Status Report on Open Source Software November 2003

From the Open Source Subgroup of EURIM's Modernising Government Working Party

This paper sets out to explain the background to Open Source Software (OSS) – what it is, what it is not, how it developed, and how it differs from proprietary software. The paper gives an overview of the debate on OSS, summarises some recent developments in both the policy arena and the marketplace, and takes a look at the different business models. Contributors include BSI, Fujitsu, IBM, Microsoft, OeE, OpenForum Europe, The Corporate IT Forum (tif).

There is a wide spectrum of very strongly held views in this arena and whilst this paper attempts to characterise some of the main actors in the debate and summarise their arguments and views, it has been necessary to generalise and simplify these positions to a great extent. The detail required to give a comprehensive account is beyond the scope of our activity. There are many excellent sources of further information, which set this material into context and these references are given in the paper.

TABLE OF CONTENTS

SECTION I – ISSUES and RECOMMENDATIONS

SUMMARY & RECOMMENDATIONS

Summary Points

Open Source Software (OSS) is software whose source code (the computer instructions that form the basis of a software program) is openly published, is often developed by individuals using the internet to support collegiate behaviour, and often by voluntary effort. OSS may typically be available for download free of charge, but most business users will choose to obtain it from a vendor in supported, packaged form.

OSS is a serious alternative solution to proprietary software for many ICT applications and is now taking a significant market share in some parts of the software market, notably infrastructure, and can be expected to advance further over time.

OSS offers a new business model, one which is widely accepted as being here to stay and will impact many in ICT, from supplier through to user, and which supports a shift in emphasis in the industry away from the vending of software code alone and towards code plus value added services provision.

In a recent survey some Government users perceived the benefits of OSS as being lower total cost of ownership (TCO), avoidance of lock-in, and potential for easier sharing of applications amongst organisations. However, OSS is not a cure-all. It is currently suitable for a limited range of applications and its suitability for those depends on a number of factors.

The debate on the respective merits and de-merits of the proprietary and open source software models appears to have become ideological and polarised. However, there are signs that pragmatic approaches are increasingly being adopted by the different stakeholders. It is worth noting in recent years that the proprietary software sector has begun to contribute its resources and discipline to the development of OSS, and numbered among the most important contributors are traditional software vendors such as IBM and HP.

There is room for both OSS and proprietary models of software development to co-exist. The market now provides a substantial middle ground where vendors, whether traditionally OSS or proprietary, are either offering packaged versions of OSS, or are integrating or incorporating OSS code within and alongside proprietary software. Other proprietary developers are adopting models that allow limited access to the source code

Current UK Government procurement policy states that there should be a level playing field for procurement so that both can indeed co-exist at least in terms of Government buying. [See OEE policy statement[1]]

In policy and strategy terms the UK is at a similar stage to other EC member states, but other countries, notably Germany, take a more proactive approach in terms of active initiatives to support the introduction and use of open source.

Recommendations for Parliamentarians

Parliamentarians should take an interest in these issues:- like many other developments in IT and technology they have important implications for economic and societal development.

Parliamentarians should recognise that as the UK moves towards a knowledge-based economy, intellectual property rights will become an increasingly important issue.

Parliamentarians must be aware that the situation is complex, and there are too many considerations, for one model to be generically favoured over another. The different software models should be treated on a level playing field as far as public policy decisions are concerned, and procurement decisions should be made on a value-for money (VFM) basis.

A whole array of different models now exists in the marketplace and it is no longer relevant to consider OSS and proprietary solutions as mutually exclusive alternatives. OSS has in many cases a commercial element and should be viewed as a valid competitor to proprietary software.

Although Government policy is clear and encourages the consideration of open source software (OSS) for appropriate applications alongside proprietary software, in view of the lack of maturity of user understanding,

[1] http://www.govtalk.gov.uk/policydocs/consult_subject_document.asp?docnum=780

the user community would welcome more practical guidance in those areas where open source can be considered.

Government must develop a very thorough understanding of the different licensing models that apply to OSS and their implications for the management of intellectual property rights.

If government does move towards increased use of OSS then policy-makers must carefully consider the need to put structures in place for arbitrating development decisions when OSS solutions are to be shared, to prevent incompatible or parallel versions being developed.

THE ISSUES

Complexity

i) Complexity of solutions. The advantages of OSS for one application may not apply to another and will differ according to the type of licence, size and structure of organisation and support available, whether in-house or external. There is a complex matrix of things to consider when comparing OSS and proprietary solutions, it is not a simple choice. OGC has provided helpful advice[2] on general advantages and disadvantages and on assessing the merits of OSS in procurement and on the broad areas of the software infrastructure and application marketplace where OSS currently has strengths and weaknesses. The most common arguments for and against OSS are summarised below.

ii) Complexity of licence regimes – contrary to some perceptions, strict licensing is not just the domain of proprietary software. There is a spectrum of licences for OSS, just as there is for conventional software, and the most common licences are summarised below (see *licensing issues* below, and Intellect's position paper[3]).

Confusion

iii) Confusion over polarised opinions – Until relatively recently, the debate had become polarised, positions were entrenched and arguments were emotive and ideological. Users no longer knew whom to believe. This was partly an inevitable result of the way that open source was initiated – as a viable alternative to proprietary software produced overtly in opposition to it at the start. This hostility still prevails among some elements, but there has recently been substantial movement towards a middle ground where a more rational debate now thrives. Within the spectrum of opinions that exist, there is a large central body that takes the view that there are some circumstances in which proprietary software is more appropriate, some where OSS offers advantages, and some where a combination of proprietary and OSS provides the best solution.

iv) Confusion over basic terminology. There is enormous confusion over definitions and this is not helped by the "cybernerd" jargon that inevitably creeps into all discussions. The glossary section of this paper and the various analogies used in this paper are designed to help address this.

BUSINESS PERSPECTIVES IN THE MARKETPLACE

How the different models operate in the marketplace

Whilst the Open Source and proprietary software development models appear relatively distinct, some have noted a movement towards the middle[4]. This argument holds that whilst the philosophical differences between the open source and proprietary models are substantial, in practice, software developers are beginning to pursue development, licensing, and business strategies that reflect elements of both models. Among proprietary firms, two trends have been noted. The first is to incorporate open source code into otherwise proprietary systems. Apple Computers' use of the FreeBSD kernel within the company's recently launched OS X operating system is cited as one such example. Another is the decision by IBM to offer the Linux operating system across all its servers and ensure interoperability with its software products.

The second trend observed among proprietary firms is the adoption of attributes of the open source model into a broader commercial strategy. For instance, Microsoft's Shared Source initiative is one example of a proprietary company seeking to emulate the benefits of source code access associated with the open source model by giving licensees the right to review – and in some cases modify – the source code for several Microsoft platform products. Microsoft argues that this enhances transparency, making it easier for

[2] http://www.ogc.gov.uk/embedded_object.asp?docid=2498
[3] Check Intellect's paper title
[4] http://www.aei.brookings.org/publications/abstract.php?pid=296

sophisticated customers to debug applications and protect against viruses, fosters deeper understanding of Microsoft products amongst developers and educators, and encourages broad-based collaboration in the development of IT industry standards. Such arguments are not supported by the OSS community.

On the other side of the spectrum, the argument holds that there is growing evidence that a number of firms traditionally identified with open source software are beginning to adopt aspects of the proprietary model. For example, several open source firms are developing and selling proprietary software to complement their open source offerings. In addition, many open source companies are said to have begun modifying standard open source programs in-house in order to meet the needs of specific customers or market segments. These companies are working to adopt what they perceive to be the best elements of proprietary software industry's development, licensing and business models within a basic open source framework.

Evaluating the business case

The selection of Open Source solutions, just as for proprietary software, requires a balanced appraisal of the options, which should include:-

- Value for Money over the whole life of the product (total cost of ownership).
- Support for relevant standards (as detailed in the e-GIF and other government frameworks and standards). This provides the freedom from lock-in.
- Adherence to OGC guidance on procurement. This has been carefully developed using inputs from both the public sector and supply organisations and is constantly being refined to improve the understanding of best practice.

Both public sector and business users should also recognise that:-

a) a spectrum of business models exists, some of which are OSS or include elements of OSS.
b) OSS is currently applicable only to a limited number of solutions, it is not a cure-all.
c) OSS should be assessed in the light of the business structure within which it will be used.
d) a good understanding of the different licensing models is essential to making an informed decision.
e) there may be community-based issues (such as repeat use of public investment) that may influence the decision.

and…

f) the situation is too complicated to leap to the conclusion that one model is inevitably better than another.

CONCLUSION

The different software business models should be able to co-exist and government and politicians should not seek to make broad-based policy preferences for one model over another or attempt to pick winners and losers in the software market. Debate on the respective merits and de-merits of the proprietary and open source software models had become ideological and polarised to the extent that an individual's understanding of OSS was highly dependent on his source of information. A dispassionate observer can see that in the UK, where we are fortunate to have free markets and a level playing field for software procurement decision-making, there is room for both models of software development and supply to co-exist. We strongly recommend that politicians should take an interest in these issues, as with many other developments in IT and technology, they have important implications for economic and societal development.

SECTION II – BACKGROUND

DEFINITIONS AND APPLICATIONS

What is Open Source Software?
Open Source Software (OSS) is software whose source code (the computer instructions that form the basis of a software program) is openly published, is often developed by individuals using the internet to support collegiate behaviour, and often by voluntary effort. It is usually available at no charge under a licence consistent with principles defined by the Open Source Initiative (OSI). OSS may typically be available for download free of charge, but most business users will choose to obtain it from a vendor in supported, packaged form. Although the software is often referred to as "free" the emphasis is on freedom of access (to the source code) rather than free as in "free beer". Broadly speaking, this OSS licence ensures that the intellectual properly rights (IPR) are made available to all. (See *"licensing issues"* below*)*. Linux is an example of OSS.

Proprietary software is software whose source code is, in principle, not openly published, and is owned by organisations who have usually invested significant resource into its development. That software too is protected by IPR and vendors usually strictly limit its redistribution through licences. Only the executable binary code (the strings of 1s and 0s) derived from it is normally made available to the licensees. Proprietary software vendors do not usually provide access to the source code, other than via limited circumstances such as under the terms of escrow agreements or under special licences such as Microsoft's Shared Source initiative. In all case they will retain their intellectual property rights and control. Microsoft Windows is an example of proprietary software.

A number of software suppliers have taken the view that OSS is here to stay, many major proprietary application software companies are now offering Linux-based versions of their software and others are concentrating on adding services to OSS systems and applications. For instance Red Hat is a distributor of OSS and provides commercial services packaging and supporting it. All IBM servers support Linux and applications running under Linux, as do HP. Sun Microsystems also supply open source products and office applications. Vendors of OSS are usually called Distributors because their primary income is derived not from the sale of software, as, depending on the licence, users have the right to share it freely with others - but from packaging, support and training services.

OSS has already taken a significant market share in some parts of the software infrastructure market, such as web servers. It is acknowledged that OSS is, and will remain, a significant and serious competitor with proprietary solutions in many ICT applications (see below).

What kind of applications is OSS used for?
Open Source Software has to date been predominantly used for infrastructure solutions, typically covering the operating system used to run a server, the software used to support internet web-based applications, security protection, shared resources for "file and print" and mail messaging. In these areas it has won a very significant share of the market, for example Linux is now the second largest server operating system by market share (s*ource IDC*), and Apache is the dominant web server application with 64% of all applications *(source, Netcraft Monthly Survey, July 2003)*

OSS is, however, now expanding into other application areas and for other uses, backed up by an increasing choice of development tools. Large enterprises are now piloting its use for mainstream critical application development, although there is little sign of it yet being considered as an alternative for packaged applications. Alternative OSS solutions are now emerging for the desktop, providing competing functionality with office products from Microsoft. There are conflicting reports as to the practical ease of use of these offerings and to date there is no sign of a widespread switch to these offerings by major end user organisations. However, such OSS solutions can be expected to grow in stature and use, but only as they solve issues of integration and end user training. This area of the market is forecast by open source advocates to change substantially over the next 12 months.

LICENSING ISSUES
In developed economies, copyight attaches to all innovative works and this is true of software. All software developers can use their IPR to retain commercial advantage from their intellectual creation and/or to control how their software is used. Contrary to some perceptions, OSS relies on IPR just as much as proprietary software and this is implemented through licensing terms that can be just as restrictive (in their own way). On the whole, proprietary developers use licensing to stop uncontrolled copying and reverse engineering,

and open source developers use licensing to impose “freedom” conditions and the onward inheritance of these to subsequent generation copies. Open Source licences are, on the whole, designed to ensure that the source code always remains available and to prevent people from unfairly capitalising on the efforts of the original individual developers.

A part of the Open Source community, known as the Free Software Federation (FSF) invented a system of copyright, called copyleft. Copyleft requires that, when a program is redistributed, restrictions cannot be added to deny other people access to the source code. Like other forms of copyright, copyleft is implemented through licences and is legally enforceable.

Among the most widely used licences are the General Public Licence or GNU GPL, which governs Linux, the Berkley Software Distribution (BSD) licence which is widely used in the academic community, and Apache, which has a large part of the web-server market. OSS licences loosely fall into two groups, “permissive” and “restrictive”. BSD is termed a permissive licence because it allows the commercialisation of changes made to the original work. GPL is termed a restrictive licence and some critics even describe it as a “viral” licence. By this they are referring to the fact that if OSS covered by GPL is used to develop derivative applications, or is embedded in applications that are used to develop derivatives, then those derivative applications are also subject to the GPL and their source code must also be made available. In this way the GPL “infects” other software and brings it under the same non-commercial licensing terms. However, GPL does not restrict the running of OSS programmes, only copying, distribution and modification. That means that you can use it to run applications freely but once you start using it to build new things, then the terms of the GPL apply and you may lose the IPR to your innovation.

Recently OSI have sought to avoid further proliferation and confusion over different licences by proposing two new forms of licence – a “restricted” version –the Academic Free Licence (AFL) and a “permissive variant – the Open Source Licence. Other than the difference between these two terms, the licences share identical wording, and now both additionally give warranty over provenance, provide a patent licence grant and termination for patent action provision and will be enforceable under contract law as well as copyright law.

As can be expected, the industry has found ways, despite the restrictions cited above, to provide dual licensing for different categories of user, without breaking licence terms, yet providing a viable business model.

HISTORY AND DEVELOPMENT

Schools of thought and ideologies

Some take the view that to get the best software, the source code must be open, and development must be collaborative. The rationale for this is that open code allows everyone to see what is going on (rather like reading the score of a Beethoven symphony rather than just listening to a recording) and that collaboration encourages efficiency and creativity (effort is not wasted in parallel activity). For work to be truly open and collaborative, access to code must be shared and equable – people must desist from protected rights of ownership that prevent other people from seeing what they have done. There is a temptation to take a software application and modify it, or use software to develop a new application and then sell it commercially, with the new applications or modifications being protected by IPR so no-one else can take the same advantage of them. Certain OS licenses seek to ensure that this is prevented because, according to this school of thought, there is no incentive to share expertise if one will not, in turn, benefit from disclosures of others’ changes. However, the significant number of programs that flourish under the BSD licence, which provides complete source code access, yet does not prevent users from restricting access to their changes, suggests that this is only partially true.

The Free Software Federation (FSF) espouses the ideology that software should not be owned by anybody and should be freely available to all, for any purpose except commercial gain from the intellectual creation. Its founder, Richard Stallman, was to a large extent the architect of the concept of “Free Software”.

The Open Source Initiative (the organisation that manages and promotes the Open Source definition) takes a slightly broader view, whilst sharing some of the objectives of the FSF. Stallman has a very strong ideological commitment to the concept of Free Software and firmly disassociates it from that of Open Source, which he views to some extent as compromising the Free Software ideals. Stallman and the FSF could perhaps be seen as Open Source purists.

The term Open Source was initially coined to describe free software but avoid the “radical implications” of "free" which some felt were off-putting to the business community. However, the definitions of OSS and Free Software have diverged although they still might seem very close to an outsider. Free Software is by definition Open Source, but not all Open Source Software is Free Software - for instance, semi-free and even some proprietary programs can be Open Source if the source code is made available, but these would not be considered Free Software by the followers of the FSF. Some of the licences accepted by the Open Source Initiative, such as the BSD, which allows the commercialisation of changes made to the original work, are considered unacceptable by the Free Software community. The distinction is held to be important by the two movements and is explained in more detail in the glossary which provides links to the definitions so they may be compared.

Whilst the objectives of the FSF and the Open Source Movement are to ensure that software will always be accessible and open to everyone, other people take the view that in software, as in any other area of business, the prospect of making economic gain from intellectual creativity is not only a legitimate business practice but also a very effective driver for innovation and economic growth for the good of all. Opponents of the FSF ask why software alone should be singled out for such an anti-commercial approach and how the software industry, a massive employer and contributor to economies worldwide, can sustain its existence and its key role in driving technological innovation if required to apply this approach.

The software industry generally takes the view that there is a place for both models of software development and open source and proprietary software can co-exist very easily whilst playing to the particular strengths of their respective models. As the UK moves increasingly towards a knowledge-based economy, so intellectual property issues become more important.

Where did GNU/Linux come from?
In the early days of software development, Richard Stallman was part of an academic community that had worked on an operating system called UNIX®. He created GNU (Gnu's Not UNIX) as an alternative operating system to UNIX® when the UNIX® source code ceased to be openly available to the software developer community. An operating system contains multiple parts, including a kernel and various other tools and components, and Stallman was originally working on building all these components under his GNU Project when a student called Linus Torvalds, working independently of the GNU project, created a kernel called Linux that was compatible with the other GNU project elements. This was timely because the GNU kernel was developing more slowly than other GNU project elements, so the Linux kernel was used instead. This suggests that today's GNU/Linux operating system is a mix of elements created from the GNU project plus the Linux kernel created by Torvalds.

Linus Torvalds is regarded as the final arbiter of Linux, which means that in theory he decides which changes are adopted and how the software develops. In practice, however, there is no evidence that Torvalds has been put into, or has put himself into that position. Some say that the existence of a final arbiter for Linux prevents the system from forking into multiple versions and suffering the same fate as UNIX® which at one stage fragmented into incompatible proprietary versions (although now all UNIX® versions are warranted to conform to set standards). However, others feel that there are disadvantages to a single arbiter who in theory could be seen as a bottleneck to development, or for whom the responsibility could become a burden, although again, in practice this has not yet appeared to be the case. Torvalds, for instance, uses a set of trusted committees, who are responsible to him but operate fairly autonomously.

If government does move towards the increased use of OSS then government must consider the need to put structures in place for arbitrating development decisions when OSS solutions are to be shared, to prevent incompatible or parallel versions being developed.

PERSPECTIVES FOR AND AGAINST OSS

Total cost of ownership
It is generally accepted that OSS is available at much lower cost than proprietary software: it can often be obtained free of charge, and even in packaged form it is relatively cheap, and it may be run on lower cost hardware. However, the cost of purchasing or licensing software alone is only a proportion – often a fraction - of the total cost of ownership, which includes integration, maintenance and support costs. Service and running costs have to be added to the cost of OSS when not in packaged form. In some cases it is also necessary to factor in potential revenue from future IPR when considering options.

User control
For many proponents of OSS the primary advantage of OSS over proprietary software is that the user community has more control of the software. Users are not tied to a vendor's upgrade cycle, can tailor their

software as they like and need not worry about obsolescence because they can update it themselves, provided they have the skills to develop it.

Pace of Development
Sharing amongst a large community of users, easily done with the WorldWideWeb, means that OSS has the potential to evolve very fast indeed. However, the pace of evolution of OSS cannot be predicted, and improvements are added on an ad hoc basis. On the one hand that means that improvements can be made to the software and new versions issued whenever they are needed, but on the other, if the community loses interest in that software, then development may slow or even cease because it depends on a critical mass of users, many of whom are volunteers. Although they are open to feedback from users on bugs and problems, proprietary software vendors tend to issue upgrades periodically but not frequently, although they will often issue interim "patches" to solve specific problems. In the past this has left sophisticated users feeling frustrated – they encounter a problem but cannot fix it for themselves since they do not have access to the source code. On the other hand proprietary software producers are committed to regular upgrades, are moving to a process of continuous improvements, available to customers online, and usually provide a consistent, high level of support for the lifetime of the software.

Functionality and User-Friendliness
Some of the frustrations voiced by individuals about proprietary systems are that many home users only want and need very limited functionality – their computer usage involves web browsing, writing a few letters and possibly adding up figures for their tax return. Most proprietary products are far more elaborate than people need, but the perception is that cheap versions with very limited functionality and the same user-friendly interface are not available. Critics say it is the equivalent of wanting a bicycle to nip up to the post office twice a week, but all that is on offer is a Rolls Royce. Bicycles aren't available. They argue that OSS has made software "bicycles" available, but because the community that has traditionally used OSS is very computer literate, they have not come in user-friendly formats. In other words, the bicycle arrives in bits and needs to be put together. You will need the equivalent of a chain de-linker and a knowledge of gearing systems. Getting software in a basic format is no problem for the cybernerd who can build his applications from the basic components, but presents severe difficulties for the average user who wouldn't know where to start. This is where new distributors (like Red Hat) come in, since they package the open source software into a user-friendly format, so that it can be installed onto the desktop (from a CD-Rom for instance) and they may offer support services. Similarly, there is a rapidly developing role for integrators. This obviously has a service cost attached which has to be weighed up against the cost of purchasing proprietary software.

Testing
Software testing is a very expensive and time-consuming business and the proprietary software sector employs highly trained developers who are paid to spend their working time testing and reviewing software to eliminate bugs (faults) and prevent illegal security breaches. OSS, however, depends on volume adoption to shake out the bugs - on the basis that the more people are allowed to test the software, the more opportunity there will be to find bugs. Everyone can look at the source code, can tinker about and suggest improvements or solutions (patches and fixes) and try them out, then submit their suggestions to the author (inventor) of the software who usually acts as arbiter. Provided that lots of people test the software in this way the software can become very robust because it is thoroughly scrutinised. Some critics of this "many eyeballs" approach to security point out that without a commercial incentive de-bugging is left somewhat to chance, relying on the hope that volunteers will review the correct piece of software, find the bug and have the necessary skills to address the problem. However, there is no evidence that OSS is of lower quality than proprietary offerings. It is worth noting that recent research has shown that many proprietary sector developers work either in their own time, or funded by their employer, on OSS development and this is likely to further improve the reliability of OSS offerings.

To an extent the OSS testing model rests on reputation over time, so people can be relatively confident in OSS like Linux and Apache but should test out most OSS very carefully and extensively in the application environment before relying on it.

Support
Whilst proprietary vendors invest in extensive training programmes to ensure a steady supply of support skills in appropriate applications, OSS support can be fragmented or difficult to obtain, because there are fewer trained administrators for open source systems. However, this problem appears to be diminishing as an increasing number of large suppliers (eg IBM, Sun and HP) are investing heavily in supporting OSS. The same can be said for supporting documentation, which is another issue that should diminish as OS solutions become more mainstream.

Security
OS proponents believe that OSS is less vulnerable to attacks and hacking than proprietary software because it was developed via the internet under real conditions of exposure, and, because the code is freely available for inspection to a large peer group, weaknesses can be quickly overcome in early design. Although there are fewer recorded instances of successful attacks on OSS, it is not such an attractive target as proprietary systems, so it may not be subject to a comparable number of attacks. Also, just like other aspects of development, locating security flaws successfully through voluntary effort depends on a critical mass of users. Opinions are divided as to whether OSS is inevitably more secure than proprietary software programmes.

Stability
There is a lingering perception that OSS is produced by men in flip-flops living in beach huts in California, and that therefore the stability of OS offerings depends on the state of the surf. This may just possibly be true of some OSS offerings – generally the ones that never see the light of day. It is certainly not the case for mainstream products where global communities are involved in their development and testing.

Due Diligence
Some critics are concerned that whilst a customer who runs an OSS program has no obligations to the author, he has no rights either and there is no guarantee that the code he is using is free from copyright or other IPR infringement, or liabilities to third parties. With no single developer there is no obvious way to protect supply and satisfy auditors. The current litigation called by SCO[5] is a case in point.

Secondly, OSS does not cover contingent liability – i.e. the consequences of software failure. Proponents of OSS say that this is a practical necessity for the developer, who cannot know how and where his prodigy will be used, that proprietary licenses are also exclusive, and cite the Java licence which advises against its use in mission-critical situations and excludes liability for the consequences of failure. However, a paying customer in an identified relationship with a proprietary software vendor is afforded some cover under legislation such as consumer protection law, so proprietary vendors cannot disclaim liability entirely.

Interoperability and Portability
OSS is supplier-independent and is not tied to one supplier because it is developed and maintained through collegiate working by individuals. This makes it portable to a wide range of platforms, which gives greater choice and enables users to avoid proprietary lock-in. That said, proprietary vendors do co-operate with one another to provide services, and frequently support standards such as the e-GIF which provide purchasers with protection from proprietary lock-in. Proponents of OSS believe that it also allows easier pooling of applications amongst public sector organisations although there is as yet no framework within which to achieve this.

Open Industry Standards
OSS is built around open industry standards although source code availability is not synonymous with open standards, while proprietary vendors are not traditionally associated with open standards. However, proprietary vendors are increasingly supporting open standards that support interoperability, such as W3C XML. There is frequently some confusion around what, precisely, the terms "open standard" and "open source" mean in the consideration of interoperability. Although these two terms are frequently used in close conjunction they are not synonymous. A recent submission from the Coordinating Committee of Business Interlocutors (CCBI) to the World Symposium on Information Society (WSIS) recently stated these differences as follows:
"an open standard is a technical specification whereas open source is a software development model, which, like any other software development model, may or may not implement open standards".
"standards do not require either proprietary or open source software for their adoption or utility, and in some cases may combine technology or intellectual property developed under both software development models. Furthermore, when these standards are open and available to all through reasonable and non-discriminatory licensing, they help all developers create products that interoperate with each other."

[5] SCO's lawsuit against IBM, filed March 2003 – see http://www.sco.com/ibmlawsuit/amendedcomplaintjune16.html

SECTION III – GOVERNMENT POLICY

WHAT IS BEING DONE AND BY WHOM?

Current situation – what is the policy?
Both the European Commission and the UK Government are positively encouraging the consideration of open source software (OSS) for appropriate applications alongside proprietary software. The EU supports the use of open source in the e-Europe Initiative *An information Society for all* [6] and makes this clear in the *June 2000 Action Plan*[7] , which states "during 2001 the European Commission and member states will promote the use of open source software in the public sector and e-government best practice through exchange of experiences across the Union (through the IST* and IDA* programmes)".

The UK Government wants a "level playing field" and "acknowledges the competitive viability of OSS solutions". Government policy[8] states that OSS solutions will be considered alongside proprietary ones in IT procurements, with contracts being awarded on a value-for-money basis, that Government will only use products that support open standards and will seek to avoid lock-in to proprietary IT products and services, and that Government will consider obtaining rights to software code where this achieves value for money. UK Government has already mandated open standards and specifications in its e-Government Interoperability Framework (e-GIF).

How is policy being implemented?
The UK Government's policy was well received by the public sector and industry and implementation is fully underway. UK Government and European Commission OSS activities are summarised below.

<u>Implementation of OSS policy in the UK</u>
UK Government is actively implementing policy through a number of initiatives:

- A UK Public Sector OSS special interest group (SIG), run by OeE with the close involvement of the Office of Government Commerce (OGC) has been established to raise awareness of the OSS policy and build competence in procurement in the public sector. It has comprehensive public sector representation and is supported in its work by the procurement guidelines published by the OGC[9].

- The OeE is working closely with DTI to explore with academia and industry the possibilities of using OSS as a default exploitation route for Government funded R&D software. The findings of this exercise will feed into future policy formulation.

- OGC commissioned a case-study based report[10] from QinetiQ to provide a documented methodology for approaching OSS case studies which would enable OGC (or any other party wishing to study OSS implementations) to approach case studies in a consistent manner.

- An OSS Community Special Interest Group has been established with the objective of keeping UK government informed on relevant issues in the OSS community.

The government's OSS Policy will be revised when appropriate to reflect developments.

Recent research by Open Forum Europe has indicated, however, that whilst the Policy document is well known, little pragmatic support or leadership is perceived as being available into how and where OSS could most effectively be considered.

<u>Implementation of OSS policy in the EC</u>
In the EC OSS policy is being implemented principally through IDA (Interchange of Data between Administrations) and the Sixth Framework.

IDA is a European Commission driven strategic initiative using advances in information and communications technology to support rapid electronic exchange of information between Member State administrations. The

[6] http://europa.eu.int/information_society/eeurope/index_en.htm
[7] http://europa.eu.int/information_society/eeurope/action_plan/pdf/actionplan_en.pdf
[8] http://www.e-envoy.gov.uk/oee/oee.nsf/sections/frameworks-oss-policy/$file/oss-policy.htm
[9] see http://www.ogc.gov.uk/index.asp?id=21908&
[10] QinetiQ study: see http://www.ogc.gov.uk/index.asp?id=21908&

objective is to improve Community decision-making, facilitate operation of the internal market and accelerate policy implementation. Initially, IDA helped set up infrastructure, establish common formats and integrate new ICT based business processes. It is now improving network services, tools, security and interoperability. IDA action lines specifically relevant to OSS proposed for the 2003 IDA Work Programme are:

- The establishment of competence centres for OSS:- the key output would be an improved information base on the usage of open source software in Europe's public sector, to promote the spread of good practice in the use of open source software by public administrations. An information focal point on OSS will be established under the eGovernment observatory, detailing the lessons learnt, providing specialist technical and economic advice on specific issues, encouraging contacts between national and regional initiatives through workshops and leading to, at a later stage, the creation of an inventory of government-sponsored applications. This proposed action would build on the work of the Feasibility Study on the pooling of OSS published in June 2002*.

- Open Source migration assessment:- the objectives are to continue to examine the costs incurred in changing from a proprietary-based office IT infrastructure to Open Source Software; and to define a reference OSS desktop PC environment. The costs and benefits of deploying Open Source Software would be addressed via case studies of administrations contemplating a migration to OSS. Recommendations and guidelines to public administrations wishing to migrate to open source software are due in summer 2003[11].

- OSS-based e-Learning tools:- the objectives are to consider emerging specifications, define functional requirements and survey available open source software-based tools for e-learning within a new and fast-developing e-learning market.

The Sixth Framework Programme for research and technological development is a major strategic tool to support the creation of the European Research Area (ERA) which aims to provide a structure to coordinate research activities and converge research and innovation policies at both EU and national levels. Within the Programme, a significant number of research projects are using or developing OSS solutions.

Other National Government Approaches to OSS

While most national governments take a pragmatic approach similar to the UK, some have taken a more proactive approach to supporting OSS as the preferred procurement model. In Europe, Germany is perceived to be taking the strongest line and has implemented a number of supporting initiatives. A Ministry of the Interior agreement was signed in June 2002 to give special conditions for administrations buying Linux/OSS and OSS is now used widely across both the state and local governments. Other countries have provided practical support via the establishment of regional capability centres. The capability centre in the Netherlands (TBC) and the Extramadura region in Spain are examples of this approach.

Proponents of OSS report vocal support in nations where there has not been a strong history of widespread use of IT. For those who take this view, the driver may well be political, on the basis that OSS provides the most effective re-use of community resource and public investment, as a fiscal opportunity to reduce balance of payment issues by reducing the license spend exiting the country or to reinvigorate or develop an internal software industry, recognising the lower cost base and market cost of entry. This view is not shared by the proprietary vendor community.

[11] see http://www.ogc.gov.uk/index.asp?id=21908&

SECTION IV – GLOSSARY and INDEX

Binary Code is the string of 0s and 1s that make up the instructions that a computer can understand. Computers work in Binary – powers of 2, whereas normally we work in Decimal – powers of 10. Hence 234 in binary is $2 \times 2^2 + 3 \times 2^1 + 4 \times 2^0 = 8 + 6 + 4 = 18$ whereas in decimal this number is $2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 = 200 + 30 + 4$. Binary code suits electronic equipment because of the on-off / charged-uncharged / positive-negative associations.
See http://www.webopedia.com/TERM/b/binary.html

BSD stands for ``Berkeley Software Distribution'' and it is an open source license that is more permissive than the GPL or General Public License because it allows the commercial exploitation of changes to the original work. It is widely used in the academic community. See http://www.opensource.org/licenses/bsd-license.php for a copy of the BSD licence.

A **Compiler** is a program that translates Source code into Object code which can then be translated into Machine code. Source code is the basic language in which the instructions are written by the programme. Machine code is the language that computers can understand in order to execute the instructions and object code is a kind of intermediary language. The compiler collects and reorganises the instructions in the source code to produce object code. Sometimes computers can read object code because it is close enough to the machine code that they understand. Other times the object code needs to be converted to executable machine code by using programmes called assemblers, binders, linkers and loaders.
Compilers are specific to certain computer types and programming languages. A particular compiler, therefore translates a certain kind of programming language into code that is readable on certain kinds of machines. This is a bit like a translator that can translate German into Russian as opposed to one that can translate German into French or French into English, except that there can be an intermediate language involved too.
Source: http://www.webopedia.com
For more information see: http://www.webopedia.com/TERM/C/compiler.html

Copyleft is a general method for making a program free software and requiring all modified and extended versions of the program to be free software as well. Copylefting ensures that anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it.
Source: http://www.fsf.org/copyleft/copyleft.html

Distinction between Open Source and Free Software Open Source and Free Software are clearly differentiated – see http://members.optushome.com.au/brendansweb/opensource/ for links to definitions of both, or go to http://www.fsf.org/philosophy/free-sw.html for Free Software definition and to http://www.opensource.org/docs/definition.php for Open Source definition.

Dual Licensing
Dual Licensing is based on the simultaneous use of both open source and proprietary licenses and has emerged as a way to maintain sustainable sources of revenue whilst using an open source development and licensing model. The same software can be available under two different licenses – a GPL type licence where the copyleft extends to derivatives and a proprietary licence where there is no such restriction. Dual licensing is explained in much more detail in Mikko Välimäki's paper *Dual Licensing in Open Source Software*, published in Systèmes d'Information et Management, Vol 8 No. 1 pp 63-75, 2003.
http://www.hiit.fi/u/valimaki/dual_licensing.pdf

ERP is short for *enterprise **r**esource **p**lanning,* a business management system that integrates all facets of the business, including planning, manufacturing, sales, and marketing. As the ERP methodology has become more popular, software applications have emerged to help business managers implement ERP in business activities such as inventory control, order tracking, customer service, finance and human resources.
Source: http://www.webopedia.com See: http://www.webopedia.com/TERM/E/ERP.html

e-GIF is short for the Electronic Government Interoperability Framework. e-GIF is a set of policies and standards to enable information to flow seamlessly across the public sector and provide citizens and businesses with better access to public services. e-GIF covers the integration of data across government departmental boundaries, interconnectivity of systems, access to information and management of all information content. This encompasses information interchange, how you describe data (use of meta data is covered by the e-GMF; electronic Government Metadata Framework), and the use of Internet standards (HTTP, SOAP, HTML, XML etc).
Source: Tessella - http://www.tessella.com/literature/articles/otherarticles/egif.htm

Escrow Software escrow applies where a trusted third party (TTP) to customers and suppliers holds software safely. In a volatile market, customers want to know that if something happened to their supplier, they could still use the software, so the TTP keeps it and is able to ensure that the software wills till be available even if the original vendor no longer exists.
Escrow also applies more generally to ensure buyers and sellers fulfil business agreements, particularly when paying online. Instead of paying a seller directly, buyers pay an escrow company, which then shuttles the payment to the seller-only after the buyer receives and approves the goods.
Source: http://www.i-escrow.com/

A **Firewall** is a system designed to prevent unauthorized access to or from a private network. Firewalls can be implemented in both hardware and software, or a combination of both. Firewalls are frequently used to prevent unauthorized Internet users from accessing private networks connected to the Internet, especially intranets. All messages entering or leaving the intranet pass through the firewall, which examines each message and blocks those that do not meet the specified security criteria.

Source: http://www.webopedia.com See also: http://www.webopedia.com/TERM/f/firewall.html

Forking happens when software develops down divergent paths, or when offshoots of the software develop The result may be independent products that are not interoperable. This happened to UNIX where different proprietary versions developed.

Free Software is software, where the user has freedom to run, copy, distribute, study, change and improve the software. The concept of Free Software is an ideology, around the idea that software should not be owned by anyone. Although zero cost is implied, the "Free" should be interpreted in the same way as "Free Speech" rather than "Free Beer". The Free Software Federation say that it is free software if, "as a user:-

- You have the freedom to run the program, for any purpose.
- You have the freedom to modify the program to suit your needs. (To make this freedom effective in practice, you must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.)
- You have the freedom to redistribute copies, either gratis or for a fee.
- You have the freedom to distribute modified versions of the program, so that the community can benefit from your improvements.

The "free" refers to freedom, not to price so there is no contradiction between selling copies and free software."
Source: http://www.fsf.org/gnu/thegnuproject.html See http://www.fsf.org/philosophy/free-sw.html

The **Free Software Federation (FSF)** is the principal organisational sponsor of the GNU project with a mission to preserve, protect and promote the freedom to use, study, copy, modify and redistribute computer software and to defend the rights of free software users. See: http://www.gnu.org/

GNU / GNU Project
The GNU Project was launched in 1984 to develop a complete Unix-like operating system which is free software: the GNU system. (GNU is a recursive acronym for "GNU's Not Unix"; it is pronounced "guh-NEW".) Variants of the GNU operating system, which use the kernel Linux, are now widely used; though these systems are often referred to as "Linux", they are more accurately called GNU/Linux systems.
Source: http://www.gnu.org/

GPL or General Public Licence, usually referred to as GNU GPL. This is a specific example of a free software licence and is designed to ensure that the source code for the software stays in the public domain, using copyleft clauses. The GPL requires that any changes to the original work also take on the same licence terms. This means that anyone distributing OSS, making modifications or improvements to OSS or using OSS to create derivative products must make the source code of such derivatives publicly available. Some view this as a restrictive licence because it restricts the opportunity to benefit commercially from developing the software. Others would say the opposite, because it ensures unrestricted access to the source code. See http://www.fsf.org/copyleft/gpl.html for copies of the GPL.

Hacker in this context is used to describe a member of the software developer community, its original meaning, and NOT someone trying to hack illegally into systems. e.g. Richard Stallman, founder of the Free Software Federation and architect of GNU, is a hacker.

IDA – Interchange of Data between Administrations. IDA is a European Commission driven strategic initiative using advances in information and communications technology to support rapid electronic exchange of information between Member State administrations. The objective is to improve Community decision-making, facilitate operation of the internal market and accelerate policy implementation.
Source: http://europa.eu.int/ISPO/ida/jsps/index.jsp?fuseAction=home

Interoperability is the ability of systems or software to "talk to" other systems, either directly, because they are compatible, or via an interface or portal. Interoperability eliminates repeated data entry (with consequent risk of error) because information held in one system can be accessed via another system. Interoperability relies on data management standards (ownership, security, quality, sourcing) and interface standards (format, protocols and process). e.g.The Police National Computer (PNC) is now linked to a Motor insurance and Driver databases so that police can check insurance and licence records of cars directly via the PNC. Local Authority One-Stop-Shops are an example of limited interoperability where information from different systems is available through a portal or screen, but the information is not combined and is held separately.
Source: EURIM Briefing 36: Interoperability – Joined Up Government Needs Joined Up Systems
http://www.eurim.org/briefings/BR36_final.htm *See also* e-GIF definitions.

Incompatible versions – software versions that are no longer interoperable.

An **Interpreter** is a programme that translates programming language from the original code in which it is written to an intermediate code and then executes it, so there is no need for machine code.
See: http://www.webopedia.com/TERM/i/interpreter.html

IPR - Intellectual Property Rights – the system for protecting ownership of a creation that does not exist in tangible form. Such rights are protected through patents, copyright, trademarks, etc.
Source: Oxford English Reference Dictionary

The **Kernel** is the central module of an operating system. It is the part of the operating system that loads first, and it remains in main memory. Because it stays in memory, it is important for the kernel to be as small as possible while still providing all the

essential services required by other parts of the operating system and applications. Typically, the kernel is responsible for memory management, process and task management, and disk management.
Source: http://www.webopedia.com see: http://www.webopedia.com/TERM/k/kernel.html

Linux – Pronounced lee-nucks or lih-nucks. A freely distributable open source operating systemr that runs on a number of hardware platforms. The Linux Kernel was developed mainly by Linus Torvalds. Because it is free and it runs on many platforms, including PCs and Macintoshes, Linux has become and extremely popular alternative to proprietary operating systems that has gained popularity because of its stability as an operating system, particularly for hosting web servers. Linux is freely available over the Internet.
Source: Webopedia - http://www.webopedia.com/TERM/L/Linux.html

Meta Data is data about data – how you describe data. The use of meta data is covered by the e-GMF; electronic Government Metadata Framework. The e-Government Metadata Standard lays down the schemes to be used by government agencies when creating metadata for their information resources or designing systems to search their information. The e-GMS is needed to ensure maximum consistency of metadata across public sector organisations. The GCL (Government Category List) is a classified list of headings for use with the subject element of the e-GMS. All electronic documents, web pages and other resources in the public sector should all have the relevant terms in their metadata.
Source: http://www.boxuk.com/server/show/ConWebDoc.183/cms/xml

Open (as in open source software or open standards) is meant in the sense of fulfilling the following requirements:
- the costs for the use of the standard are low and are not an obstacle to access to it;
- the standard has been published;
- the standard is adopted on the basis of an open decision-making procedure (consensus or majority decision etc);
- the intellectual property rights to the standard are vested in a not-for-profit organisation, which operates a completely free access policy;
- there are no constraints on the re-use of the standard.

Source: Dutch Programme for Open Standards and Open Source Software in Government (OS/SOS)

Open Forum Europe is a new initiative, whose objective is to accelerate and broaden the market take-up of Open Source Software (OSS) including Linus. Major vendors and distributors, software houses, services and integration companies, as well as major users are all supporting the project.
Source: http://www.openforumeurope.org/

The **Open Source Initiative** is a non-profit making organisation dedicated to managing and promoting the Open Source Definition, specifically through OSI certified software.
See http://www.opensource.ac.uk/mirrors/www.opensource.org/

Open Source Software (OSS) is software whose source code is openly published, is often (but not always) developed by voluntary efforts and is usually (but not always) available at no charge under a licence defined by the Open Source Initiative (OSI) which prevents it from being redistributed under a more restrictive licence.
OGC Guidance on implementing OSS defines it as "Software where the source code (the language in which the program is written) is freely distributed with the right to modify the code, and on the condition that redistribution is not restricted, and indeed is obtainable for no more than the reasonable cost of reproduction" See http://www.ogc.gov.uk/embedded_object.asp?docid=2498 OGC also note: In contrast, vendors of closed, proprietary, software provide only executable binary code, and not the human readable source from which that code is derived. Proprietary software vendors usually also place very specific limits on redistribution of the software" . e.g. Linux is an open source operating system. Apache is open source.
Source: OGC, OeE. See also OSI website definition: http://www.opensource.org/docs/definition.php

An **Operating System** is the underlying technology that enables software applications to run on the computer hardware – the interface between them. The operating system is the most important program on a computer and enables the computer to run other programs. It also performs basic tasks like recognising input from the keyboard, sending output to the screen, and keeping files and programs separate. e.g.Microsoft Windows is an operating system.
Source: http://www.webopedia.com , see http://www.webopedia.com/TERM/O/operating_system.html

Permissive Licence - tends to describe a licence that allows commercial benefit to be gained from derivatives or modifications to OSS whilst ensuring that the original source code remains open.
It is worth noting that the confusion over licensing is not helped by the descriptive terms used. For instance, very different types of licence are described as "restrictive" depending on perspective - FSF proponents believe that proprietary licences are restrictive because they restrict freedom to use the software. In contrast, others would term the OSS licences restrictive because they restrict the commercial benefits that use or development of the software can provide.

Proprietary Lock-in or Vendor lock-in is a situation in which a customer is dependent on a vendor for products and services and cannot move to another vendor without substantial costs. This is often a result of incompatibility between different computer systems which intentionally or unintentionally force a customer to continue to use products and services from a particular vendor. Open Standardisation processes are meant to provide a way to reduce the risk of vendor lock-in when creating new standards. One argument for OSS is that it reduces the risk of vendor lock-in.
Source: Wikipedia - http://en.wikipedia.org/wiki/Vendor_lock-in

Proprietary Software is software that is privately owned and controlled. A proprietary design or technique is one that is owned by a company. It also implies that the company has not divulged specifications that would allow other companies to duplicate the product. Generally proprietary software is software for which there is no access to the source code, although there are exceptions (See below). Some refer to proprietary software as **closed source.**

For the purposes of this EURIM paper, we differentiate between 'proprietary' and 'commercial' software. NB - these two terms are not necessarily related, even as opposites. They describe different states. We see 'proprietary' software as where the ownership of the intellectual capital in the software is vested in an individual or company as their property. They may well make this property openly available for use free of charge by others, while retaining their ownership. Owners of proprietary software may keep the technical detail of the software code private to themselves and only license its use to customers, or they may choose another relationship with users - exposing the source code to some while keeping it secret from others and with a range of payment options - while still retaining the intellectual property.

We see commercial as relating to software - whether based on proprietary code or on open source code - which forms the basis of a commercial transaction. Thus 'commercial' could apply to the terms of use of software developed using LINUX, depending on the product licence terms. More often it will apply to proprietary software made available on normal business terms. An example of proprietary but non-commercial software is JAVA language, which has been made openly available to software developers worldwide at no charge, while remaining the property of Sun Microsystems (TM).

Proxy Servers are servers that act as interfaces between users and other servers, such as web servers. They intercept requests before they reach the server and answer them if they can, if not they send the request on to the server. In this way they act as a memory cache, a bit like the temporary internet files on a PC, but much bigger, and this speeds up access to information. They can also act as filters to prevent users accessing certain kinds of website.

"**Radical Implications** of Free Software". The concept of "free software" was seen by some to have radical implications as part of a larger "free" or anti-capitalist movement. It was also felt that people in business would underestimate free software simply because it was free, on the mistaken assumption that anything that was a giveaway couldn't be any good. Hence the term "open source" was developed to play down these associations whilst reinforcing the concept of access to the source code. It also reflects better the reality that the software is not necessarily free of cost.

Restrictive Licence:- Some use the term restrictive licence to describe a licence like the GPL which restricts the way in which the software can be used commercially because further work on the software to develop it, make modifications or derivatives cannot be protected by IPR. However, others take the opposite view, and would describe proprietary licences as restrictive because they limit the ways in which software can be used, and access to the source code is restricted.

A **Server** is really a computer that provides some kind of service for other computers on a network. There are different kinds of server – print servers are computers that manage printers, file servers are computers that manage lots of files. Because other computers (and often whole networks) rely on them, servers are usually high specification, and may also have built in contingency setups, for example back up power sources should there be a power failure.

Source code is the original instructions for a programme, and is the only format that is written in a language that can be understood by humans. In order for it to be read by machines, source code has to be translated into Executable code, or sets of instructions that a machine can carry out. Machine code is an executable code. Usually, the code that is supplied with software is in a machine language format which means that they will automatically give instructions that the computer hardware understands and can execute, but cannot be read or modified. Source code is readable and can therefore be modified. See www.iunknown.com/Files/Test.pdf for an example of some source code.
Source: http://www.webopedia.com *See* http://www.webopedia.com/TERM/s/source_code.html

UNIX (pronounced *yoo-niks)* is an operating system first developed at Bell Labs in the early 1970s. UNIX was designed to be a small, flexible system used exclusively by programmers and as a result it has traditionally been user-unfriendly. UNIX was one of the first operating systems to be written in a high-level programming language, namely C, which made it very portable because it could be installed on virtually any computer for which a C compiler existed. This natural portability combined with its low price made it a popular choice among universities. (It was inexpensive because antitrust regulations prohibited Bell Labs from marketing it as a full-scale product.) Bell Labs distributed the operating system in its source language form, so anyone who obtained a copy could modify and customize it for his own purposes. By the end of the 1970s, dozens of different versions of UNIX were running at various sites.
Source: http://www.webopedia.com See: http://www.webopedia.com/TERM/U/UNIX.html

Value Added Services Provision means the inclusion of support and other services with the software.

W3C XML stands for the World Wide Web Consortium Extensible Markup Language. Extensible Markup Language is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.
Source: http://www.w3.org/XML/

A **Web Server** is a computer that delivers web pages. Any computer can become a web server by installing server software and connecting it to the internet. There are many web server software applications such as Apache, Microsoft or Netscape. When you search for a website using a browser it requests it from the web server which then finds and serves up the home (index) page unless a more specific page is requested.
Source: http://www.webopedia.com