodes and links, faculty and students can map relationships between concepts, ideas and digital content. VUE has more sophisticated instruments for constructing mind maps than Free Mind, and its website is packed with examples and video tutorials.VUE is a cross-platform, Java-based, and is available under Educational Community License [111] (ECL)

Web applications

Moodle [56] is another Open Source application that is having a big impact in the education sector. Moodle provides on-line courses, blogs, forums and similar facilities. It has largely displaced equivalent proprietary software applications in Further Education, a sector that engaged web based learning much earlier than schools. Click here [112] to see the take up statistics for Moodle over the last 5 years. It seems very likely that Moodle will become the de facto [113] standard for on-line courses for schools and colleges around the world with 78 languages already supported including Welsh. One of the arguments often used against Open Source is that there is no support. As far as Moodle is concerned, this is clearly not the case and probably Moodle is the best supported application of its type in the World. There are commercial companies supporting Moodle as officially sanctioned partners and a huge community of users where the culture is strongly orientated to sharing.

Moodle is not the only open source software intended to support learning communities. There are so many content management and community support applications available we are spoilt for choice. This web site is built using an application called Drupal [114] which some schools have used to build their own web sites [115]. There are also students using Drupal for e-portfolios and blogging [116] about their work. The software makes it very easy to delegate site maintenance to individuals who need only a web browser to update and develop new pages. Another prominent content-management system (CMS) similar to Drupal is called Joomla. Joomla [117] has a very large and vital community of developers and supporters, which make available online tens of thousands of add-ons - components (functionality), modules (how content is being shown), plugins (small code snippets for very narrow tasks). Drupal, Joomla and other CMS software enjoy a remarkable selection of freely available appearances, known as themes and templates.

Similar applications include Mahara [118] which is designed to support e-portfolios. Here [119] are examples from students teachers sharing learning resources in the field of teaching and learning. When we look at initial learning of younger and less experienced learners we find a lot of dependency on desktop files. Here [120] is an example where the links made in the developing portfolio are all to desktop files. A better approach would be to link to web pages because these would not require the user to have any specific software other than a web browser to access the information. It would also be quicker for the author to prepare these and integrate them more seamlessly with other internet based resources. This is why learning about which ICT tools to use and why, is important. In this particular case, the use of proprietary file formats, when not strictly speaking necessary, further reinforces dependency on paying for software licenses. This reinforces the digital divide and is contrary to policies related to social inclusion.

Mostly the Open Source software that supports these community software projects is modular so some components such as rich text editors, forum management and blogging can be interchanged between applications and indeed Mahara and Drupal can both use some of the same software components. Small groups can work on individual components and make contributions to the software which can then be used in new and evolving applications. This leads to economic development through more efficient use and re-use of existing resources. There are single sign on facilities for Moodle, Drupal and Mahara so that if a user creates an account in one application the account can be automatically replicated in the others.

A logical extension of this is that there is no such thing as a Learning Platform. The internet is the platform with user information being stored in many inter-connected databases and operated on by a range of internet based software tools (see cloud computing above). This makes the concept of "buying a learning platform" obsolete. Schools will be buying services to support learning not specific technologies.

Note that the Open Source Software aspect is the actual code that manages the users' information. The content in the form of e-portolios etc could still be open or closed. With the growing mass of Creative Commons content available such as that from Wikipedia and You Tube, even our current concept of buying licenses for content is not safe. It is easy to see how learning might be strengthened by the learner creating their own contributions to the Creative Commons rather than passively consuming commercially licensed content. Businesses would then shift to facilitating that process rather than specifically focussing on producing software and content themselves and selling it.

While each individual copyright owner can decide how they want to license their work, the work of one individual is unlikely to get extensive use on its own unless it is particularly excellent. Collectively the work of many students will be more likely to get noticed if it is of sufficient quality. This is the Open Source principle of "give a brick and get a house". If everyone makes a small worthwhile contribution each individual gets back many times their contribution to the shared resource and a lot of learning can take place in that process. The internet makes it possible to share and publicise such resources inexpensively and challenges business models based on monolithic developments. Wikipedia [121] is only one example of the success of the new approach and demonstrates the relationship between Web 2.0, Open Source Software and Open Standards.

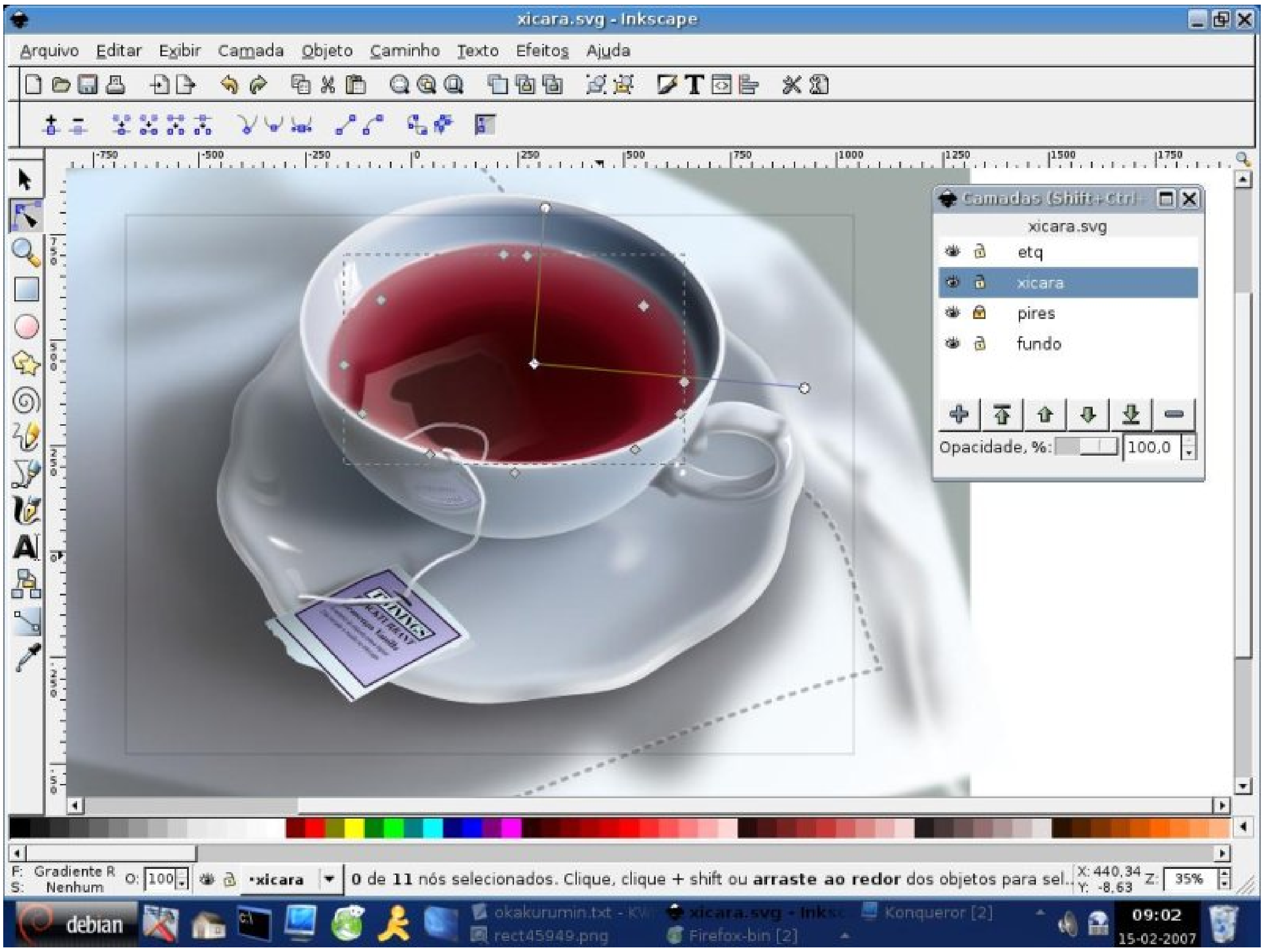
Finally, here [122] is Larry Lessig arguing that the cultural and educational importance to build creatively on existing resources transcends the way copyright law is currently widely applied.

Using Open Source Software in your school

There are many Open Source applications that run on the Windows Operating System. In most cases these are cross-platform and run on GNU/Linux and often Apple computers too. Starting with one of these applications is simple and requires no great investment or technical knowledge. It is therefore a good starting point for any school wanting to try out Open Source practically.

Audacity [123] is a sound editing program that is very popular in schools. It can be used to digitally edit and mix audio tracks. Here [124] is a You Tube video that shows how to use Audacity to edit a sound track. There are many video tutorials like this one for Open Source applications. It is certainly untrue to say there is no support for Open Source Software. Indeed if you know where to look, the support can be better than for Closed Source and you can choose to have free support like these You Tube videos or you can buy support and training from Open Source specialist companies. [125]

Inkscape [126] is a very capable graphic design program enjoying increasing take up in schools. It is suitable for use in primary schools but powerful enough to produce professional quality graphic images like this.



0 de 11 nós selecionados. Clique, clique + shift ou arraste ao redor dos objetos para sel..

Downloading and installing inkscape is simple. Go to the Inkscape [web site](#) [127] and on the left click on download and click on the latest stable release eg Inkscape-0.46.win32.exe. Once the file is downloaded double click and follow the installation as with any other Windows application. You can of course provide the .exe file to as many users as you want. This means all learners can take it home on a USB memory key and use it freely and legally at home supporting inclusion.

[Here](#) [128]is a quick You Tube video demonstrating the use of Inkscape.

Strategies for migrating to Open Source Software

Installing and using Audacity and Inkscape should demonstrate that obtaining good quality Open Source Software is practical and feasible. No license keys, no installation CDs, no purchase orders to process. But what of a broader strategy to [migrate](#) [129]as far as possible to Open Source? Here is a table with general classifications of software and some of the key issues associated with migration.

Type of Application	**Issues**
General productivity tools **eg Inkscape, Audacity, OpenOffice.org, GIMP, Firefox, Thunderbird**	**Interoperability and data exchange** Applications that generate data files should be able to export these files in open formats as we proprietary formats. This means that they will be able to exchange data with proprietary softwa applications. In practice, there are few problems with data exchange except in highly specialis where, for example, customised macros or programming have been used around MS Office a Sometimes interoperability is better supported by the Open Source applications eg OpenOffic

	better interoperability with MS Office than MS Works has with MS Office. Audacity supports the open industry standard svg vector format as well as png and jpg for exp images. Audacity supports uncompressed wav and aiff, the proprietary mp3 and open .ogg fo compressed audio and can import midi files.
	Learning how to use The learning curve for applications is very variable. Learning an entire Office suite like OpenO its details will take years but most people don't need that detail. If you have already learnt wha similar software such as MS Office most of that learning will transfer to OpenOffice.org and th comprehensive including fee user support groups. The learning required in a major upgrade i MS Office 2003 to MS Office 2007 is similar to the learning required to migrate from MS Offic OpenOffice.org 3.0. Inkscape is probably simpler to learn than most of the many proprietary v programs and is comparable to Xara Xtreme in its function. There are many video tutorials on Open Source applications.
	Installation and set up There is no real difference between installing any of the above applications on Windows comp proprietary equivalents. Network set ups and deployments will need specific considerations su permissions and security but no more so than with proprietary software. Setting up Firefox on likely to be more involved than installing Inkscape. On Linux installation can be a lot easier sin applications come pre-installed with the operating system and up dates and new applications line from a safe repository. This means no CDs to manage, no license or registry keys issues pay for updates or upgrades to later versions.
	People Some people will accept a new application discarding any previous tools they used, others wil at having to change. In short individuals vary a lot and people are probably more important in equation than the technologies. Investing in education is a good idea for people that believe in There are QCA accredited courses designed specifically for this purpose that are compatible National Curriculum. Since computer storage is now very inexpensive, you might make both o applications available alongside each other for a trial period. You might buy only sufficient lice proprietary version of an application that are needed just in case something doesn't work. The universal answer but try to think flexibly. It is not an all or nothing situation.
Web applications eg Moodle, Drupal, Joomla, Mahara, OpenGroupware, One to Zero	**Interoperability and data exchange** Firstly ensure that the application is accessible through a standards compliant Web Browser. I Firefox it almost certainly will be. Avoid any software that will only work with a particular brows your supplier for written assurance that your data will always be accessible through a browser supports W3C open standards. These applications will all use SQL databases to store informa pages to access it. In principle it should be possible to move data between SQL database app whether Open or Closed Source. If content files are based on Open Internet standards which Open Source applications, file exchange will be assured. In practice the complexity of the data involved in specialised aspects such as structuring lessons, is likely to result in a less than pe when moving information to another similar application. For example, a new Virtual Learning E with a specific classification structure for the "3 part lesson" will not necessarily have fields co this structure in the one used currently. Getting locked into proprietary web applications is a hi so as a minimum be sure that information can be exported in open formats.
	Learning how to use There are differences but once one application is mastered the methods are usually similar an

	the Web interface. Consistency of approach and user interface makes web applications gener learn. While specific applications will have particular procedures concepts such as forums, blo loads, RSS and making links are common to most similar applications whether running on Op software or Closed Source software.
	Installation and set up Setting up of web applications requires a server and a supporting software stack. This could b school or by a specialist provider. If you want an Open Source support stack it will probably be Apache, MySQL and PHP/Perl/Python. By out-sourcing to a server hosting company you can to know anything about any technicality other than using the application. You also enable your be made available to your community 24/7 with a faster internet connection than you are likely for at your site. If you use a closed source stack it will probably be MS IIS server running on a server and a proprietary database such as MS SQL server. If you do this check that there is n dependency on using other proprietary software applications or you could find yourself locked unanticipated costs. Many hosting providers offer a one-click installation service to their clients widely used CMS.
	People Since web applications are more recent and more similar in user interface they are more likely accepted than change from a mature and familiar desktop application. It therefore makes sens carefully before deploying any Closed Source and proprietary web applications because this is where Free and Open Source applications are having the biggest immediate impact and there to get locked in to a single vendor. Once people get used to a particular product it will be diffic to migrate. With hindsight, allowing large software monopolies to build on the desktop paradig has been a very expensive lesson. The difficulty in getting people to understand the collective change if it makes life in the short term even marginally more difficult for them is very apparen great deal of risk in getting locked into proprietary web products at a time when many of the ve products are unlikely to survive. Then what happens to your data?
Network management tools eg GNU/Linux Server, LTSP, Karoshi, iTalc, Webmin, gnome nettool	**Interoperability and data exchange** The main issues for interoperability and data exchange at the server level is when client mach dependent on proprietary protocols or there is dependency on Active Directory for file permiss policies. There are ways round these issues but it requires specialist technical knowledge. Wit services such as firewalls, web serving, proxy servers and such like there should be no real pr interoperability.
	Interoperability and data exchange The main issues for interoperability and data exchange at the server level are when client mac dependent on proprietary protocols or there is dependency on Active Directory for file permiss policies. There are ways round these issues but it requires specialist technical knowledge. Wit services such as firewalls, web serving, proxy servers and such like there should be no real pr interoperability.
	Learning how to use The learning required for Network management tools is the widest in range. Some aspects re no technical knowledge at all whereas others require a good knowledge of network security ar issues. This is an area where practical "investing in people" is important. If the technical suppo no access to learning about new technologies and approaches they will be unable to support o lower cost systems.

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');

	Installation and Setup Many of these tools eg Karoshi are designed to make it easy to set up and administer GNU/Li schools. The main issue with set up is understanding wider issues such as security and the ph takes to prepare software and machines. This is not necessarily any worse than for Close Sou software but it is an overhead in terms of migration.
	People The main aspect is technical expertise and willingness to keep up to date. If you have technici they are willing to implement Open Source solutions and have demonstrated ability to support a limited scale you have a valuable asset. Investing in further training is likely to bring real imp cost-benefit and efficiency to the organisation.

The exact strategy adopted will depend on the current level of expertise and readiness for change in the organisation. People are just as important a consideration as the technologies themselves and they need to be given time and incentives to learn. Adopting a limited number of Open Source applications on Windows such as Inkscape and Audacity is a simple way to start and requires no more knowledge or expertise than for any other similar application. Encouraging youngsters, who tend to be less change averse than adults, to participate practically is a good idea.

If you don't like the idea of installing software without trying it out first then try "PortableApps". Portable Apps are Open Source applications installed on a USB pen drive so you don't have to install anything on your computer to try out a whole range of software including Inkscape, Audacity, GIMP, OpenOffice.org etc. The principle is that

the applications run from a portable medium, usually a USB flash memory stick. Applications can be a bit slow to load because getting data from a pen drive is slower than getting it from the hard drive but once running the program is in the computer's memory chips so it runs just like normal. There is a good site to find out about and download portable applications here. [130]

Click on "download now" and put your USB memory in place and you are ready to go. The instructions are on the site. The concept is that portable apps installs an Open Source menu program onto your USB key from which you can then choose and add a wide range of programs to your key. The applications are installed on the pen drive and so any computer that has the pendrive inserted into a usb port effectively has the software instantly installed but not onto its hard drive. Take out the pen drive and the applications are uninstalled.

Click on "applications" on the top menu bar on the site for a full list of available applications. All this software is Open Source so it is perfectly legal to copy it to hundreds of USB keys. Every child in the school could have one to use at home and at school contributing to social inclusion and bridging the digital divide. Since all the applications move with the USB key, you can use them on any computer without having to install anything and without leaving any code behind when you finish.

If you click here [131], you can download an entire operating system and all the general productivity applications you are likely to need in a school entirely for free. You get free automatic updates directly from the internet, no viruses and no spyware. You can add new applications from a vast and growing on-line library again all for free. If you are nervous about installing the software on the hard disc of your computer, you can simply run everything directly from a CD without having to install anything permanently. You down load an entire CD image of software that has already been installed so there is no installation to do. Just run it from the CD in the same way as portable apps run from a USB key. If you like it you can install the software from the CD to your hard drive either as a replacement for your current operating system or to run alongside so you can choose which to use. This is the flexibility enabled by Open Source licensing.

Costing change

It is very difficult to cost [132] an entire shift from Closed to Open Source technologies and probably impossible to do so with meaningful precision. On the other hand, it is very easy to work out the cost benefit of a limited change. For example, what would it cost to provide all learners in your organisation with a proprietary commercially licensed graphic design program so that they could use it on any computer at school or at home? What will it cost to distribute Inkscape to the same extent? In simple cases like this you can control the variables and work out the cost-benefit. With an entire migration you would have to put a value to not having to cope with viruses and spyware and then try and guess if this would always be the case and then weigh this against not running a piece of software one HOD thought vital to his department or similar indeterminate variables. How would you cost issues of ethics or the value of learner participation in an open source community inspired by using a particular open source application? Total Cost of Ownership (TCO) [133] is very popular with those with a vested interest in maintaining the status quo simply because it provides a seemingly rational reason to delay change when in fact it is never going to give a rational result in anything but trivially simple circumstances. Despite years of debate on the subject there is no definitive TCO study and there is never likely to be.

On this basis, if saving money is your main goal, start with things that will definitely do that and keep things simple. Don't get bogged down in grandiose TCO arguments. If you get to the position where most applications are web based or Open Source, you can consider migrating the lower level infrastructure such as operating systems because dependency on running application on Windows is massively reduced.

If you really want to get an indicative idea of the potential savings from a more major change, work out how many PCs you are likely to buy over the next 5 years and cost £70 for each for Windows licences. Work out your anti-virus costs and any other support licence costs for network software. Work out any administrative overhead related to license keys and auditing software to be sure licensing is legal. Then add any other license fees that can be eliminated through using Open Source alternatives. There are further grey areas such as will you save on hardware by using less resource hungry software? What is the main cost involved to weigh against these savings? Some would argue that there isn't one because training should be considered an investment rather than a cost, especially in an education environment committed to investors in people. Education, training and knowledge are the main requirements for migration and there is no systematic evidence to the contrary.

Migration Strategy Summarised

1. Learn about Open Source software through simple practical experience. Try out some simple applications.
2. Ensure that the people in your organisation are prepared - there are QCA accredited vocationally related qualifications in Open Systems available from The Learning Machine Ltd [134] designed to support children and their teachers learning about the issues presented here.
3. Work out whether your organisational culture puts a value on the ethical and community aspects of Open Source.
4. Analyse the tasks performed using computers in your organisation and classify each of them by how easy it would be to migrate to an Open Source Application. What would be the implications?
5. Start with the easiest applications in your circumstances and ensure your first encounters are successful. Give your users a good experience.
6. Cost limited change and go first to those things that cost very little to do but clearly have the capacity to save the most money.

Digital Inclusion

A key purpose of Open Systems including the deployment of Open Source software in the public sector is digital inclusion.

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');

A draft assessment unit on Digital Inclusion (QCF level 2, EQF level 3) [135]

Summary

1. Open Source Software is software that is licensed so that users have the freedom to
   - Run the software for any purpose
   - Study and modify the software
   - Copy the software
   - Improve the software
2. In order to make all these freedoms effective the source code of the software must be freely available, hence the name Open Source. Some software is available free of charge, but without supporting all of the freedoms. It is referred to as free as in free beer rather than free as in freedom.
3. The freedoms reduce risk associated with a range of "closed source" software issues including lock-in to particular products, reduced competition, user inconvenience in managing licences, environmentally damaging packaging and commercial interests over-riding those of the customer.
4. The internet has enabled Open Source software to grow from an ethical concept to the biggest change taking place in the way software is produced and licensed since the nineteen eighties.
5. Social systems and the law have not kept pace with technological change giving rise to micro-political conflicts between different interest groups in the area of copyright and patents in the field of digital technologies.
6. The synergies between Open Source Software and Open Content development models and licensing, the use of Open Systems and Web 2.0 technologies provide massive potential to improve learning.
7. If technological education is to reflect true technological literacy, an understanding of the issues surrounding Open Source Software, Open Content and Open Systems is essential.
8. Learning in the future will reflect technological development and to remain relevant, schools need strategies that enable them to support these new ways of learning.
9. Return on Investment is a more practically useful consideration for managing change than Total Cost of Ownership.
10. Expenditure on training and learning how to change is an investment that will transfer beyond simple technological savings.

Useful links

Here are some quick links which may help readers explore and find free and open-sorce software to ease and improve their study or work. The list is in no way extensive, it is rather a hint on where one may start searching for answers and solutions which technology, and the work of millions of people around the globe, have already made available.

CMS Matrix is a content-management systems comparison tool. Very useful for those who have established the need to use a CMS but still have not decided which particular software package to use.

http://www.cmsmatrix.org/ [136]

Wikiversity is a Wikimedia Foundation project devoted to learning resources, learning projects, and research for use in all levels, types, and styles of education from pre-school to university, including professional training and informal learning. A fine helper tool for both learners and trainers.

http://www.wikiversity.org/ [137]

Wikimedia Commons is a large and ever increasing database of freely usable media files. A nice place to find graphical material for any website or project.

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');

http://commons.wikimedia.org/ [138]

SourceForge is the world's largest open source software development web site and also the largest collection of open source tools and applications. Software projects are ordered in tree-structured categories.

http://sourceforge.net/ [139]

Teaching Materials Catalogue contains many useful materials on teaching open source.

http://teachingopensource.org/index.php/Teaching_Materials_Catalogue [140]

Source URL: https://theingots.org/community/FOSSv2.0

Links
[1] http://bit.ly/tRmgeK
[2] http://en.wikipedia.org/wiki/Software_license
[3] http://weblog.infoworld.com/openresource/archives/2007/09/gartner_ignorin.html
[4] http://www.youtube.com/watch?v=UnFj2_yFb7M
[5] http://www.youtube.com/watch?v=ihxGJueWb-I&amp;feature=related
[6] http://www.google.com/
[7] http://code.google.com/soc/2008/
[8] http://news.cnet.com/2100-1001-249750.html
[9] http://www.sun.com/
[10] http://www.novell.com/home/index.html
[11] http://www.canonical.com/
[12] http://en.wikipedia.org/wiki/Hacker_%28free_and_open_source_software%29
[13] http://www.theregister.co.uk/2008/07/25/microsoft_gpl/
[14] http://www.theinquirer.net/gb/inquirer/news/2008/08/05/microsoft-menaced-open-source
[15] http://www.apache.org/
[16] http://www.youtube.com/watch?v=mi4vjhc8nDE&amp;feature=related
[17] https://theingots.org/community/../../www.wikipedia.org
[18] http://yupnet.org/zittrain/archives/16
[19] http://en.wikipedia.org/wiki/Jonathan_Zittrain
[20] http://en.wikipedia.org/wiki/Oxford_Internet_Institute
[21] http://en.wikipedia.org/wiki/Berkman_Center_for_Internet_%26_Society
[22] http://en.wikipedia.org/wiki/Jimmy_Wales
[23] http://en.wikipedia.org/wiki/Larry_Sanger
[24] http://en.wikipedia.org/wiki/Wikipedia
[25] http://wikimediafoundation.org/wiki/Home
[26] http://www.dcsf.gov.uk/rsgateway/DB/TIM/m002012/index.shtml
[27] http://www.catb.org/%7Eesr/writings/cathedral-bazaar/cathedral-bazaar/index.html#catbmain
[28] http://en.wikipedia.org/wiki/Reliability_of_Wikipedia
[29] http://en.wikipedia.org/wiki/Solution_stack
[30] http://en.wikipedia.org/wiki/MediaWiki
[31] http://en.wikipedia.org/wiki/Free_software
[32] http://en.wikipedia.org/wiki/Open_source_software
[33] http://en.wikipedia.org/wiki/Wiki_software
[34] http://en.wikipedia.org/wiki/PHP
[35] http://en.wikipedia.org/wiki/MySQL
[36] http://www.gnu.org/
[37] http://www.squid-cache.org/
[38] https://theingots.org/community/../../www.schoolforge.org.uk
[39] http://limulus.wordpress.com/2007/08/28/the-myth-of-intellectual-property/
[40] http://www.youtube.com/watch?v=OGg8A2zfWKg
[41]

ttp://technology.timesonline.co.uk/tol/news/tech_and_web/article3874599.ece?openComment=true
[42] http://www.youtube.com/watch?v=T0QJmmdw3b0
[43] http://www.facebook.com/
[44] http://www.myspace.com/
[45] https://theingots.org/community/../421
[46] http://en.wikipedia.org/wiki/Software
[47] http://wiki.python.org/moin/SimplePrograms
[48] http://www.tekmom.com/buzzwords/binaryalphabet.html
[49] http://www.youtube.com/watch?v=LERFIwHaApc
[50] http://www.theglobalist.com/globalicons/syndication/sample.htm
[51] http://blogs.zdnet.com/open-source/?p=2594
[52] http://searchdatacenter.techtarget.com/sDefinition/0,,sid80_gci212472,00.html
[53] http://www.opensource.org/
[54] http://www.riehle.org/computer-science/research/2007/computer-2007-article.html
[55] http://www.youtube.com/watch?v=n7g-XS5Q3mc
[56] http://www.moodle.org/
[57] http://www.ipo.gov.uk/whatis.htm
[58] http://www.youtube.com/watch?v=INnQjctMAnk&amp;feature=related
[59] http://en.wikipedia.org/wiki/Intellectual_property
[60] http://www.computerworld.com/continuing_coverage/000/001/000/continuing_coverage_000001008_primary_article.jsp
[61] http://www.w3.org/People/Berners-Lee/
[62] http://en.wikipedia.org/wiki/HTML
[63] http://www.fsf.org/licensing/essays/free-sw.html
[64] http://www.youtube.com/watch?v=wMspaCHsNec
[65] http://www.informationweek.com/news/software/enterpriseapps/showArticle.jhtml?articleID=209903394
[66] http://opensource.wikia.com/wiki/Vendor_lock-in
[67] http://searchcio.techtarget.com/sDefinition/0,,sid182_gci214270,00.html
[68] http://www.youtube.com/watch?v=sqMR6pe0N_g&amp;feature=related
[69] http://www.dwheeler.com/essays/open-standards-open-source.html
[70] http://www.openoffice.org/
[71] http://www.referenceforbusiness.com/management/Int-Loc/International-Organization-for-Standardization.html
[72] http://en.wikipedia.org/wiki/Vendor_lock-in
[73] http://en.wikipedia.org/wiki/File_format
[74] http://en.wikipedia.org/wiki/Protocol_%28computing%29
[75] http://free.financialmail.co.za/projects07/sa2008/usa.htm
[76] http://aseigo.blogspot.com/2008/04/deploying-kde-to-52-million-young.html
[77] http://www.freebsd.org/
[78] http://www.sun.com/software/solaris/index.jsp
[79] http://www.unix.org/what_is_unix/history_timeline.html
[80] http://en.wikipedia.org/wiki/BSD_license
[81] http://store.apple.com/us/product/MB427Z/A?mco=47638CE4&amp;fnode=home%2Fshop_mac%2Fsoftware%2Fapple
[82] http://www.gnu.org/licenses/gpl.html
[83] http://www.opensolaris.com/
[84] http://en.wikipedia.org/wiki/CDDL
[85] http://www.webopedia.com/TERM/o/operating_system.html
[86] http://voip.about.com/od/voipbasics/g/whatisserver.htm
[87] http://news.cnet.com/2100-1001-275155.html
[88] http://www.yolinux.com/TUTORIALS/LinuxTutorialWebSiteConfig.html
[89] http://www.aboutdebian.com/proxy.htm
[90] http://www.fs-security.com/
[91] http://www.linuxworld.com/podcasts/linux/2008/011008-linuxcast.html
[92] http://www.zimbra.com/about/
[93] http://www.ltsp.org/
[94] http://en.wikipedia.org/wiki/Porting
[95] http://www.mozilla-europe.org/en/firefox/

[96] http://clagnut.com/blog/205/
[97] http://www.opera.com/
[98] http://www.apple.com/safari/
[99] http://www.konqueror.org/
[100] http://www.youtube.com/watch?v=55yMCYsG7o0
[101] http://www.adobe.com/products/acrobat/adobepdf.html
[102] http://filext.com/file-extension/DOCX
[103] http://www.microsoft.com/PRODUCTS/works/default.mspx
[104] http://redmondmag.com/news/article.asp?EditorialsID=9882
[105] http://www.abisource.com/download/
[106] http://www.gnome.org/projects/gnumeric/downloads.shtml
[107] http://en.wikipedia.org/wiki/Mind_map
[108] http://freemind.sourceforge.net/
[109] http://vue.tufts.edu/
[110] http://www.tufts.edu/
[111] http://www.opensource.org/licenses/ecl2.php
[112] http://moodle.org/stats/
[113] http://en.wikipedia.org/wiki/De_facto
[114] http://drupal.org/about
[115] http://www.qems.org.uk/
[116] https://theingots.org/community/../../blog/1714
[117] http://www.joomla.org
[118] http://www.mahara.org/
[119] http://cterfile.ed.uiuc.edu/mahara/view/view.php?id=327
[120] https://theingots.org/community/../2487
[121] http://www.wikipedia.org/
[122] http://www.youtube.com/watch?v=7Q25-S7jzgs
[123] http://audacity.sourceforge.net/
[124] http://www.youtube.com/watch?v=M1IqWoWu8gU&amp;feature=related
[125] http://www.opensourceconsortium.org/
[126] http://www.inkscape.org/
[127] https://theingots.org/community/../../www.inkscape.org
[128] https://theingots.org/community/../../a%2520wide%2520range%2520of%2520programs
[129] http://open-source.gbdirect.co.uk/migration/
[130] http://portableapps.com/
[131] http://www.ubuntu.com/
[132] http://www.serverwatch.com/tutorials/article.php/3529871
[133] http://www.computerworld.com/managementtopics/roi/story/0,10801,76240,00.html
[134] https://theingots.org/community/../../www.theINGOTs.org
[135] https://theingots.org/community/FOSSv2.0
[136] http://www.cmsmatrix.org/
[137] http://www.wikiversity.org/
[138] http://commons.wikimedia.org/
[139] http://sourceforge.net/
[140] http://teachingopensource.org/index.php/Teaching_Materials_Catalogue

(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){ (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o), m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m) })(window,document,'script','//www.google-analytics.com/analytics.js','ga'); ga('create', 'UA-46896377-2', 'auto'); ga('send', 'pageview');