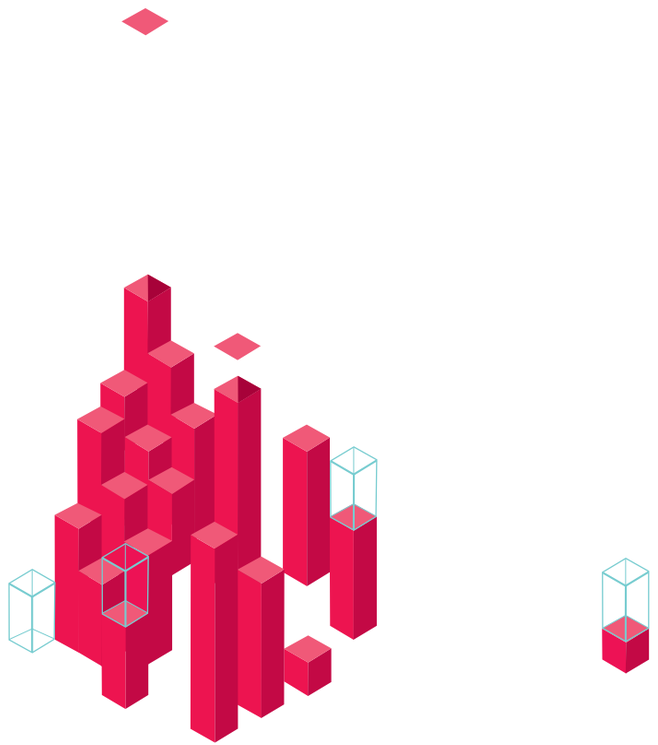
Futurelab
1 Canons Road
Harbourside
Bristol BS1 5UH
United Kingdom

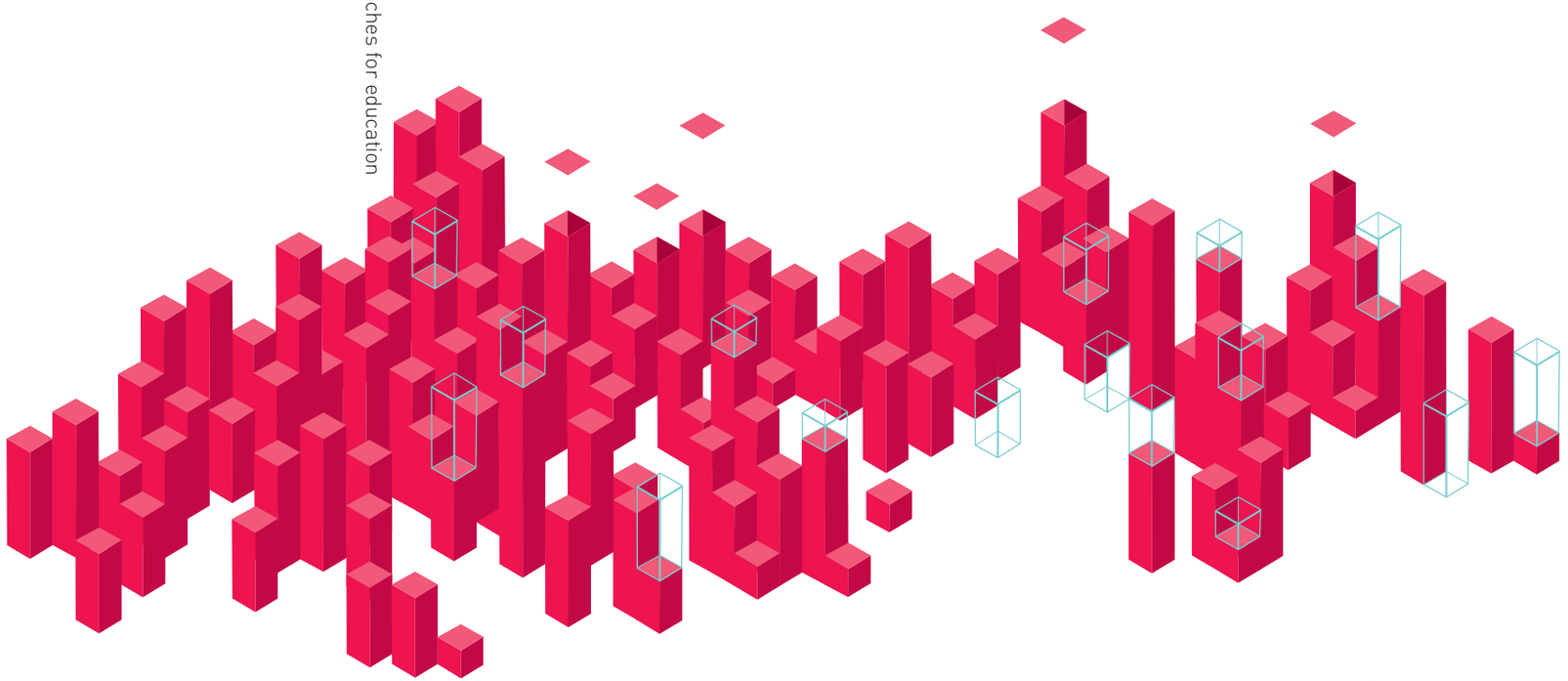
tel +44 (0)117 915 8200
fax +44 (0)117 915 8201
info@futurelab.org.uk
www.futurelab.org.uk



futurelab

OPENING EDUCATION

The potential of open source approaches for education



futurelab

By Seb Bacon and Teresa Dillon

Contents

Glossary

Given the acronyms and technical terminology that abound within the FLOSS community, this glossary is intended to provide a quick guide to the language used within this paper. The explanations provided have been abbreviated from various sources such as the North Regional Educational Laboratory[1] and the Australian government website[2].

For the purpose of this review we are using the 'catch-all' phrase Free, Libre and Open Source Software (FLOSS) as it encompasses notions of free and open software and takes account of recent developments in this field. Other acronyms used interchangeably with FLOSS and which refer to a similar approach to software development include Open Source (OS), Open Source Software (OSS) and Free Software. An overview of these terms is included below.

Apache a web server platform for hosting web pages developed and released as open source software

Backend the servers and other hardware that support a network; contrasted with frontend, which may refer to a web browser or other desktop software

BSD (Berkeley Software Distribution) a version of the UNIX operating system that exists in both open source and proprietary formats, comparable to Linux (FLOSS operating system). Also a model licence for FLOSS developed by the University of California Berkeley, originally for the BSD operating system

Binary the ones and zeros of a computer program as read by a computer. This is 'compiled' from human readable source code

Compatibility how well a program works with a different program, especially sharing files. Similar to interoperability

Copyleft an adaptation of copyright that aims to preserve the right of users to copy and change a work in perpetuity

Copyright the licence that protects a creator's right to control copies and changes to a work

Free software computer program with accessible source code that typically uses copyleft to prevent future closed source versions of the software. Often confused with gratis software

Free Software Foundation (FSF) founded by Richard Stallman to promote the idea and practice of free software

Freeware software distributed at little or no cost, but often without FLOSS licences or source code

GNU the ongoing project to develop a complete UNIX-style operating system that runs only on free software

GNU Image Manipulation Program (GIMP) an open source image editor. Very similar to Adobe Photoshop

GNOME one of the leading projects to create a graphic user interface for Linux

GPL the GNU Public Licence: a model open source licence that aims to protect the rights and freedoms of the user in perpetuity

Gratis software software of any type that is given away free of charge.

Internet a global network of connected computers. The internet and FLOSS are interdependent

Interoperability how well a program works with another program, especially different operating systems

KDE one of the leading projects to create a graphic user interface for Linux

Kernel the heart of an operating system. All software must run on top of a kernel, which provides safe access to system hardware and software

Licence the terms of use for a specific program. A software licence is a legal document since it formally restricts the rights of the user

Linux the leading and most successful FLOSS kernel. Also used more generally to refer to any operating system that uses the Linux kernel

Lock-in when users can't choose a new company's software because they have invested too much time or have too much data in their current software, and the change is too costly or otherwise impossible

Mozilla an open source web browser; the Mozilla Public Licence (MPL) is a model licence for open source software

Open source any program whose source code is made available for use or modification as users or other developers see fit

Glossary cont.

OSS (Open Source Software) any computer program with accessible source code. Anyone is legally and technically able to change and/or redistribute the software. The official open source definition encompasses free software, but also software whose source code can be included in future closed source software

OpenOffice.org a FLOSS office suite, includes word processing, spreadsheets, and slideshows. Very similar to Microsoft Office

OS (Operating System) the essential software to control both the hardware and other software of a computer. It consists of a kernel plus supporting programs

OSI (Open Source Initiative) the non-profit organisation behind the open source movement

Proprietary software software without accessible source code. Sometimes called 'closed source' or commercial software

Server any computer that provides a service to other computers in the network

Shared source a Microsoft initiative to make some of its software transparent to some users. This is not open source because only some people have access to the source code and they can't make changes or copies

Source code The human readable language a program is written in, before it is compiled to binary. A programmer needs to source code to change a program

Thin client a minimalist workstation connected to a server

Total cost of ownership (TCO) the total price in money, time and resources of owning and using software including purchase price, cost of upgrades, maintenance and technical support and training (or re-training)

Transparent anyone with sufficient skill can see how a program works

Foreword

Opening Education is a new series of publications from Futurelab. As its name suggests, in this series we hope to open up areas for debate, to provoke, to challenge, to stimulate new visions for education. At the same time, we hope to literally 'open up' education, to not only bring together ideas from educational practice and research but also to draw on the fields of creative arts, media and technical innovation. Finally, in the ideas we present we hope to 'open up' the walls of the educational institution – to present models of learning that show how we can create connections between learners in different settings, how we can enable collaboration between different organisations and institutions, how we can make links between different approaches to and forms of learning.

The publications in this series are intended to act as a complement to our existing resources. Where our literature reviews provide a survey of existing research in the field and our handbooks provide practical overviews and guidance on implementing new approaches, these Opening Education publications offer a space to think, to challenge, to open up new ideas that may not yet be ready for 'implementation' or rigorous research. They are the early warning systems, the 'canaries' down the mine of educational strategies and practice.

Research into innovation in industry and commerce suggests that having a superfluity of ideas is essential for growth and development – education is no different. We need to have a surplus of ideas and strategies and visions and plans so that we have enough to draw on when we face the serious challenges for education that social, economic and technical change presents us with. Not all ideas will become a reality, not all ideas will survive in the form in which they were presented, but what cannot be denied is that education and educators need to know that there is scope to dream, to think about new approaches and different ways of doing things; to know that the ways we do things now will not be always and forever the same.

So, we're sending these ideas out to see if they live or die in the light of wider debate. They are experimental and exploratory, both in their ideas and, in subsequent publications, in the forms in which we publish – they won't all look the same, feel the same, say the same thing. They will all, however, attempt to open up a new area for debate and for action and we look forward to hearing from you and working with you to determine their fate.

Keri Facer
Research Director

Executive summary

Free, Libre, Open Source Software (FLOSS) refers to any software distributed under a licence that allows users to change or share the software source code. The three most important characteristics of FLOSS are that:

- it allows free (unrestricted) redistribution
- the source code is available at minimal cost
- derived works may be redistributed under similar non-restrictive terms.

These principles have emerged from a long and complex history that is intricately bound up with early development practices around mainframe computers, debates over the nature of knowledge and information, and the emergence of home PCs and the commercial software market. FLOSS principles have, from these origins, inspired new approaches to copyright (such as Creative Commons) and have come to inform a cultural phenomenon that is underpinned by technological development with the aim of contributing to the public good.

Futurelab's interest in this area stems from the belief that FLOSS provides an example of peer-production which is driven by collaborative, social modes of interaction and knowledge exchange. This paper discusses some of the potential ways in which the approaches that characterise FLOSS might be applied in educational contexts; specifically, whether they can act as a model for education in:

- offering new approaches to teaching and learning, specifically enabling personalised learning and enhanced learner voice
- enabling knowledge sharing and collaboration between teachers
- overcoming structural divides between developers of educational software and its users.

The paper does not discuss the pros and cons of schools adopting open source software systems, but examines the possibilities opened up by pursuing an open source philosophy.

FLOSS approaches enable the creation of distributed collaborative networks of people working together to solve problems. This might provide a powerful way of thinking about how learners might work together, within or across schools, to generate new knowledge and practice of relevance to them. It offers the opportunity for learners to identify small tractable problems and together create ultimately significant contributions to knowledge.

With regard to teachers, FLOSS approaches provide an insight into how knowledge can be shared, modified and adapted across the teaching profession and in different contexts. We could conceive of networks of teachers and researchers working together on different educational challenges to create new approaches that are open to and usable by all. These approaches raise questions about the growing trend towards copyrighting and selling of teaching strategies, curricula and schemes of work.

It is also possible to conceive of young people or teachers working together as programmers to create new resources and tools that are of relevance to them in supporting their own learning. These approaches go beyond the traditional distinction between 'users' and 'producers' of educational resources, instead, they offer models of innovation in which these communities are intermingled, the notion of ownership is changed and the economical model of cost and reward is reworked. These new hybrid models of innovation that FLOSS exemplifies require us to ask what models of ownership we might need to develop; what mechanisms might need to be put in place to encourage exchange between sectors; what role users of educational resources might play in the creation of resources; and what business models would need developing to allow further exploration in this area.

FLOSS is more than software: it is of relevance to our understanding of how people learn and produce knowledge; of how communities collaborate and work to solve problems; and of how innovative practices emerge. As a movement it raises a provocative set of challenges for educators and developers of educational resources.

1 Introduction

FLOSS defies easy description. The research literature describes it variously as a phenomenon; a community of practice (Edwards 2001, p442; Tuomi 2005); a scene (Lehmann 2004); an approach to licensing (Perens 1999); an economics model (Khalak 2000; Lerner and Tirole 2000); a value and social system (Lessig 2004; Stallman 1992); and a hybrid model of innovation (Lin forthcoming). These diverse interpretations indicate the multidimensional aspects of FLOSS and its interdisciplinary relevance to a wide number of domains including governance, policy, culture and industry.

All the above characteristics apply to FLOSS and in this respect it is a potentially powerful example of new modes of production which challenge received wisdom about the most effective means of generating and sharing knowledge. It raises an interesting set of questions for education, particularly for the development of education systems which have as their goal the nurturing of young people able to react flexibly and creatively to the world around them.

Despite the success of FLOSS over the last ten years (for example, FLOSS servers such as Apache run over 70%[3] of the world wide web), its relevance as an approach to innovation and knowledge sharing is not always visible or easily accessible to educators. FLOSS has been, to date, a closed community riddled with jargon, often led by the technically savvy and with its origins buried deep within varying historical accounts scattered across the world wide web.

This discussion paper provides educators and those working to create educational technologies with a historical overview of the development of FLOSS and the key contemporary debates in the area. It aims to give an understanding of some of the potential ways in which the approaches which characterise the FLOSS movement might be applied in educational contexts as a model for young people's knowledge creation, teachers' community building and innovation in the development of educational resources.

The paper draws links between FLOSS approaches and current educational agendas in an attempt to make the relevance of the FLOSS approach to

education more accessible and visible. It discusses the potential application of this approach in three key areas:

- new approaches to teaching and learning, considering young people as knowledge builders and creators
- supporting teachers within a community of practice
- the development of new models of innovation and software development in education.

FLOSS approaches are only beginning to be explored in education and at the present time the research literature is partial and fragmented. As a result, it is not possible to make definitive statements about the most beneficial applications of FLOSS approaches in education. We do believe, however, that this is a debate that is worth exploring further and that by doing so, we can raise a set of provocative and powerful questions that are relevant to debates in education today.

These approaches may have relevance specifically to discussions relating to:

- the personalisation of education (the degree to which learners are able to shape educational experiences to meet their own needs)
- the development of the teaching workforce (the degree to which ongoing collaborative knowledge sharing is likely to be important in the future)
- the design of digital resources for learning (the degree to which learners, teachers and educators are able to work alongside programmers to create resources tailored more specifically to their educational needs).

At a time of rapid change in our relationship to knowledge and our views of teaching and learning in 'information societies' it is timely to consider how FLOSS approaches may be of relevance to educational policy and practice. The bottom-up organisation, where distributed self-motivated individuals creatively collaborate and work together on shared problems, has relevance both in terms of the creation of digital technologies we use for education, and as an approach that could be adopted as part of the teaching and learning process.

2 Some key terms

2.1 Open source

The term 'open source' refers to what has been described as the "bill of rights for the computer user" (Perens 1999). In other words, the licence that describes how the software should be distributed, and the rights you have as a user to change or share the software source code.

Source code (often called **source** or **code**) refers to the human-readable expression of the instructions required to make a computer carry out tasks. When a developer creates software, they write it in a programming language: something that resembles human language in superficial ways, with its own grammar, verbs, nouns and adjectives. The human-readable code runs through a process called **compilation** which converts it into binary, effectively a long string of zeros and ones, which the computer can then use to perform the necessary actions.

Code, therefore, shares some superficial similarities with human language. As a result it is considered by many working in this field as a form of expression; software that accomplishes exactly the same result (say, sorting a pack of cards) may theoretically be written in an infinite number of ways, depending on the particular intuitive or artistic preferences of the programmer. It is not by chance, therefore, that many of the legal debates around software are couched in terms of artistic copyright or free speech, and it is in this conception of software, as expression rather than as a form of machine, that the origins of FLOSS software lie.

2.2 Free software

The term 'free' is applied to software in two senses. In the first instance it can mean software that has a zero purchase price. In the second instance it is software that is not restricted by licences or has a licence that ensures its freedom. It is the latter meaning that is applied throughout this review, with the term 'gratis' used to describe the former.

2.3 FLOSS

Early approaches to FLOSS, in particular the Free Software movement, emphasised the rights and freedoms of the expressive programmer. In 1998

the Open Source Initiative[4] (OSI) was established, with the goal to develop a coherent understanding of FLOSS and the legal licences under which it operates for a wider business-oriented community. The following ten criteria provide a snapshot of how FLOSS is most commonly understood and defined. Of the ten criteria[5], the first three are the most important:

a. **Free to redistribute**

b. **Open source code** – the program must include source code, and must allow distribution in source code as well as compiled form. If some form of a product is not distributed with source code, there must be a well-publicised means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the internet without charge.

c. **Derived works** – the licence must allow modifications and derived works, and must allow them to be distributed under the same terms as the licence of the original software.

d. **Integrity of the author's source code** – ensures that developers use the source code in a productive way. Ensures that the originator's coder reputation is protected.

e. **No discrimination against persons or groups** – ensures maximum diversity to avoid 'locking' anybody out of the process.

f. **No discrimination against fields of endeavour** – the licence must not restrict anyone from making use of the program in a specific field of endeavour. Also encourages commercial users to join the community, not feel excluded from it.

g. **Distribution of licence** – rights attached to the program must apply to all who use it, without need for additional licences.

h. **Licence must not be specific to a product** – the rights attached to the program must not depend on the program's being part of a particular software distribution.

i. **Licence must not restrict other software** – the licence must not place restrictions on other software that is distributed along with the licensed software.

j. **Licence must be technology-neutral** – no provision of the licence may be predicated on any individual technology or style of interface.

2.4 Commons-based peer-production (CBPP)

There are many approaches to FLOSS which have resulted from a mesh of control factors and choices of licence, governance and artisan models. Underlying all these different approaches, however, is a method known as 'commons-based peer-production', or CBPP (Philips 2005)[6]. Within this review we have adopted a similar understanding of FLOSS, considering it as an exemplar of CBPP.

The term was first coined by Professor Yochai Benkler and refers to the collaborative efforts and outcomes of a large number of people working incrementally on a problem or artifact without being organised on either a market-based, managerial or hierarchical model[7]. Benkler, like many others (Lessig 2002, 2004; Stallman 1992; Boyle 2001), considers CBPP as an emerging "third model of production" (Benkler 2002), which maximises human creativity through the use of ubiquitous computer communication networks (eg mobile telephony, internet). According to Benkler such networks are bringing about a dramatic change in the scope, scale and efficacy of peer production. One key example of this is FLOSS, where the internet has been central to the movement's development.

Drawing on the work of Benkler, the following key characteristics of CBPP which underlie the FLOSS approach to software production, have been derived:

- **Motivation:** individuals are not necessarily motivated by money when working on a project; many are happy to be involved for other reasons such as recognition, public good and so forth on a project.
- **Chunking:** many individuals work incrementally and asynchronously on small aspects of a problem. Problems are usually fine-grained so they do not take much time but are intrinsically motivating to those working on them.
- **Multi-disciplinarity:** peer-production projects usually include a large number of contributors, with varying but complementary expertise, working on fine-grained aspects of the same problem.
- **Integration:** successful peer-production enterprises are usually low-cost, which maximises chances for successful integration.

2.5 Open standards

'Open standards' and 'open systems' are terms often used interchangeably with open source. The term 'open standards' refers to the detailed, descriptive overview of a process, protocol or format. This overview is formulated through stakeholder consensus, must be openly published and usually has no legal or intellectual property (IP) restrictions. Open standards are generally defined by groups within standards organisations.

There is greater synergy between FLOSS and open standards than between proprietary software and open standards. The general consensus is that open standards provide users with greater choice, open competition and accelerated progress, which in turn can lead to the commoditisation of a particular technology (Australian Government Report 2005). Open standards are useful for creating higher levels of interoperability and greater choice in component procurements, which leads to more competitive markets. However proprietary companies do not always choose open standards because that approach can reduce their opportunities to tie consumers to their own products. Moving from proprietary software with closed standards is difficult and gives rise to product lock-in.

2.6 Open systems

In the early days of computers, systems were expensive and decompartmentalised. Pressure came from users and from the computing industry to achieve interoperability and cost-effectiveness between different platforms sold by different vendors. One of the key outcomes of these pressures was the development of POSIX-capable systems (POSIX being an open standard) interconnected on open systems (the TCP/IP networks).

3 Evidence and resources in the field

There are multiple voices, at times contested and competing, that have influenced and continue to influence the development of FLOSS approaches. Given the commercial sensitivities in this area, much of the research to date on the potential of FLOSS in education or the public sector focuses primarily on FLOSS as a business model. There are comparatively few discussions of how FLOSS as an approach might have implications for educational approaches and philosophies (Becta 2004, 2005a; Moyle 2003; Staring, Titlestad and Gailis 2005; also refer to www.ossite.org).

It is important to note that FLOSS is still in its early stages. It is only in the last five years that major quantitative studies of the implications of FLOSS at a national and EU level have been undertaken (Becta 2005b). To date, there are no empirical studies which offer a reliable insight into the potential uses of FLOSS approaches as a basis for teaching and learning or the development of teacher communities, although there are parallels that might be drawn between FLOSS approaches and emerging pedagogies such as those advocated by adherents to constructivist and 'knowledge building' practices (Chawner 2005; Edwards 2001; Lin 2004; Stürmer 2002).

Academic research in FLOSS tends to be sporadic and driven predominately by individual researchers' questions and backgrounds. Consequently, one tends to find that researchers of law focus on legal issues, economics and production models; researchers from computing science focus on the code; while researchers for the social sciences, arts and education focus on new modes of collaboration and the implications this has for knowledge building[8].

The history of FLOSS is documented in a relatively small number of key sources. 'Hackers' by Steven Levy (Levy 1984; 1994) describes the cultural software engineering milieu which gave birth to FLOSS. It is compiled by Levy from both written accounts and hundreds of interviews with some of the most important figures in computing, including Bill Gates (author of Altair BASIC, which become the foundation stone of Microsoft), Professor Mariv Misky (who ran the MIT AI lab), hackers such as Richard Greenblatt (often cited as founding the hacker community), Steven Jobs (co-founder of Apple), and Richard Stallman (who founded the free software movement and has been a central figure in FLOSS).

Much of the primary material on FLOSS was for a long time only available from scattered locations across the internet, but in 1999 the most important first-hand accounts were collected in 'Open Source: Voices from the Open Source Revolution' (DiBona, Ockman and Stone 1999). 'Free as in Freedom: Richard Stallman's Crusade for Free Software' (Williams 2002) gives a compelling biographical account of the most important and controversial figure in the movement, Richard Stallman. The manifesto that launched FLOSS into the wider public consciousness, 'The Cathedral and the Bazaar', is available under a Creative Commons licence and is reproduced in many places across the internet.

The FLOSS community itself provides a valuable, if partisan, source of information. This form of descriptive, anecdotal, case-based material has been invaluable in raising the profile of the community, addressing critical debates and influencing practitioners to adopt FLOSS. Online forums, archives, FLOSS companies and key advocates in this field have utilised the internet to its full potential. What the novice reader has to quickly learn is how to distinguish between the informative and the evangelistic. Within the field of education, key online communities and resources are:

- Schoolforge[9], the international resource, web forum and archive
- Schoolforge UK[10], which is similar to the international branch, runs an online wiki of resources, information, support and training for the FLOSS education community in the UK
- SIGOSSEE[11], a pan-European project which runs conferences and holds regular meetings, runs a comprehensive weblog and projects; key issues the group tends to focus on are standardisation and interoperability
- FLOSS Posse[12] is a wiki focusing on EU and international discussions on FLOSS within education.

Recent years have begun to see an increase in government and public sector funded research into FLOSS models at UK, EU and international levels. Significant amongst these is a major international project FLOSSWorld[13], which aims to promote international collaboration between the EU and developing countries. The project will be the first to carry out research and

development on policy related to open source software at a global level. Another EU project FLOSSPOLS (at the University of Maastricht, The Netherlands) has provided valuable information on the uptake, uses and potential application of FLOSS within Europe[14].

Many EU countries are implementing policies in this area, which has led to a number of projects primarily investigating the potential costs of such initiatives (see for example Shankland 2003). Such government initiatives seem to be changing how software is procured in some countries, which has consequently led major commercial companies, such as Microsoft, to respond. One such outcome is that Microsoft started a programme to share the source code underlying Windows 2000, XP, Server 2003 and CE with some governments[15]. Another key area of government interest, particularly in countries such as South Africa, Brazil and Venezuela, is in the potential for FLOSS to challenge 'digital divides' and boost economies (see for example Kingstone 2005; Proffitt 2002).

Work carried out by Demos[16] in the UK or by government agencies such as the Joint Information Systems Committee (JISC)[17] and the British Educational Communications and Technology Agency (Becta)[18] is of specific relevance to the UK education sector, and discusses in particular the question of Total Cost of Ownership (TCO) in relation to FLOSS in education. The JISC website also provides a practical overview of the field, archiving and disseminating resources to its members as well as discussing in lay terms the key issues around the debate. The UK Office of Government Commerce (OGC) and the Office of the E-Envoy also provide resources and information.

From industry sources, the QuinetiQ report (Peeling and Satchell 2001) was one of the first of its kind in the UK, which focused on the 'Analysis of the Impact of Open Source Software'[19]. The report concludes that FLOSS highlights a "fundamental change in the software infrastructure marketplace, and is not a hype bubble that will burst". Evidence for this claim drew on market research which showed that Linux has become one of the fastest growing operating systems[20]. Discussing the growth of FLOSS, its benefits and issues for developers and users, future trends and implications, the report concludes that FLOSS offers a "new paradigm for funding software in communities-of-interest (eg Health and Education)", and recommends that the UK government should test the viability of the OSS approach to such software.

As noted, FLOSS is still quite a recent approach to software development and research into it is still within its infancy. As a result, much of the work to date consists of attempts to address the potential implications of FLOSS and the models of development and social interaction which underlie it. Much research remains to be done in proving or disproving the validity of different approaches, and this discussion paper is therefore presented in the spirit of generating debate and discussion, rather than advocating specific strategies in this area.

4 Historical overview

The following history provides readers new to the field with an outline map of some of the diverse and overlapping narratives which inter-relate in the history of the FLOSS movement. It is a history which interweaves legal issues and licences with programming practices, technological developments and new forms of collaboration and communication. All of these factors need to be considered in understanding the implications of FLOSS for educational practice.

In the early 1960s, before software development was a major industry, most code was created in an academic-style environment, albeit one often embedded in a commercial context. It was circulated around communities of experts, discussed, peer-reviewed, and built upon. Computers were vast physical objects the size of rooms and the cost of running programs was high.

Consequently, all software was developed in well-funded organisations, often in technologically advanced academic institutions in collaboration with the military, security forces and large commercial enterprises. As the number of software producers and consumers was too small to allow for a commercial software ecology, source code was not considered a valuable commodity. As it had marginal economic value, it was natural to consider source code a public good.

Many of the enthusiasts who understood how to program were more interested in solving problems than making money. In this environment, new software moved around quickly on magnetic tape, on punch cards or on paper printouts.

4.1 UNIX and the birth of the internet

In 1969, as part of their work with Bell Labs (a division of AT&T telephone company), software engineers Thompson and Richie developed a new operating system called UNIX (see Salus 1994). Since so few computers existed (all room-sized mainframes) UNIX wasn't conceived as having potential commercial benefit. In addition, anti-trust legislation forbade AT&T from entering the computer business[21], so UNIX maintained its academic-style culture of free circulation and peer-review (McGowan 2005).

The University of California at Berkeley started licensing UNIX from AT&T in 1975 and researchers developed the code further. Recent changes to the software were packaged and released under the name Berkeley System Distribution (BSD) to anyone who requested it. UNIX began to develop very fast until so much of the source code had been rewritten and extended that it was hard to see the BSD as the same as UNIX.

In 1984 AT&T entered the computer business (McGowan 2005) with UNIX. Initially, BSD was more successful, partly because it contained more desirable features, but the BSD project continued to share their work with the UNIX originators and this flow of innovation allowed UNIX to dominate a growing commercial marketplace. Berkeley focused on academic research and exploring the next generation of technological developments.

Under a contract with DARPA (United States Defence Advanced Research Projects Agency) the team at Berkeley added networking code to BSD so that it ran with other types of network. This part of BSD, known as TCP/IP[22], became the building block of the internet and allowed BSD (and consequently UNIX) users across the world to communicate using a single networking system. As the TCP/IP parts of BSD had been developed entirely without AT&T UNIX code, the BSD team decided to release those portions independently, under a very unrestrictive licence which stipulated that anyone could use and copy the code as long as they credited BSD.

This was the first instance of an open source licence and was crucial to the later development of the FLOSS movement and to the further spread of the internet; much of the world's internet software is derived from the original BSD TCP/IP and networking code, including some of the networking software inside Microsoft Windows.

Attention turned to the question of whether it might be possible to remove all AT&T code from BSD to create a standalone release completely free of restrictive licences. It was widely regarded as a nearly impossible feat, but in fact was made possible by an innovation enabled by BSD in the first place: the internet.

Keith Bostic[23], one of the creators of the BSD system, saw the potential of mass development of software using the internet whereby people who had never met each other could work together on a single project. The enthusiast culture that had grown up alongside BSD, combined with the unique collaborative potential of the internet, allowed the removal of nearly all of the AT&T code from BSD within two years (McKusick 1999).

The success of BSD, combined with how easily it could be copied and incorporated into commercial code, led to a dispute between the University of California at Berkeley and AT&T through its spin-off company USL (UNIX System Laboratories). The case was settled with a crucial stipulation: USL would not sue anyone using the settled version of BSD as the basis for their own system.

Today, there are four major variants of BSD derived from the original BSD code that are widely used, including Darwin (the basis of Apple Mac systems) and NetBSD (an open source project for building a BSD-based operating system that is both free and available for many platforms). These BSD licences are examples of 'permissive' licences that impose almost no conditions on what a user does with the software. This means that redistributors can use it for proprietary products, so derived works do not need to be open source.

4.2 The birth of GNU and copyleft

The freedom offered by the licensing arrangements to use and modify BSD code were not considered 'free' enough by some.

The hacker culture encompasses its own jargon, symbols and expressions, many of which explicitly oppose or subvert the 'mainstream' (defined in terms of 'normal' values and working habits) expressed in the proprietary software model to which the FLOSS paradigm is opposed[24]. The most relevant aspect of the culture to the emergence of FLOSS is the 'hacker ethic': the idea that information sharing is a powerful and positive good, and that hackers have an ethical duty to increase the public information commons. The ethic is often summed up in the phrase "information wants to be free" (Stallman 1992).

The most significant development that emerged from MIT's hacker culture was Richard Stallman's GNU project, in many ways a direct reaction to the increasing commoditisation of software. The explosion of the software market led to the closure of source code, which became a trade secret. Prior to this, hackers and others with programming knowledge were able to fix computing breakdowns because they had the ability to 'get into the engine' on the machine. Users of software like Microsoft's MS-DOS were no longer able to fix errors themselves.

Stallman's dictum that source code should be freely available began to take on a totemic quality and he decided to attempt to develop a complete UNIX-like operating system, to be called GNU[25]. In 1985, Stallman published the 'GNU Manifesto', which aimed to "knit the diffuse post-1980 community of hackers into a coherent social machine for achieving a single revolutionary purpose"[26]. He labelled his philosophy 'free software' and formed the Free Software Foundation[27] (FSF) as a vehicle for implementing his manifesto. In particular, Stallman developed the 'Four Freedoms of Free Software'[28]. These are:

- the freedom to use software however you wish (a freedom most programs give you)
- the freedom to change software to suit your needs
- the freedom to distribute the software to anyone else, and in doing so the freedom to 'help your neighbour'
- the freedom to distribute altered versions of the software, and in doing so cultivate a community based around the evolution of that software.

In order to preserve these freedoms permanently, Stallman came up with a novel and inverted way of using the legal system of copyright – by developing what is known as copyleft (DiBona et al 1999).

Copyright is a form of intellectual property that allows the licence holder the sole legal right to copy their works of original expression, for a defined period of time. It is designed to protect the creator's right to control copies and changes to a work, and by default copyright is usually used by the originator of a work to **restrict** how it is used. Stallman ingeniously used copyright to restrict peoples' rights to distribute the GNU software **unless** they agreed to distribute it under exactly the same terms as they received it. Where copyright protects the creator of the work, copyleft protects the right of users to copy and change a work in perpetuity. By creating a licence that protects the commons from private appropriation, the concept of copyleft gave power back to the user and made various other forms of FLOSS licences and its derivatives such as the Creative Commons possible (Lessig 2004)[29].

4.3 The growth of FLOSS

In 1981 IBM's personal home computer was launched and it became clear that software could also be viewed as a commodity. Companies, such as Microsoft, that specialised in particular variants of software started to appear. The exceptional skills of the hackers at the MIT AI Lab became more valuable and, as public funding of the lab tailed off, members began to work in other commercial and non-commercial contexts. Many set up their own spin-off companies, drawing on the fruits of their earlier research. Eventually the main research project at the lab, the Lisp machine[30], was taken completely into the proprietary domain, and the source code was no longer distributed as a public good (Williams 2002).

The internet provided a networked, global resource through which software could be shared. With the release of Stallman's GNU code and GPL licence over the net, other programmers began to build and extend it. One key addition was made in 1991 by a young Finnish computer science graduate, Linus Torvalds.

Torvalds wanted to adapt the GNU system to run as a fully functional operating system on his home PC. The part that was missing was the 'kernel', which he developed under the name Linux[31]. Three key factors converged to make his vision possible: he was able to use the GPL to ensure that his work would remain fully free; he was able to use the internet to distribute his code; and affordable home computers meant that development was in reach of people without access to expensive mainframe computers.

With Linux the phenomenon of free software took on a new characteristic. The movement had always been about collaboration, but typically this collaboration took the form of swapping ideas between individuals or small teams. The combination of circumstances that led to the fast spread and development of Linux was mainly characterised by the sheer size of the developer community.

In 1999, Eric Raymond, self-styled "tribal historian and resident ethnographer" of hacker culture, published 'The Cathedral and the Bazaar', an essay examining the importance of the 'scale' phenomenon to the successful development of Linux (Raymond 1999). Raymond was astonished to find, associated with Linux, a complete and usable system – he had been an enthusiastic user of free software for some years without ever seeing the movement create complete systems (the GNU operating system had been a decade in the making).

His analysis was that the 'old' model of software development, typified in projects like GNU, was top-down: run by a small number of key hackers, who acted as the gatekeepers to the source code. Contributors submitted their software changes to the gatekeepers, who then made a decision on whether or not to incorporate those changes into the central source code. Raymond (1999) compares this to the construction of medieval cathedrals.

By contrast, Linux had developed in a similar way to the nascent free market represented by the medieval bazaar, where order arose spontaneously when enough people met to exchange and socialise. In this account, Linux evolved in an organic, even chaotic manner. Developers exchanged and contributed source code without any direction from above. Software releases were frequent, and new features were tested by large numbers of people as soon as they had been sketched out. A process of natural selection eventually allowed the good ideas to thrive.

For Raymond, this insight is the key to the phenomenal success of Linux. By spreading the ideas and software as far as possible, an emergent, evolutionary process ensured the progression and quality of the software. Raymond coined the phrase "many eyes make bugs shallow" to express one advantage of the distributed FLOSS approach: when the source code is available to all, the software becomes more robust and secure. This phenomenon is counterintuitive to many from outside the FLOSS community: surely a piece of encryption software, for example, is far more secure if its internal details are effectively locked away? However, there is a wide consensus that exactly the opposite is true when it comes to cryptographic software, and many in the FLOSS world believe this same principle applies to all software.

4.4 Free versus open source

Raymond's account of the software culture that emerged from the early UNIX and MIT camps is informed by an anarcho-capitalist, market-oriented worldview. This is in strong contrast to Stallman's adherence to his understanding of the principle of freedom. Indeed, Stallman's 'viral' free software definition of FLOSS has been argued in America to be restrictive, anti-competitive, and even un-American, and has been subject to sustained challenge by Microsoft (Raymond 2001). Stallman, however, strongly refutes the charge of economic leftism, claiming that the four freedoms of the GNU manifesto encourage true competition rather than closed monopolies (Stallman 2002). Nevertheless, there is a strong suspicion across all FLOSS subcultures that commercial interests tend to co-opt the socially-oriented goals of the developers – a suspicion which is strongest amongst the GNU/FSF advocates.

Raymond formulated a plan to move the free software movement towards a more business-friendly arena, and took the position that in order to maintain and extend the success of the free software approach it was necessary to package it in a format palatable to business and the media. The phrase 'free software' was felt to be ambiguous: it confuses free (unhindered) with free

(gratis). The phrase 'open source' was picked to represent the movement, and a foundation was set up to certify licences as open source. Both BSD-style licences and GPL-style copyleft licences are open source in that they permit the programmer to read and modify the source code of a piece of software to which they apply. The ideas expressed in 'The Cathedral and the Bazaar' helped the software industry understand how free software could actually help them in achieving high quality, secure software.

The first high-profile open source convert was Netscape Communications, who had effectively lost the browser wars of the 1990s to Microsoft's Internet Explorer software. They decided their best strategy for preventing Microsoft from using its position to dominate the web was to release the entire source code for the Netscape Communicator web suite to the community. It took far longer than anyone imagined, but the seed they planted is now growing in the form of the Firefox web browser[32]. Other companies are also exploring the open source route: IBM is now an open source consulting firm and Intel is moving towards releasing its latest chip specifications under an open source-inspired licence[33].

4.5 Internal controversy - coining the FLOSS term

The two major camps of free software (on one side the FSF/GNU community voiced by Stallman, and on the other Raymond's OSI and the BSD communities) often engage in vigorous, sometimes acrimonious debate. In particular, Stallman argues that non-copyleft licences inherently undermine the principle of freedom in allowing third parties to co-opt and close the source code, while the BSD and OSI side claims that the restrictions of copyleft are themselves not commensurate with the goal of freedom. Others (Berry 2004; Collman and Hill 2004) have argued that the philosophy is politically agnostic, and analyse the competing interpretations as "iconic practices" which lend themselves to multiple meanings that belie the political inclinations of those who espouse them.

Table 1: Difference between free and open source software

	Free software culture	Open source software culture
Organisation	Free Software Foundation	Open Source Initiative
Figurehead	Richard Stallman	Eric Raymond
Favoured licence type	Gnu Public Licence: 'viral', ensuring openness of code in perpetuity	BSD-style licences: not placing any restrictions on use of software
Perceived benefits of FLOSS	Freedom of information	Better quality software
Values outside software	Leftist; communitarian; idealist	Libertarian; laissez-faire; pragmatist

At the present time, while acknowledging the difference between free and open source software, many people have chosen to use the encompassing term FLOSS. This is a compromise term but it encapsulates all possible meanings that have evolved from the free software movement: Free, Libre, Open Source Software.

4.6 Widening influence of FLOSS approaches

The development of a strong hacker subculture in parallel with the emergence of the proprietary software model from an initially heavily regulated sector was a major factor in the evolution of FLOSS, from a quasi-academic way of developing operating systems to a movement with global influence. Linux, for example, is the fastest-growing sector in the server

market (Gould, Barnett, Kimberly, Dowling and Hogan 2005) and has become a longstanding, major player in this area; and the arrival of software such as OpenOffice, and the web browser Firefox, has meant that the world of FLOSS has begun to penetrate the edges of the public consciousness and to challenge for the consumer software market (Fisher 2004).

The FLOSS movement is beginning to have implications outside the world of software. Despite the sometimes profound disagreements within the community, one issue has become a totem for the entire movement: that of patent law. As hackers see software as a form of expression, the application of patent law to software is seen as absurd, just as it would be ridiculous to patent the music of Beethoven. The anti-software patent position is widely held across all FLOSS subcultures (for example Lessig 2004).

Many in the movement, spearheaded by Stallman, have taken the argument about patents further, to encompass all intellectual property issues (Stallman 2002). Inspired by technological developments that have made the circulation of mass media artifacts such as films and music simple, proponents claim that technology has rendered the old IP system redundant, and that the music and film industries must adapt or die out.

Even further from software, FLOSS licences like the GPL have inspired a slew of licences that can be applied to forms of expression other than software – in particular the Creative Commons project aimed at building a layer of reasonable copyright. Two significant recent developments along these lines are the open-content encyclopaedia Wikipedia (edited by any visitor who cares to do so), which aims to constantly improve following the Bazaar principle described by Raymond, and the BBC iCan website, which aims to create a platform for cooperation between people to get information about and address issues that concern them (eg starting local campaigns). iCan is an example of a open model that allows individuals to pursue hobbies of interest or in order to develop social capital (Davis 2004).

5 Discussion of some strengths and weaknesses of FLOSS approaches

The following section draws on Benkler's (2002) work on commons-based peer-production models, and discusses some of the common strengths and weaknesses of FLOSS approaches today as they might apply within educational settings.

5.1 Transparency

Open standards and systems such as FLOSS ensure that participants understand what they are contributing to and make the code visible and available for others to use in an open way. However it is important to realise that much of the programming work carried out in the FLOSS community is done by a dispersed and often small group of programmers. Consequently just because the code is available and transparent does not mean that anyone can pick it up and start to work it. Like any computer language, you need the skills to be able to make FLOSS software. So despite advocates claiming that FLOSS empowers anyone to make their own work, you still need to be motivated and have the skills to work in this field.

5.2 Ease of use

Some forms of FLOSS software such as Linux, Apache and Mozilla are mature and therefore easy to install. In most circumstances in companies and schools the installation of such backend operating systems and servers is carried out by a technician or system administrator. Consequently, for most users this side of FLOSS may not be obvious to them as they are not involved or interested in how their computer servers work. Like all software, some FLOSS products are easier to install than others but for the most part they do not provide the lay person with a user-friendly or engaging interface.

For those interested in software and using FLOSS there are a range of products available on the market (refer to Appendix 1: Commonly used FLOSS solutions within schools). Looking through the myriad of applications available, one notices that many simply replicate existing proprietary products with little improvement, and in some cases are actually worse.

In this respect one has to tread carefully when making the case for FLOSS being easy to use; user interface design is costly and many small FLOSS companies do not have the resources to adequately invest in design or in appropriate documentation (eg user manuals, help sections). Consequently ease-of-use features tend to arrive later than commercial products and in some cases never arrive at all. For learners and teachers, who simply want to 'pick-up-and-go' and do not have the time for experimenting, much of the existing FLOSS educational software needs to be further developed.

5.3 Community structure

Commons-based peer-production methods generally comprise of a large number of individuals working on fine-grained aspects of a larger shared problem. This model can be very productive, responsive to incoming needs and cost effective. Key to this success is a shared common goal and the belief that the outcomes of the project will have wider benefit. Programmers working on large as well as small-scale FLOSS projects often consider bug fixing, adding new features or peer reviews as important and valuable insights which motivate them to continue to work on the problem, without monetary gain (Lehmann 2004). However as with any large-scale collaborative activities, minority groups can co-opt a project for their own ends, distorting the project's original intention (Mulgan, Steinberg and Salem 2005). Fissures and dissension within the group can be the downfall of FLOSS projects, which is why open source licences have been created and why open forums on the internet retain a certain level of centralised control or operate within a clear set of guidelines or rules. So although many FLOSS communities are often considered non-hierarchical, it is misconception to consider them as anarchic. Although they may not be as tiered as commercial organisations, successful projects often have a central figure or company who is the main facilitator and decision maker (eg Linus Torvalds is the central figure within Linux; the Apache Foundation is the central company coordinating the Apache server).

5.4 Creativity

Although many successful FLOSS projects bring together multidisciplinary groups of people, the group's composition does not necessarily mean it is more creative. Many FLOSS products are not new but open replications of existing proprietary software. However as FLOSS works on the basis of open standards and systems, the availability of such software has been shown over the long term to increase competition and creativity within the market place (Australian Government 2005; Peeling and Satchell 2001).

5.5 Peer efforts

One of the core strengths of the FLOSS approach is the peer review systems that underlie much of the development. Many people working on a problem at the same time can solve the problem quickly and easily. However, for a FLOSS product to be successful it needs to have enough critical reviewers in the pool who are reviewing and testing the software, and enough programmers who will fix the bugs. Projects can sometime die if this critical mass does not exist.

5.6 Business mode

Much is made of the cost-saving aspects of FLOSS. Reports by Becta (2005a, 2005b) demonstrate that the total cost of ownership is lower for schools when using FLOSS, indicating that FLOSS may be a useful solution for the deployment of large operating systems and servers in the public sector. However, this does not mean that moving to FLOSS means no cost at all; FLOSS is a business which utilises a different model from current proprietary software. Proprietary companies generally expect users to pay money upfront for their products, with the idea that they are 'buying' into a product line. Commercial FLOSS enterprises, on the other hand, do not expect users to pay upfront and do not lock users into one product type. They receive their money by providing services to the user; in this respect there is a shift from paying upfront to paying over the long term. The benefits of long-term investment are that as the product develops the user experiences are continually improving. In addition as FLOSS products comply with open standards, users are not tied in to one product and there is greater chance of interoperability between other FLOSS products (though not with proprietary software).

5.7 Funding new projects

As the code is open and available to all in FLOSS products, funding projects can be an issue. For example, investment capitalists are reluctant to invest in a venture that has a slow turnover. In addition, many FLOSS enterprises were developed or conceived as a result of non-open source funding, often secured through universities. For example, Linux is an open source version of the UNIX operating systems, parts of which were developed by funds from the Defence Advanced Research Projects Agency, and Linux was developed when its originator was at university. However, FLOSS methods of development continually push the boundaries of how software is made and developed. For example, the online peer-production encyclopaedia Wikipedia has been produced with little initial investment.

In sum, as Mulgan, Steinberg and Salem (2005, p23) note:

"Open source ways of developing software projects are exciting and powerful, but they are by no means universally applicable, or without downsides even when applied successfully. They share some of the more general limitations of networks, which tend to be relatively poor as a means of raising capital, concentrating resources or sustaining themselves through crises."

Acknowledging the limitations of networked-based approaches to peer production is necessary as it provides greater understanding of the appropriateness of such models for development. The following section builds on this discussion, focusing on the use of such models for education.

6 Peer-production approaches for education

In the fields of biotechnology (Rai 2005), politics (Starr 2002; Blume 2000), and religion[34], open source has begun to inspire more open and collaborative ways of thinking (Mulgan, Steinberg and Salem 2005). The main intention of this paper is to explore the wider implications of the FLOSS approach when applied to knowledge generation[35].

We take a particular focus on three areas of education: learners as knowledge builders; teachers as a community of professional practitioners; and innovation in software development for education. We have chosen these areas of focus in the light of current debates about the nature and goals of education, and in the light of calls for different approaches to the creation of educational resources. In particular, this discussion relates to:

- Current concerns with personalised learning and learner voice, both of which raise the question of how learners can be enabled to have more control and responsibility for their learning (Green et al 2005; Leadbeater 2004)[36].
- Questions about the nature of the curriculum, and whether learners should be encouraged to develop competences rather than simply acquire content knowledge in the education process (Bayliss 1999; RSA 2005)[37].
- Debates on the nature and future role of teachers in deploying and developing resources for teaching[38] and the current trend towards schools developing and selling curriculum materials and schemes of work[39].
- How to overcome the structural divides between developers of educational software and its users in order to create digital resources for education that more closely meet the needs of teachers and learners[40].

This discussion is not a marginal issue relating only to the question of whether schools should adopt open source systems to run their school networks, but is more profoundly related to issues central to current educational concerns.

6.1 FLOSS methods as a new approach to learning

There are two main ways in which we believe it may be possible to learn from FLOSS approaches to commons-based peer production in rethinking our approaches to teaching and learning.

6.1.1 Collaborative networks

In the first instance, distributed collaborative networks working together to solve problems could provide a powerful way of thinking about how learners might work together, within or across schools, to generate new knowledge and practice of relevance to them. This is primarily related to adopting FLOSS approaches and using digital technologies to mediate these in the goal of exploring other knowledge domains.

For a substantial period of time there has been concern that mass, industrial, teacher-directed pedagogies are not the most effective means of teaching in all instances. Instead, researchers and educators have led for calls for more collaborative, learner-centred approaches to education (Kumpulainen and Mutanen 1999; Littleton and Häkkinen 1999). This has led to a body of academic and practical work where emphasis is placed on facilitating learners in constructing knowledge through peer, cooperative and collaborative networks (Kruger 1992; Lave and Wegner 1991; Mercer 1995; Rogoff 1990). Some approaches, for example in the work of Scardamalia and Berieter (1991), have drawn on the principles of young people peer-reviewing the work of each other, collaboratively working on chunks of problems and coming to create a body of knowledge that is larger than anything an individual child might achieve on their own. Central to this idea is that learning and development are fundamentally the results of participation in social interactions and culturally organised activities with others[41].

It may be possible to draw on the lessons of the FLOSS movement and link these with the work in education theory, to create educational experiences for young people that allow them to work collaboratively on shared, practical problems to create knowledge of relevance to themselves and the wider community. Given the ongoing research into networked learning environments and computer-supported collaborative learning, it is possible to see how various research strands could provide us with a greater understanding of the role and implication of FLOSS for learning.

However, to date only limited research and practice has been carried out in linking FLOSS approaches with educational strategies, primarily in the field of work-based learning and higher education (Crowston and Howison 2005; Kim

2003; Lin forthcoming; Staring et al 2005). Much of this research focuses the social role and relationships, and forms of coordination and collaboration that occur when working on FLOSS projects. We suggest this is a line of enquiry that is worth pursuing at primary and secondary school level, particularly as it is now becoming feasible to consider the creation of education networks that link every child in every school through broadband connections and virtual learning environments. What might happen if mathematics problems, historical debates, scientific experiments and others were presented as shared challenges for young people to take up and tackle collectively, using the principles of free access to and use of information between children?

6.1.2 Children as programmers

In the second instance, it is possible to conceive of young people working together as programmers to create new resources and tools that are of relevance to them in supporting their own learning. This is more specifically related to using the programming practices of the FLOSS community to enable young people to engage with the business of building their own resources.

There is a long history of studies of children as programmers. Led by Seymour Papert's work with Logo this identifies clear benefits for children in participating in programming activity[42]. To date, however, there are few opportunities for young people to progress from Logo to other forms of participation in programming, nor sustainable models for them to enter the communities of practice surrounding technological development. Would the adoption of FLOSS approaches which blur the boundaries between developers of educational resources and users of these resources provide an interesting opportunity in this area?

It is useful to diverge to a related and contemporary field of interest that explores young people's out-of-school practices with digital technologies. 'Modding', generally defined as the practice of modifying and extending officially released games with fan-produced content, is one of the most distinctive features in current computer gaming culture. According to researchers at the Games Research Lab, Tampere, Finland:

"With its collaborative DIY aspects, appropriative nature and innovative use of new technology, modding represents a particularly illustrative example of the so-called 'participatory culture'."[43]

Participatory culture refers to the new media environment where the boundaries between audiences and producers have become increasingly blurred because of rapid advances in digital technology (Jenkins 1992). We are not suggesting that FLOSS and modding are the same thing, but that modding is an example where young people actively engage in a community of programming and software development, which shares some of the underlying principles that we find in peer-production approaches to software development. A principal difference between modding and the FLOSS approach, however, is that users are locked-in to the game - you can only play the mods on the original game platform[44].

Although this aspect of modding radically departs from FLOSS' core principles of open standards and format policies, it does indicate how access to source code can have greater good: pushing businesses to change their products, extending the life of a product, creating new communities of developers and in some cases even creating new economies and forms of learning and training. Mod communities provide a non-formal learning space for people to develop their skills, working on real, authentic and motivating problems, which have public significance. In addition, some companies pay modders and some modders become employed by the commercial companies; similar relationships and benefits between commercial enterprise and FLOSS communities have also been discussed (Lin forthcoming; Staring et al 2005). Modding demonstrates the benefits of engaging young people in programming problems where they can work on small chunks of larger games, changing them for their own ends. Encouraging young people to develop and customise programs, with facilitation from appropriate peers and mentors, is an avenue we believe worth further exploration. Would it be possible to develop (or further develop) open source educational resources where young people, able to access the underlying code, played a participatory role in improving, co-designing and creating these resources?

6.1.3 UK school examples

Early UK adopters of FLOSS, such as Richard Rothwell[45], provide anecdotal evidence to indicate that training young people and making them more aware of different computing systems has been beneficial for their learning. Reluctant to continually spend money on new computers, Rothwell began to explore FLOSS as an alternative. His school runs all its desktop computers using Linux, with OpenOffice as their main word processing application. Working with students to build the initial system in 2002-3, the school tested several small networks. With input from Sun Microsystems, the school improved its systems designs and added LTSP (Linux Terminal Server Project) to the existing Linux system, which allowed it to connect low-powered thin-client terminals (ie a network computer without a hard disk drive which uses a central server) to the main server. In 2003, Rothwell and colleagues formed the UK arm of Schoolforge which has become a central hub of activity and dissemination about FLOSS within the UK[46].

Rothwell encourages pupils to recycle and build computers for use in the school and wider community. The school continues to offer a wide variety of application choice to pupils, training young people and making them more aware of different computing systems and continually training staff in the migration to FLOSS. The school has received special Maths and Computing Status and because of this will be making all of its learning resources available online.

Another key UK example in this field is Orwell High School, also a specialist technology college, where the Deputy Head, John Osborne, has been instrumental in leading the school's FLOSS program. The escalating costs of running computer networks forced Osborne to consider FLOSS alternatives. With the support of the company the Cutter Project (www.cutterproject.com) the school significantly reduced the costs of buying in new computers by running a thin client on its existing machines. It runs a variety of FLOSS applications (Linux, Rosegarden, StarOffice) and has been able via the thin client architecture to embed work stations across the school. With the money saved, the school has supported teachers working with FLOSS applications in their own subject areas and provided access to pupils and teachers outside normal school hours[47].

It might be possible to conceive of FLOSS as offering models both for new approaches to collaborative, peer-reviewed educational experiences (perhaps also applicable to assessment systems) and for enabling young people to participate in the creation of digital learning resources. To date, however, evidence remains anecdotal and in order to further explore the potential benefits in this area we need strategic collaborations between the education software industry, the FLOSS community, teachers and learners and education researchers.

6.2 Knowledge sharing and collaboration between teachers

The FLOSS movement and its underlying open principles of software development and knowledge sharing have been considered as a prime example of collaborative learning in practice. As Kim notes:

"The open source software communities are one of the most successful - and least understood - examples of high-performance collaboration and community-building on the internet today. Other types of communities could benefit enormously from understanding how open source communities work." (Kim 2003)

Although Kim highlights that we lack an adequate understanding how FLOSS communities operate, research to date draws on the notion of community of practice as a way to analyse and discuss the practice and approaches of the FLOSS movement (Edwards 2001; Krishnamurthy 2002; Mockus, Fielding and Herbsleb 2002; Stürmer 2002; Tuomi 2000). In this section we discuss how the underlying principles of FLOSS could be applied to develop open-teacher based practices to enhance knowledge and practice in the education field (see Hargreaves 2003).

6.2.1 Communities of practice

Before summarising some of the research projects that have analysed FLOSS as a community of practice, it is necessary to explain what we mean by the term. For many years researchers (Brown, Collins and Duguid 1989; Lave 1979, 1988; Lave and Wegner 1991; Resnick and Resnick 1989) have argued that to understand learning we need to take into account how people learn not only in schools but also in everyday, home and community settings.

One of the most influential researchers within this area has been Jean Lave. One of Lave's (1979) main interests was how people solved problems in everyday life. Taking an anthropological perspective Lave considered everyday life as the fundamental context of problem-solving and together with Etienne Wenger, in their seminal book 'Situated Learning, Legitimate Peripheral Participation' (Lave and Wegner 1991), explored how people move from being novice outsiders to key members of a community of practice.Through a series of case study apprenticeships from tribal Yucatec midwives, to Vai and Gola tailors, navel quartermaster and meat cutters, Lave and Wenger discussed how newcomers to a particular practice engage in real, legitimate work that is connected to the work of the old-timers or masters. In doing so the apprentice becomes socialised into the field and their participation becomes more central.

Wenger (1998) extended this view of learning as a social process in his conception of a community of practice. Communities develop around things that matter to people (1998) where members of the community are involved in a set of relationships over time (Lave and Wegner 1991, p98). Participants are linked via a common purpose or particular area of knowledge and activity gives members a sense of identity and joint enterprise, that is, they have a common goal or set of goals pursued by the community. For a community to function there also needs to be a sense of mutual engagement and interdependence, which can in part be generated through a shared repertoire of ideas, commitments and memories. It also needs to develop various resources such as tools, documents, routines, vocabulary and symbols that in some way carry the accumulated knowledge of the community.

Examining FLOSS from the perspective of communities of practice, Tuomi (2005) draws attention to parallels between Lave and Wenger's work and how networked, distributed communities such as FLOSS operate. Tuomi, like many researchers (Edwards 2001; Krishnamurthy 2002; Mockus et al 2002; Stürmer 2002), considers FLOSS as a prime example of what Lave and Wenger would consider as an active community of practice. In describing FLOSS in this way, Tuomi notes:

"The new developers can learn their skills and work practice by developing code that extends the system's functionality but does not interfere with its core functionality. Gradually, the novices can then earn a reputation as reliable developers, and become masters and gurus in the project communities. This process of social integration and skills development is closely related to the architecture of the technical system that is being developed." (Tuomi 2005, p437)

Relating these understandings to teachers' practice one can begin to see how FLOSS could augment new modes of collaborative knowledge creation and sharing; specifically, how the FLOSS approach could extend teachers' practice through open content development and knowledge sharing.

6.2.2 Teaching communities of practice

Many researchers are concentrating their efforts on developing interoperable tools through which teachers, in collaboration with other professions, can develop new learning resources. Stephen Downes (National Research Council of Canada), in his keynote speech to the Open Source for Education conference[48], argued that we are moving towards a networked learning environment, central to the success of which is open content and software. Downes' vision for such networks is that they:

- are driven by user-generated content (eg like Wikipedia)
- are portable and genuinely owned by the participants
- support mixed modes of interaction
- are centres for personalised learning
- are diverse, interwoven and open.

Drawing an analogy between Benkler's (2002) analysis of commons-based peer-production, Downes concludes with a similar solution to how open learning networks should develop in the future. The key, he believes, is not in developing large-scale, integrated systems but small flexible components that can be easily integrated and responsive to changes in the environments. The following list provides an indication of some of the work in this area (for a detailed inventory refer to www.ossite.org):

- **MIT's OpenCouresWare (OCW)**[49] - offers open access to course materials to MIT courses.
- **Becta's Learning Platform Conformance Regime**[50] - aimed at improving interoperability between UK schools in relation to the information and purchasing of different platforms. The scheme's goal is to address standardisation issues and ensure that schools and will be able to 'plug and play' with transparency.
- **Stanford University's CourseWork systems**[51] - a website development and distribution system for educators to share materials.
- **MIT's Open Knowledge Initiative**[52] - a multidisciplinary group aimed at developing the next generation of networked e-learning resources.
- **eXeLearning**[53] - a project developing an offline authoring environment to assist teachers and academics in the publishing of web content without the need to become proficient in HTML or XML markup.
- **Schooltool**[54] - an ambitious project to develop a common global school administration infrastructure or framework that is freely available under an open source licence.
- **Skolelinux**[55] - a network architecture solution tailored for use in schools.

The relevance of FLOSS approaches to knowledge sharing and collaboration is not limited only to the potential new forms by which digital resources could be designed for education. It relates more profoundly to the question of how knowledge can be shared, modified and adapted across the teaching profession and in different contexts (Hargreaves 2003). It also relates to ongoing concerns about the relationship between educational research and practice.

We could conceive of networks of teachers and researchers working together on different elements of educational challenges to collaboratively create new approaches that are open to and usable by all. FLOSS approaches also offer potential routes by which educational research tools might be developed, refined and tested in different contexts with knowledge and results shared within the community. These approaches also raise questions about the growing trend towards copyrighting and selling on teaching strategies, curricula and schemes of work.

There is significant interest in this field at present, with online communities for teachers, school leaders and school advisors rapidly emerging. What the FLOSS approach highlights is the motivating factor of common goals and common problems around which people can work together through new networks. It offers an approach to these resources which moves them from information dissemination and sharing to collaborative problem solving.

6.3 Hybrid models of innovation

FLOSS embodies an exemplar community of practice engaged in the peer-production of software which has implications beyond the software arena. It offers a hybrid model of innovation where user and producer communities are intermingled, the notion of ownership is changed and the economical model of cost and reward is reworked. Luke notes:

"...When communities of practice are fostered and created around the use of new technologies, digital divides can be ameliorated, accommodated, and overcome. These communities are the Community Learning Networks, the Community Access Centres, the Open and Public Knowledge Initiatives, and the informal community based groups that share information and knowledge, support the acquisition of information and knowledge, and support and encourage community members to participate in the knowledge based society... cultivating communities of practice and learning networks that foster civic engagement and ensure open access is a key step in ensuring that digital citizenship is founded upon fundamental rights of participation and engagement for the public good." (extract taken from Luke's homepage: luke.rcat.utoronto.ca/thesis.html, accessed 15 May 2005)

For many advocates of FLOSS the wider socio-economic and cultural ramifications and implications of these working practices are of utmost importance (DiBona et al 1999; Ghosh 2005; Lerner and Tirole 2000; Lessig 2002; Tuomi 2000, 2005). As Lin notes:

"The development of FLOSS democratises software innovation processes and allows lay people to develop their understanding and knowledge of a shared problem or issue, especially through the web, to challenge established views on the issue." (Lin 2004, p1)

In this respect FLOSS can be considered as a "hybrid model of innovation" between the public and private sectors (Lin forthcoming). To best utilise this model, Lin, Cook and Burt (2001) argue that it is necessary for us to understand how this community mobilises social capital, achieves its shared goals and resolves internal differences. Underlying this community is a structure in which technology-producing communities substantially overlap with technology-using communities (Tuomi 2005).

This emphasis differentiates FLOSS from proprietary commercial models of software development where users and developers are separated and only connect through economic transactions. This is of particular significance in the education arena, as argued in the UK Department for Education and Skills e-learning strategy consultation document 2003:

"The lack of a direct relationship between the users and the suppliers means that the products developed are less likely to meet learners' and teachers' real needs. We have not yet found the right mechanisms for the partnerships we need between developers and users. We have to create the conditions in which innovative ideas for e-learning pedagogy will flourish... Commercial suppliers usually employ teachers at some stage in the design process, but unless the partnership is close, and educational requirements lead the development, there is little chance of achieving either good pedagogy or profitable products."

The FLOSS model also leads to alternative understandings of ownership whereby control lies in the developmental dynamic of an evolving product (Tuomi 2005, p442)[56]. This naturally leads to differences in the value, cost, benefit, distribution and consumption of goods. Benefits to the programmers involved in such endeavours are not immediately visible or easily quantifiable. For example the developer many not gain a monetary payment but may gain access to information and system functionality that they may otherwise not have (von Hippel and von Krogh 2003).

FLOSS provides a challenge to our current economic models, which are based on traditional notions of supply and demand and the scarcity of resources, as it demands new models and ways of thinking about innovation and economic development (Tuomi 2005). New economic models of technological innovation bring together traditional thinking with a contemporary understanding that consumers have now also become producers (Jenkins 1992). At the heart of this model is the concept of collective production and social exchange for the public good. This has led to the notion of new hybrid systems of innovation that use the openness of FLOSS while revising our notions of ownership and intellectual property. Lessig (2002) notes that this requires us to understand the values of FLOSS as essentially the values of a free society, and proposes that our challenge is to find ways to get people to see the value of the common good as well as intellectual property.

The FLOSS approach offers open models of innovation where code is shared between public and private sectors so that new innovations can emerge. This form of knowledge sharing has been a precursor to many major technological innovations. The free sharing of code characteristic in the early stages of software development continues today in many academic institutions. This may be effective as a method of encouraging industrial exploitation of academic research (Peeling and Satchell 2001). Drawing a similar conclusion from her research on hacker culture, Lin (2004) notes that there has been a:

"...Strategic collaboration between the public (ie the free software community) and the private (ie information technologies corporations) sectors [which] symbolises a pattern of hybrid innovation that entails complex communications and networks."

The complexities of this relationship are often overlooked or ignored. FLOSS has not developed in isolation from the proprietary market. In many ways, one side of the coin cannot exist without the other and this interconnected model has been very popular within the US for many years. As Peeling and Satchell note, it is "hard to over-state the beneficial effect this has had on the technology and the wider computer industry" (2001).

In the creation of digital education resources, these hybrid models of innovation open up a number of questions and challenges. They require us to ask:

- What models of ownership might we need to develop for public-private collaboration?
- What mechanisms might need to be put in place to encourage the early stage exchange of code between different sectors (notwithstanding the fact that many cutting edge technologies are primarily developed for the business rather than education sectors)?
- What role might users of educational resources want to play in the creation of these resources, and what might need to be put in place to facilitate this?
- What business models would need developing to allow further exploration in this area?

These are strategic and fundamental questions that will need addressing in order to fully exploit the potential for improved technologies offered by the model of sharing code openly between different communities.

7 Conclusion

FLOSS is more than software: it is a cultural phenomenon that is underpinned by technological development with the aim of contributing to the public good. It is of relevance to our understanding of how people learn and produce knowledge; of how communities collaborate and work to solve problems; and of how innovative practices emerge.

The early history and development of FLOSS has been dependent on the availability of technologies that connect large numbers of people in a distributed network. The availability of such technologies continues to increase and the power of them continues to expand. Educators are just starting to understand the implications of such technologies and to incorporate them into teaching and learning models.

In an attempt to accelerate debate in this area we propose a series of questions that should stimulate thinking and highlight challenges. If you use these questions or have comments to make on any of the issues raised we would be very interested to hear from you at research@futurelab.org.uk.

Teachers and learners as co-creators

- To what extent would learners and teachers want to or be able to become involved in co-creating digital resources?
- What might be the barriers to this? What might be the enablers?
- What kinds of open systems would teachers and learners want and what levels of functionality and ease-of-use would they require?
- What kinds of authoring tools need to be developed in order to support teachers and learners to become more actively involved in co-creation of resources?
- What kinds of resources and policies do we need to put in place in order for the different communities (FLOSS, software industry, researcher and educators) to work together?
- What business models would need to be developed to enable co-creation to support all sectors?

Developing learning within communities of practice

- How can FLOSS as an example of commons-based peer-production be applied to other areas of learning, both in and out of schools?
- For learners and for teachers as researchers, what common problems or goals would be most tractable for a commons-based peer-production approach to be beneficial?
- What infrastructure would need to be in place to enable commons-based peer-production across and within schools?

Supporting hybrid innovation at policy level

- What time-spans are we working towards – 5, 10, 15 years? What would our shared visions for these periods look like?
- What levels of 'openness' would need to be embedded within any FLOSS models in education?
- How can we ensure any FLOSS systems that are developed are flexible and scaleable to suit small and large scale organisations?
- What kinds of resources and expertise do school ICT departments need to develop if they are to explore FLOSS approaches - eg what forms of documentation, training and assistance are necessary, do they exist and if not how we can begin to develop such resource?
- How can we support existing good practice with FLOSS approaches?
- What kinds of resources are necessary to invest in schools that have already migrated to FLOSS, so as to ensure they have a healthy future?
- How can we best distil and disseminate the lessons learnt and the best practices that have emerged from the wider FLOSS community to education?

Links

This list is a starting point for links relating to FLOSS software and resources. For a more definitive list refer to Becta (2005b), Bruggink (2003), Vuorikari (2003).

General

History of open source: **www.opensource.org/docs/history.html**
Eric Raymond, author of 'The Cathedral and the Bazaar', homepage: **www.catb.org/~esr/**
Lawrence Lessig: **www.lessig.org/blog**
Creative Commons: **creativecommons.org**
Open Source Initiative (OSI): **www.opensource.org**
Newsforge: **www.newsforge.com** - online paper on Linux and open source
Sourceforge: **sourceforge.net** – main industry site
Red Hat: **www.redhat.com** - international company providing FLOSS solutions
Linux: **www.linux.org** - official site for this FLOSS operating system

Education and general policy

Becta, Open Source Teaching: **www.becta.org.uk/research/research.cfm?section=1&id=3197**
Uk Government policy and guidelines on FLOSS: **www.ogc.gov.uk/index.asp?id=2190&**
EU FLOSS forum: **www.ossite.org**
Samba: **us1.samba.org/samba** - an Australian-based open source software initiative providing interoperability between computers running Linux/UNIX and those running Windows
eEurope 2005 Action plan: **europa.eu.int/information_society/eeurope/2005/index_en.htm**
UK e-Government Interoperability Framework (e-GIF): **www.govtalk.gov.uk/schemasstandards/egif.asp**

FLOSS educational examples and resources

Schoolforge-UK: **www.schoolforge.org.uk**
Cardiff Schools: **www.cardiffschools.net**
Schools Interoperability Framework Association: **www.sifinfo.org**
SchoolTool: **www.schooltool.org**
The Association for Free Software: **www.affs.org.uk/education**
Linux in education: **seul.org/edu**
BBC resources: **www.bbc.co.uk/opensource**
MIT course projects for schools: **ocw.mit.edu**
General information on best practice for schools: **www.ict-register.net**
K-12 Linux Project: **www.k12linux.org** – large-scale US project, supporting the use of Linux in K-12
KDE Edutainment Project: **edu.kde.org** - focusing on developing open source educational products for 3-18 year-olds

References

Australian Government (2005). A Guide to Open Source Software for Australian Government Agencies: Developing and Executing an ICT Sourcing Strategy. Australian Government Information Management Office. Retrieved 1 November 2005. www.sourceit.gov.au/__data/assets/pdf_file/42065/A_Guide_to_Open_Source_Software.pdf

Bayliss, V (1999). Opening Minds: Education for the 21st Century. London: The Royal Society for the Encouragement of Arts, Manufactures and Commerce (RSA)

Becta (2004). Using ICT to Share the Tools of the Teaching Trade. Coventry: British Educational Communications and Technology Agency (Becta)

Becta (2005a). Open Source Software in Schools. Coventry: British Educational Communications and Technology Agency (Becta)

Becta (2005b). A Study of the Spectrum of Use and Related ICT Infrastructure Costs. Coventry: British Educational Communications and Technology Agency (Becta)

Benkler, Y (2002). Coase's penguin, or Linux and the nature of the firm. The Yale Law Journal, 112

Berry, DM (2004). The contestation of code: a preliminary investigation into the discourse of the free/libre and open source movement. Critical Discourse Studies, 1(1)

Brown, JS, Collins, A and Duguid, P (1989). Situated cognition and the culture of learning. Educational Researcher, 18(1), 32-42

Bruggink, M (2003). Open Source Software: Take it or Leave it? The status of open source software in Africa: a study towards informed decision-making on ICT platforms. www.ftpiicd.org/files/research/reports/report16.pdf

Chawner, BL (2005). Participant Satisfaction with Open Source Software. Victoria University, Wellington

Clements, DH (2001). Mathematics in the preschool. Teaching Children Mathematics, 7(4), 270-275

Cole, M (1991). On socially shared cognitions. In L Resnick, J Levine and S Behrend (eds), Socially Shared Cognitions (pp136-170). Hillsdale, NJ: Erlbaum

Collman, B., & Hill, M. (2004). How Free Became Open and Everything Else Under the Sun. MC: Journal of Media and Culture, 7(3).

Crowston, K and Howison, J (2005). The social structure of open source software development teams. First Monday (peer-reviewed journal on the internet)

Davis, W (2004). Trade Union Membership and the Internet: Lessons from Civil Society. London: IPPR

DiBona, C, Ockman, S and Stone, M (eds) (1999). Open Source: Voices from the Open Source Revolution: A Set of Primary Texts from the Key Players in the Development of FLOSS. Sebastopol, CA: O'Reilly & Associates

Edwards, K (2001). Epistemic Communities, Situated Learning and Open Source Software Development. Retrieved 16 May 2005 from opensource.mit.edu/papers/kasperedwards-ec.pdf

Fisher, D (2004). Open Source and the Corporate Desktop. Retrieved May 15 2005, from www.nccmembership.co.uk/pooled/articles/BF_WEBART/view.asp?Q=BF_WEBART_124159

Ghosh, RA (2005). Free/Libre/Open Source Software as a Learning Environment. Paper presented at the International Symposium on Open Source Software, Abano Terme, Padova

Gillespie, CW (2004). Seymour Papert's vision for early childhood education? A descriptive study of head start and kindergarten students in discovery-based, Logo-rich classrooms. Early Childhood Research and Practice, 6(1)

Gould, M, Barnett, L, Kimberly, Q, Dowling, Q and Hogan, L (2005). Open Source Usage is up, but Concerns Linger. Forrester

Green, H, Facer, K, Rudd, R, Dillon, P and Humphreys, P (2005). Personalisation and Digital Technologies. Bristol, UK: Futurelab

Hargreaves, D (2003). Working Laterally: How Innovation Networks Make an Education Epidemic. DfES/Demos: London

Jenkins, H (1992). Textual Poachers. Television Fans and Participatory Culture. New York & London: Routledge

Khalak, A (2000). Economic Model for Impact of Open Source Software. Retrieved 16 May 2005, from opensource.mit.edu/research_directory.php

Kim, EE (2003). An Introduction to Open Source Communities. From opensource.mit.edu/papers/blueoxen.pdf

Kingstone, S (2005). Brazil Adopts Open Source Software. Retrieved 3 June 2005, from news.bbc.co.uk/1/hi/business/4602325.stm

Krishnamurthy, S (2002). Cave or Community: An Empirical Examination of 100 Mature Open Source Projects. Retrieved 15 May 2005, from www.firstmonday.dk/issues/issue7_6/krishnamurthy/

Kruger, A-C (1992). Peer collaboration: conflict, co-operation or both. Social Development, 2(3), 165-182

Kumpulainen, K and Mutanen, M (1999). The situated dynamics of peer group interaction: an introduction to an analytic framework. Learning and Instruction, 9(5), 449-473

Lave, J (1979). What's special about experiments as contexts for thinking? In YEM Cole and O Vasquez (eds), Mind, Culture and Activity: Seminal Papers from the Laboratory of Comparative Human Cognition (pp57-69). New York: Cambridge University Press

Lave, J (1988). Cognition in Practice. Cambridge University Press

Lave, J and Wegner, E (1991). Situated Learning: Legitimate Peripheral Participation. Cambridge, England: Cambridge University Press

Leadbeater, C (2004). Personalisation Through Participation. London: Demos

Lehmann, F (2004). FLOSS Developers as a Social Formation. Retrieved 31 May 2005, from www.firstmonday.org/issues/issue9_11/lehmann/

Lerner, J and Tirole, J (2000). The Simple Economics of Open Source. Retrieved 16 May 2005, from www.nber.org/papers/w7600

Lessig, L (2002). Open code and open societies [draft]. Paper presented at Free Software - a Model for Society? Tutzing, Germany

Lessig, L (2004). Free culture: how big media uses technology and the law to lock down culture and control creativity. New York: The Penguin Press

Levy, S (1984). Hackers. New York: Anchor/Doubleday

Levy, S (1994). Hackers: Heros of the Computer Revolution. Garden City: Anchor Press

Lin, N, Cook, K and Burt, RS (eds) (2001). Social Capital: Theory and Research. New York: Aldine de Gruyter

Lin, Y (2004). Hacking Practices and Software Development: A Social Worlds Analysis of ICT Innovation and the Role of Free/Libre Open Source Software. University of York Science and Technology Studies Unit, Department of Sociology, York

Lin, Y (forthcoming). Hybrid innovation: how does the collaboration between the FLOSS community and corporations happen? Knowledge, Technology and Policy

Littleton, K and Häkkinen, P (1999). Learning together: understanding the processes of computer-based collaborative Learning. In P Dillenbourg (ed), Collaborative Learning, Cognitive and Computational Approaches. Pergamon

McGowan, D (2005). Between logic and experience: error costs and United States v Microsoft Corp. Berkeley Technology Law Journal, 20(2)

McKusick, MK (1999). Twenty years of Berkeley UNIX: from AT&T-owned to freely redistributable. In C DiBona, S Ockman and M Stone (eds), Open Sources: Voices from the Open Source Revolution. Sebastopol, CA: O'Reill

Mercer, N (1995). The Guided Construction of Knowledge, Talk Among Teachers and Learners. Clevedon, Philadelphis, Toronto, Syndney, Johannesburg: Multilingual Matters LTD

Mockus, A, Fielding, T and Herbsleb, JD (2002). Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11(3), 309-346

Moyle, K (2003). Open Source Software and Australian School Education. Education.au limited for consideration by the MCEETYA ICT in Schools Taskforce

Peeling, N and Satchell, J (2001). The Analysis of Impact of Open Source Software. London: QinetiQ

Perens, B (1999). The open source definition. In C DiBona, S Ockman and M Stone (eds), Open Sources: Voices from the Open Source Revolution. Sebastopol, CA: O'Reilly & Associates

Philips, S (2005). Modelling Open Source. Paper presented at FLOSSIE (Free, Libre, Open Source Software in Education), Bolton, UK

Proffitt, B (2002). Venezuela's Government Shifts to Open Source Software. Retrieved 15 May 2005, from linuxtoday.com/developer/2002083001126NWLLPB

Rai, AK (ed) (2005). Open and Collaborative Research: A New Model for Biomedicine. Washington, DC: AEI-Brookings Press

Raymond, ES (1999). The Cathedral and the Bazaar. Sebastopol, CA: O'Reilly & Associates

Raymond, ES (2001). Why Microsoft smears - and fears - open source. Spectrum, 38, 8, 14-15

Resnick, LB and Resnick, DP (1989). Assessing the Thinking Curriculum: New Tolls for Educational Reform. Washington, DC, National Commission on Testing and Public Policy

Rogoff, B (1990). Apprenticeship in Thinking: Cognitive Development in Social Context

RSA (2005). Opening Minds, Giving Young People a Better Chance. The Royal Society for the Encouragement of Arts, Manufactures & Commerce (RSA): London

Scardamalia, M and Bereiter, C (1991). Higher levels of agency for children in knowledge building: a challenge for the design of new knowledge media. Journal of the Learning Sciences, 1(1), 37-68

Shankland, S (2003). Munich Breaks with Windows for Linux. Retrieved 31 May 2005, from news.zdnet.com/2100-3513_22-1010740.html?tag=nl

Stallman, RM (1992). Why Software Should Be Free. Retrieved 31 May 2005, from www.gnu.org/philosophy/shouldbefree.html

Stallman, RM (2002). Free Software, Free Society: Selected Essays of Richard M Stallman. GNU Press. www.gnupress.org/gnupresspub.html

Staring, K, Titlestad, O and Gailis, J (2005). Educational transformation through open source approaches. Paper presented at the Information Systems Research Seminar in Scandinavia (IRIS), Kristiansand, Norway

Stürmer, M (2002). Open Source Community Building. Never of Bern, Bern

Tuomi, I (2000). Internet, Innovation, and Open Source Actors in the Network. Retrieved 15 May 2005, from www.firstmonday.org/issues/issue6_1/tuomi/

Tuomi, I (2005). The future of open source: trends and prospects. In M Wynants and J Cornelis (eds), How Open is the Future? Economic, Social and Cultural Scenarios Inspired by Free and Open Source Software. Brussels: VUB Brussels University Press

von Hippel, E and von Krogh, G (2003). Open-source software and the 'private-collective' innovation model: issues for organisation science. Organisation Science, 14(2), 209-223

Vuorikari, R (2003). Why Europe Needs Free and Open Source Software and Content in Schools. Brussels: European Schoolnet

Vygotsky, LS (1978). Mind In Society: The Development of Higher Psychological Processes. Cambridge, MA: Harvard University Press

Vygotsky, LS (1988). On inner speech. In MB Franklin (ed), Child Language: A Reader, pp181-187. London, Oxford University Press

Wenger, E (1998). Communities of Practice - Learning, Meaning and Identity. New York: Cambridge University Press

Wertsch, J and Tulviste, P (1998). LS Vygotsky and contemporary developmental psychology. In D Faulker, K Littleton and M Woodhead (eds), Learning Relationships in the Classroom. London and New York: Routledge in association with The Open University Press

Williams, S (2002). Free as in Freedom: Richard Stallman's Crusade for Free Software. Sebastopol, CA: O'Reilly

Yelland, N (1995). Logo experiences with young children: describing performance, problem-solving, and social contexts of learning. Early Child Development and Care, 109, 81-74

Appendix 1: Commonly used FLOSS solutions within schools

School networks – administration and management

FLOSS is a very popular backend systems solution (backend solutions refer to the servers and other hardware that support a network, eg internet gateway, web servers, and e-mail servers). The technicians that build backend computing systems are usually looking for the simplest, most powerful, secure and cost-effective solution. Linux is an international, leading edge FLOSS operating system and as noted has become one of the greatest success stories of the FLOSS movement, while Apache has become a similar leading edge web server. Many schools use Linux and Apache without users even realising they are using FLOSS tools. Schools don't have to use Linux to use open source software, as there are many open source programs for use with Microsoft Windows or Apple Macintosh.

Computer software

Some of the most commonly used content FLOSS applications such StarOffice[57], OpenOffice[58] and GIMP[59] offer schools significant advantages with easily deployable solutions, low risk and costs and, as suggested in the recent Becta report (2005b), have proven to be extremely successful in schools. Becta found a wide range of different FLOSS programs installed in the schools they examined, the most popular of which are described in Table 2. For more detailed descriptions of the range of FLOSS software used in primary and secondary schools please refer to the appendix at the end of the Becta (2005b) report.

Table 2: Commonly used FLOSS alternatives

Purpose	Open source software	Equivalent or similar proprietary software
Operating system	Linux	Microsoft Windows
Web browser	Mozilla[60]	Internet Explorer
Office suite	OpenOffice.org or StarOffice	Microsoft Office
Scaleable database	MySQL[61]	Oracle or Microsoft SQL Serve
Scripting language for the web	PHP[62]	ASP or Coldfusion
Web server	Apache[63]	Microsoft IIS
Online learning content management package	Moodle[64]	Microsoft Press
Audio editor	Audacity[65]	Sony Sound Forge or Steinberg Wavelab
Real-time circuit simulator	Crocodile Clips[66]	LEGO Mindstorms
Graphic viewer for Windows	IrfanView[67]	Adobe Photoshop
Image editor	GIMP	Adobe Photoshop
Desktop publishing	Scribus[68]	Adobe Photoshop
Drawing	Inkscape[69]	Adobe Illustrator
Music sequencing package	Rosegarden[70]	Stenbergs Cubase or eMagic Logic

Taken from www.netc.org/openoptions/pros_cons/comparing.html, North Regional Educational Laboratory, Portland, Oregon)

Notes

1 www.netc.org/openoptions/pros_cons/comparing.html (retrieved 8 August 2005)
2 www.sourceit.gov.au/__data/assets/pdf_file/42065/A_Guide_to_Open_Source_Software.pdf (retrieved 9 August 2005)
3 news.netcraft.com/archives/web_server_survey.html
4 The Open Source Initiative: www.opensource.org (retrieved 15 May 2005)
5 Abbreviated from source: www.opensource.org/docs/definition.php (retrieved 8 August 2005)
6 For a full report of the conference at which this paper was presented see www.schoolforge.org.uk/index.php/FLOSSIE_2005_Report (retrieved 10 August 2005)
7 www.yale.edu/yalelj/112/BenklerWEB.pdf
8 For a comprehensive snapshot of research in this field refer to opensource.mit.edu. This site attempts to establish a community in which information is freely exchanged, with the aim to further understanding of open source and its implications outside the realm of software development
9 www.schoolforge.net (retrieved 11 November 2005)
10 www.schoolforge.org.uk/index.php/FLOSS (retrieved 11 November 2005)
11 www.ossite.org (retrieved 11 November 2005). The SIGOSSE project closed in December 2005. The project will continue under the guise of a new EU network initiative called 'The Bazaar', which will provide a European platform for all stakeholders around open source for learning. Details of the project will be available in 2006 at www.bazaar.org
12 flosse.dicole.org (retrieved 11 November 2005)
13 FLOSSWorld project: flossworld.org
14 FLOSSpols: flosspols.org/index.php
15 msdn.microsoft.com/library/default.asp?url=/library/en-us/modcore/html/deconsourcecodesharing.asp
16 www.demos.co.uk
17 www.jisc.ac.uk
18 www.becta.org.uk
19 www.govtalk.gov.uk/documents/QinetiQ_OSS_rep.pdf
20 For example refer to: news.netcraft.com/archives/2004/01/28/debian_fastest_growing_linux_distribution.html (retrieved 15 November 2005)
21 Regulatory Keynesian models of economy were the norm, and strict regulations were designed to hold back large corporations from the monopolistic excesses of the utility industries of the 1920s and 1930s
22 TCP/IP (Transmission Control Protocol/Internet Protocol) refers to the suite of communications protocols (ie the set of rules which defines how devices communicate to each other) used to connect hosts on the internet. TCP/IP uses several protocols, the two main ones being TCP and IP. TCP/IP is built into the UNIX operating system and is used by the internet, making it the de facto standard for transmitting data over networks. Even network operating systems that have their own protocols, such as Netware, also support TCP/IP (definition taken from www.webopedia.com/TERM/T/TCP_IP.html, retrieved 31 May 2005)
23 One of the senior technical members of the Computer Systems Research Group at the University of California, Berkeley, who joined in 1986 and is now the founder member of Sleepycat www.sleepycat.com (retrieved 11 November 2005)
24 The hacker culture which has become central to the open source philosophy grew out of early mainframe users at MIT who had an ethos of sharing and co-development (Levy 1984)
25 Refer to the GNU Operating System site: www.gnu.org (retrieved 31 May 2005)
26 See www.gnu.org/gnu/manifesto.html
27 Free Software Foundation: www.fsf.org (retrieved 31 May 2005)
28 Taken from www.gnu.org/gnu/thegnuproject.html (retrieved 8 August 2005)
29 creativecommons.org
30 Lisp is a functional programming language developed at MIT by John McCarty and is used today in a variety of programs. For a history of Lisp refer to en.wikipedia.org/wiki/LISP#History
31 www.linux.org
32 Mozilla Firefox: www.mozilla.org/products/firefox
33 Intel Open Source Products: www.intel.com/software/products/opensource

34 www.opensourcejudaism.com (retrieved 15 November 2005)
35 Although Mulgan and colleagues consider this positive, they reiterate that 'open source' is still a very recent term, which specifically describes work where the source code is open. Mulgan et al believe that applying it to non-software areas muddies the waters and is unhelpful and confusing. As a result they claim we need new terms which "acknowledge the growing diversity of open methods". In this respect we are careful in applying FLOSS approaches to non-software areas, to be sensitive with the terminology we use
36 www.demos.co.uk/catalogue/personalisation (retrieved 30 November 2005); www.futurelab.org.uk/research/personalisation.htm (retrieved 30 November 2005)
37 www.thersa.org/projects/publications.asp
38 Refer to Ralph Tabberer's presentation at Futurelab's 'Beyond the Blackboard' conference, 2004: www.futurelab.org.uk/events/past/bb_pres/rt01.htm (retrieved 30 November 2005)
39 For example: learningmatters.com/ThomasTelfordTraining (retrieved 30 November 2005)
40 For example: www.dfes.gov.uk/publications/e-strategy (retrieved January 2004)
41 Underpinning most of these approaches is a sociocultural theory of learning drawing on the work of Vygotsky (see Cole 1991; Rogoff 1990; LS Vygotsky 1978, 1988; Wertsch and Tulviste 1998)
42 Although there have been conflicting results as to the cognitive benefits of Logo (Yelland 1995), many researchers have found that it can influence, for the better, children's ability to construct ideas (Gillespie 2004), their spatial awareness, general problem-solving ability (Clements, 'Metacognition, learning, and educational computer environments' #116, 2002 #117) and mathematical understandings (Clements 2001)
43 www.uta.fi/hyper/projektit/mc2/research_intro.html (retrieved 11 November 2005)
44 One of the first games to enable modding was Doom (id Software 1993), the company actively encouraging players to change the game by designing it to be modified (Kushner 2003, 165-169). www.idsoftware.com (retrieved 11 November 2005)
45 Head of Computing, Handsworth Grammar School, Birmingham: handsworthgrammarschool.co.uk
46 www.schoolforge.net
47 www.orwellhs.suffolk.sch.uk
48 To view Downes' PowerPoint presentation, refer to www.downes.ca/files/heerlen.ppt (retrieved 17 November 2005)
49 ocw.mit.edu (retrieved 15 November 2005)
50 www.becta.org.uk/industry/advice/advice.cfm?section=5&id=4370 (retrieved 15 November 2005)
51 aboutcoursework.stanford.edu/overview.html (retrieved 15 November 2005)
52 www.okiproject.org (retrieved 15 November 2005)
53 exelearning.org (retrieved 15 November 2005)
54 www.schooltool.org (retrieved 15 November 2005)
55 www.skolelinux.org/portal (retrieved 15 November 2005)
56 www.dfes.gov.uk/publications/e-strategy/strategy.stm (retrieved 5 January 2004)
57 www.staroffice.org
58 www.openoffice.org
59 www.gimp.org
60 www.mozilla.org
61 www.mysql.com
62 www.php.net
63 www.apache.org
64 moodle.org
65 audacity.sourceforge.net
66 www.crocodile-clips.com
67 www.irfanview.com
68 www.scribus.org.uk
69 www.inkscape.org
70 www.rosegardenmusic.com (educational licence is £40)

About Futurelab

Futurelab is passionate about transforming the way people learn. Tapping into the huge potential offered by digital and other technologies, we are developing innovative learning resources and practices that support new approaches to education for the 21st century.

Working in partnership with industry, policy and practice, Futurelab:

- incubates new ideas, taking them from the lab to the classroom
- offers hard evidence and practical advice to support the design and use of innovative learning tools
- communicates the latest thinking and practice in educational ICT
- provides the space for experimentation and the exchange of ideas between the creative, technology and education sectors.

A not-for-profit organisation, Futurelab is committed to sharing the lessons learnt from our research and development in order to inform positive change to educational policy and practice.

Registered charity 1113051