



# OPEN SOURCE AND FREE SOFTWARE

**Andrew Bowen**  
**TESSELLA SUPPORT SERVICES PLC**

Issue V1.R1.M2  
July 2004



## Introduction

Open Source Software (OSS) has often been characterised as the poor relation to proprietary software, being the preserve of hobbyists rather than professionals. Nothing could be further from the truth. OSS underpins many key components of the Internet and many of the contributors to OSS are professional software developers and organisations. OSS can no longer be said to be only an out-of-hours pastime for enthusiasts as more and more commercial organisations, including the likes of IBM, take strategic decisions to embrace OSS.

As OSS continues to rise in importance, the IT industry is becoming polarised into its supporters and opponents. There are many commercial interests at stake and market share to be won or lost. In this climate there is a considerable amount of propaganda and posturing on both sides that can make it difficult to see where the benefits and drawbacks of OSS lie.

As a way to put this discussion into context this supplement will consider the relative advantages/disadvantages of OSS as a whole and then look at some concrete examples of real OSS used in IT infrastructure and software development. But first software licenses and what the term Free Software (FS) means will be discussed.

## Common Licensing Methods

Before beginning a discussion of Open Source Software it is worth briefly describing the many software licensing methods currently in use. As with any legal agreement the devil is in the detail, so it is essential to examine each and every licence as they can differ significantly. However, it is possible to classify software-licensing methods into a number of broad categories.

### **Proprietary (commercial) Software**

Ordinary proprietary software is what many organizations will be most familiar with. Licence agreements carefully stipulate terms of use and prohibit re-distribution in any form. Often users are obliged to purchase original media from authorized distributors, although increasingly commercial products are available to download and purchase from the Internet. Source code is almost never made available.

## **Shareware**

Shareware is proprietary software. Users must agree to the terms of a licence and pay a fee in order to use the software (usually after a free evaluation period). The 'share' in shareware is simply a reference to the means by which the product is distributed – generally anyone is entitled to re-distribute the software (usually the evaluation version) but licence fees are still paid to the owners of the copyright. Source code is almost never available for shareware.

## **Freeware**

Freeware is also proprietary software. Users must agree to the terms of a licence but they do not have to pay a fee for using the software. There are sometimes restrictions on the number of free licences that an organisation is entitled to use. Re-distribution is normally also prohibited or severely restricted. Although freeware seems like a good deal, the company supplying it will almost always have commercial interests at heart in one way or another. Source code is almost never made available for freeware.

## **Open Source and Free Software**

OSS/FS is also subject to a licence and the author/s will retain the copyright. Suppliers are entitled to charge for supplying the software to you, although in practice the software is often free to download from the Internet. The important feature of OSS/FS is that you are also entitled to a copy of the full source code. You are then at liberty to use the source code to understand/enhance the software, and, if you choose, supply modified versions to others. There are differences between Open Source Software and Free Software which will be described later.

## **Public Domain Software**

When authors put software into the public domain they give up all rights to the software. Anyone is then entitled to use it or modify it for any purpose, without restriction of any licence. This would appear to be the most libertarian approach, but as supporters of OSS point out, software in the public domain can be modified and then incorporated into a commercial product making it proprietary without any acknowledgement (and thus keeping any enhancements out of the public domain). Depending on which country you are in 'Public Domain' may not even have any legal meaning.

## **When access to source code does not mean Open Source Software!**

Some organisations have taken the step of making some source code for commercial products available for others to look at. The intention can be to make life easier for partners and developers in creating software to work alongside a commercial product, and can be a useful step. Sometimes the intention may be to get free assistance from others in locating bugs in a commercial product! Often, however, the disclosure of source code is

piecemeal and contributes relatively little to openness. Such source code remains the property of the organisation and cannot usually be modified or re-distributed in any way.

## The Open Source Initiative and The Free Software Foundation

### The Free Software Foundation (FSF)

In the mid 1980s a new movement appeared within the IT community, that was dedicated to openness and freedom in the development of software. The Free Software Foundation (FSF), more commonly known by the name of its best known project GNU (**GNU's Not Unix**), was formed to champion the development of Free Software for the greater common good. 'Free Software' was the phrase chosen to describe the means by which the software would be licensed to others, 'Free' being a reference to the liberties granted to the licensee rather than the price of the software (although Free Software is commonly available for little or no charge). The basic freedoms that are guaranteed to the user of Free Software can be summarised as follows:

1. Freedom to run the software and use it for any purpose
2. Freedom to re-distribute the software to anyone else
3. Freedom to examine the software to find out how it works and modify it in any way
4. Freedom to distribute modified versions of the software to anyone else

Clearly, in order to be able to examine the software and modify it the source code must be available. So along with the freedoms granted with a Free Software licence there are accompanying obligations, the most important of which are:

1. The authors of the original, or any subsequent modified version of the software, must make the source code available to anyone to whom they have supplied a copy of the software.
2. For some licenses (e.g. GNU GPL), the licensee of Free Software must pass on exactly the same licence conditions to everyone else when distributing the original or any modified copy of the software.

The second obligation is often called '*copylefting*' and guarantees that Free Software can never be incorporated into a proprietary product and become 'closed'. There are several licences in use that satisfy the principles of Free Software, but by far the best known is the GNU General Public Licence or GPL. There are many other more permissive Free Software licenses in use that do not include '*copylefting*' clauses.

## Open Source Initiative (OSI)

At the end of the 1990s the Free Software movement split into two camps. Free Software was felt by some to have an image problem in commercial companies and so a new organisation, the Open Source Initiative (OSI), was formed to promote the advantages of Open Source Software development. The OSI makes the case for collaborative open source development in the commercial world. The benefits that they cite are numerous and tangible; these will be described a little later. FS always qualifies as OSS but the reverse is not necessarily true.

In practice there is a great deal of common ground between the Free Software and Open Source movements, as they arose out of the same roots, however there are differences in philosophy and motivation between the two groups. The FSF champions the cause of the non-proprietary software through the freedoms exemplified by the GNU GPL, whereas the OSI makes the case for the open collaborative method of software development. You can find out more about the beliefs and aims of the FSF and OSI on their respective web sites. It is however important to note that the two organisations still collaborate on projects and are to all intents and purposes 'on the same side' with respect to proprietary software.

## Licences – The Legal Detail

The OSI and FSF websites list over 50 different Open Source and Free Software licences. The aims of all of these licences are broadly the same: to make software free in its use and open to scrutiny, modification and re-distribution. However, with over 50 licences to choose from there are inevitably differences and incompatibilities; the biggest differences usually being in the terms of re-distribution. In the worst case this can mean that software distributed under two different Free Software licences cannot be legally linked together and re-distributed.

If you are considering combining different pieces of free software it is important to check the relevant licences closely to make sure that you are not infringing the conditions of either. If you then re-distribute the modified software as a whole, however, you need to be careful in your choice of the licence or licences that you pass on.

It is important to note that for Open Source Software and Free Software the authors will still hold copyright. So, although any licensee has freedoms to modify and re-distribute the software, the original authors are still entitled to acknowledgement for their work. The licence will stipulate how this acknowledgement must be given, but it is usually sufficient for the source code to carry the relevant copyright notices.

NB: This article uses the blanket abbreviation OSS to refer to Free **and** Open Source Software unless the difference has some significance in the discussion.

## The Advantages and Disadvantages of Open Source Software

### What does it cost?

This is probably the first question that most people ask and ultimately sits at the heart of many business decisions. Unfortunately, the question is not a straightforward one to answer. The first thing to say is that the cost of acquiring the right to use a piece of software, the licence fee, is usually only a fraction of the TCO (Total Cost of Ownership). Hence asking the question “How much does the software cost to buy?” is only going to uncover a small part of the story; this is equally true of OSS and proprietary software. Consequently the fact that OSS can be acquired for low cost (often downloaded from the internet for no cost) is *not* always a decisive factor in its favour. When calculating the ‘cost’ of software all of the following will probably have to be taken into account:

- **Installation and configuration:** installation and basic configuration can be as simple as pressing a few buttons or can require hours of expert assistance. Simple installations are all well and good but if you need to install a large user base the costs soon add up
- **Customisation:** no software will ever do exactly what you need right ‘out of the box’, you may want to customise it if you have that option (even commodity price word processing software will usually need a few custom templates)
- **Downtime and Failures:** the consequences of software failures can include lost working time, lost data, cost penalties, lost sales and a damaged reputation
- **Security:** malicious attacks on computer systems are now commonplace; the less secure the system the higher the cost of protection and probability of a successful attack. The consequences will be the same as for downtime and failures above
- **Software support:** for reporting bugs, obtaining fixes, patches and updates. Any software that is anything other than completely trivial will have bugs. You must also consider that, as time goes by, changes in your IT environment are inevitable and your system may need to change along with it
- **User training:** whether the training is outsourced or in-house your cost calculation must include the working time of those being trained
- **System administration:** everything from adding new users to running backups. You should also consider the cost of auditing your systems to make sure that you have adequate commercial licenses; a non-trivial task

- **'Helpdesk' support:** for answering routine user queries and problems. This can be anything from a fully outsourced service to the well-known local 'expert'
- **Hardware:** the system may sit on shared hardware but part of the cost should be allocated to the system
- **Licences:** costing anything from nothing to of thousands of pounds or dollars per user or machine

The cost of licences, even when significant, can easily be outweighed by all of the other costs. Consequently when comparing OSS and proprietary software a broader view of costs and benefits is needed. Each of the points that follow below has a direct cost implication and should be considered in the cost-benefit equation. The discussion is by its nature in general terms, so specific examples may deviate from the general pattern.

### **Installation and Configuration**

**Proprietary:** The writers of proprietary software will try to make the basic installation and configuration of their products as painless as possible. Shrink-wrapped commodity level products, like office productivity software, will probably be easy to install. Of course this is little consolation if your organisation has a thousand desktop PCs. If the proprietary product is more of an enterprise level solution (e.g. a database) or niche specialist product (e.g. LIMS) then installation and configuration is likely to require the work of experts.

**OSS:** Installing OSS can be more challenging, but this really depends on what it is. Commercial distributors of OSS like Linux have made great strides in easing the installation process. After that you still have to make configuration changes, which is likely to require the services of an expert. Enterprise level or niche products are going to need expert installation and configuration.

**Comparison:** The most important cost factor in installation and configuration is probably going to be related to scale. If you have hundreds of desktop PCs you will be able to sort out installation issues on a few target PCs and the main cost then comes in rolling the system out to everyone (any lost working time of your users needs to be added in to this cost). An enterprise level system with many users is going to need significant work of experts regardless of whether it is OSS or proprietary.

## Customisation

Every business is different and wants to support different processes. Customisation is about getting software to support your processes rather than having the software dictate processes for you or only partially fulfilling your requirements.

**Proprietary:** Getting proprietary products to do what you want is usually possible in one of three broad ways:

- Configuration options – system settings and options to suit your preferences
- Built in customising features – typically using a built in programming or macro language
- Bespoke tailoring – software development undertaken by the vendor or their agent to alter the product at the source code level to suit your requirements

Configuration options may be all that you have available in commodity priced products and offer the least flexibility. Built-in programming languages are common in some office productivity software and enterprise level products. Bespoke tailoring is likely to be offered for niche market products where the customer base is small.

If you make your own customisations to a proprietary product, for instance through a built-in language, then this has to be considered in the overall costs. You will have the benefit of owning copyright on your customisations but they may be useless in any environment other than the product for which they were developed.

Bespoke tailoring can be a very expensive option since you may not have any choice in who does the work. The customisation may be charged in addition to the typically high licence fees charged for niche products. You may also find that you do not have any copyright or intellectual property rights over the customisations made on your behalf.

**OSS:** Broadly speaking you can get the same range of options as for proprietary software: configuration options, support for a programming or scripting language, or bespoke tailoring. The greatest difference will be in the capacity for bespoke tailoring. By definition this option is open to you for all OSS, not just niche market systems. Also, you will have the advantage of being able to shop around for the skills required to perform the customisation or do them yourself. You can retain copyright on your customisations and are under no obligation to share them with anyone else – although if you do the recipient may be entitled to the full source code and OSS rights to modify and distribute as they see fit (depending on the original OSS licence).

**Comparison:** If you cannot find a product that adequately satisfies your requirements 'out of the box' then you will have to engage in some customisation. If there are equivalent proprietary and OSS products to choose from, then going with customisation of OSS gives you freedoms that you will not get from proprietary software including copyright of your customisations, freedom to shop around for skilled resource, and unlimited opportunity to change software. If you engage a proprietary software vendor to tailor their product for you then they will probably offer support, but this may be at additional cost to supporting their standard offering.

### **Reliability (downtime and failures)**

Depending on what part of your business the software supports, the costs of system downtime can be very high. For example, if the software in question is a shopping website and it becomes unavailable, customers are likely to just go elsewhere.

There are two things that tend to affect the reliability of a piece of software:

- How well it was written in the first place
- How well it has been reviewed and tested to find problems

The first is a question of software quality; that is specification, design, coding and documentation. This is dependent on the skills of the developers and the way in which they are organised. The second is a question of the skill of the reviewers/testers and crucially the amount of time spent on that scrutiny. It therefore follows that later versions of a piece of software tend to be more reliable than the earlier versions. Hence, the reliability of the software is dependent on how it has been developed, not simply whether it is OSS or proprietary. There are examples of highly reliable software from OSS and proprietary sources. Equally you can get unreliable software from both.

The discussion on the relative merits of both approaches is therefore about the type of development that each tends to encourage.

**Proprietary:** By its very nature proprietary software is developed in a closed environment. This means that the size of the pool of skilled developers will have some limit, depending on the size of the organisation. Commercial pressures may dictate the release schedule for the product. Development of software in this type of environment can be of a high quality if budgets and release schedules are realistic, but it is common to find that 'Version 1' of the product has numerous problems resulting from the pressures under which it was produced.

Software will usually become more reliable over time as patches and new versions are released. For a proprietary piece of software, problems will often be found by customers and fixed by the vendor hopefully in time for the next release. The environment in which the fixes and enhancements are made is the same closed environment in which the software was developed in the first place.

**OSS:** Historically some OSS software products have been developed in a similar way to proprietary products, that is by a relatively fixed pool of developers, and so suffer the same drawbacks. The development model that proponents of open source software now prefer is quite different. The model is to develop in a completely open way by releasing software early, often and to include as many people as possible in development and testing. This is the big selling point for open source. The early releases may still have problems but the code is made available to a wide audience for testing and peer review. In this environment the people who find a problem in the software will commonly fix the problem themselves and offer it back to whoever is running the project. This approach also promotes critical appraisal of the design. For a more effusive description of the advantages of open source development you can read Eric. S. Raymond's paper 'The Cathedral and the Bazaar'\*.

**Comparison:** The reliability of software products is highly variable. Early versions or releases of software tend to be less reliable than more mature versions that follow later – this is equally true of OSS and commercial products. Reliability tends to come with the maturity of the software. The case for open source development is made on the grounds of the quality and reliability of the software that are produced. In some cases open source software does win over commercial software in the reliability stakes; the Linux/Apache web-server combination had a much better reliability record than Windows/IIS before the release of IIS 6.0. This is not to say, however, that most closed commercial products are not reliable – they generally are.

### **Security**

There are two schools of thought here: one says that security is best served by secrecy, the other by openness. At first sight it is plausible to suppose that an open source product could be easily investigated for vulnerabilities and so more easily subjected to attack. The counter argument is that open source software receives greater scrutiny and peer review, thereby ensuring that flaws are identified and fixed before they can be exploited. As evidence that secrecy is no guarantee of security just refer to the number and frequency of security patches to Microsoft products. So there is no valid reason to suppose that open source means low security; in fact there are arguments to the contrary.

\* See References at end

## **Security of Supply**

What do you do if the company that develops and supports your software goes out of business? If you rely on commercial products then the best outcome is that another software company buys the rights to the products to provide continuity of support. In the worst case you are left with a redundant and unsupportable system. If you use OSS then failure of a supplier is not such a problem; there will always be someone else prepared to provide support and you have the option of providing support yourself.

The risks of supplier failure have been causing concern for some time. It is possible that in future legal action may be available to force failing suppliers to make commercial product source available to customers. However, the source by itself, without a community of developers who are familiar with it, may be cold comfort to the customers of a failed supplier.

Having said all of this, the complete failure of a significant software vendors is rare these days – mergers and takeovers are much more common. But takeovers still present problems for customers as vendors rationalise their product lines. By contrast, OSS can never be ‘dropped’ by a supplier; there is always the option for someone to actively support and develop it.

## **Support and Vendor Lock-in**

If you buy proprietary software then you commit yourself to taking support, bug fixes and enhancements from one supplier. Depending on the supplier you may have very little control over the future direction of software product, product lifecycles and support costs. If the software is peripheral to your business then this may not matter, but if the software is key to your business – perhaps your IT infrastructure – then the decision to buy from one particular vendor can be a far-reaching one. With OSS you have a choice about who supports your software and ultimately you can choose the future direction that your software will take.

## Operating Systems

Although there are many commercial and open source operating systems in use, they can be broadly classified into two groups: Unix-like operating systems and Microsoft Windows. Microsoft Windows is the archetypal commercial software product. By contrast the complex family tree of UNIX-like operating systems arose out of code sharing, multiple development branches, open source and commercialisation at various times.

Some of today's Unix-like operating systems are distributed under OSS agreements and some of the current commercial products are likely to become open in the future (at the time of writing SUN Microsystems are openly talking about the possibility of releasing Solaris under the GNU GPL). But for now Linux is the leading OSS alternative to Microsoft Windows and the commercial UNIX-like operating systems.

The issues with using an operating system on the desktop as opposed to the server are quite different and worthy of separate consideration. For example GNU/Linux and commercial UNIX have a much larger share of the server market than of the desktop market because of the different demands placed on those two environments and the third party support that each attracts.

**GNU/Linux:** The GNU/Linux operating system is perhaps the best-known example of successful OSS. The GNU project (GNU's Not Unix) was instigated by the FSF to create a complete Free Software Unix-like operating system and developers all over the world contributed to its development. Linus Torvalds developed his Linux kernel under the FSF General Public Licence and so provided the means for the GNU operating system to function as a completely Free Software product. The whole operating system is now commonly referred to as Linux, although strictly speaking that is only the name of the kernel. Many other OSS components, including a large number from the GNU project, are required to form the complete operating system.

**Commercial Linux Distributions:** There are a number of commercial organisations offering their own distribution of GNU/Linux operating systems. The distributions are usually tailored in some way to add value and will often include additional open source products like office productivity software. The price you pay for these commercial distributions gives you some level of product support, although both the price and support model can vary widely. There are a number of commercial Linux distributions from the likes of SUSE, Mandrake, Debian and Red Hat. Red Hat is currently the market leader and has recently changed its approach to include two separate streams: the Red Hat commercial product range offering premium support services and the Red-Hat-Sponsored community-supported Fedora open source project. Although their Red Hat

Enterprise Linux product is distributed under the GNU GPL, it is effectively prohibited to re-distribute it 'as is' because it contains numerous trademarks that the user does not have the rights to distribute. This obliges the customer to pay for each and every product installation in return for receiving support. The advantage of paying for the Red Hat Enterprise Linux and similar products is that the vendor guarantees a certain product stability and compatibility that it couldn't provide previously. Of course, once you have a legitimate copy of a commercial Linux distribution there is nothing to prevent you from stripping out any trademarks or other impediments and then freely re-distributing it – but your rebuilt system will no longer have the support of the vendor.

**Hardware Support.** For every new piece of hardware developed by manufacturers (e.g. graphics cards) there is the potential requirement for a new piece of driver software. For PC hardware most manufacturers will develop and release Microsoft Windows drivers as a matter of course. By contrast, not all hardware manufacturers routinely develop new Linux drivers (although quite a few do). The result is that the Linux kernel usually lags behind Windows for hardware support, at least until someone in the open source community takes the time to write the new driver where the manufacturer has not done this. As long as you are not intending to use new leading edge hardware, this is not a problem, but you will need to check that the hardware you intend to use is supported before making a move to Linux.

**On the server.** Servers are mainly used for file storage, web servers, databases and high performance numerical computing. In most organisations servers are much fewer in number than desktop PCs and are generally the domain of highly skilled IT staff. This is part of the reason that GNU/Linux has made a bigger impact on the server market than the desktop – on the desktop the ubiquitous Windows is familiar and most accessible to end users, on the server the IT department has a real choice and can go with what best suits their needs.

A large part of the market share that Linux has on the server has been taken from commercial UNIX. The ownership costs for commercial UNIX server systems are generally much higher than for Linux or Windows because of the specialist hardware that is required. Moving from a commercial UNIX server to a Linux server is relatively easy for most IT departments because of the large similarities between the two. The big benefit of a move to Linux is that it runs on cheaper commodity PC hardware with an attendant reduction in hardware capital expenditure and support costs.

**On the desktop:** The world's desktop market is still the almost exclusive domain of Microsoft with its ubiquitous Windows operating systems. When developers write applications for the desktop they will naturally develop for the market leader and develop to the capabilities and strengths of that particular

operating system. The result is that there is much more software to choose from to run on the Windows platforms than any other. So, before considering whether GNU/Linux is a viable choice for your desktop you need to ensure that you will be able to find all of the applications that you need to run on it. For many organisations that means email, web browsing and office productivity software, all of which are readily available for GNU/Linux platform. For anything more specialised (e.g. CAD, GIS) then you may find that your choice of applications that will run on GNU/Linux is limited and less sophisticated than equivalent Windows or commercial UNIX compatible software.

**Administration:** Debate continues over whether Windows or Linux is easier to administer. Windows has nice GUI tools for administrators to use, whereas a Linux administrator needs to feel comfortable with the command line. That might sound like Windows would win for ease of use, but operating systems are complicated and you need expert administrators whatever you are using – having GUI tools is nice but you still need to know which one to use and what to do with it. Getting your administrators to switch from a commercial UNIX to Linux should be painless; asking administrators to switch from Windows to Linux or vice versa will require a substantial investment in retraining.

**Stability and Reliability:** It is widely reported that Windows system administrators spend a significant proportion of their time applying patches and service packs to their systems – more than an equivalent Linux administrator does. Whilst Windows servers used to have a reputation for unreliability (the famous blue screen crashes), the more recent releases have much improved reliability. Windows systems are, however, subject to many more successful virus attacks than Linux or other UNIX-like systems.

**Total Cost of Ownership:** TCO is always difficult to quantify accurately; it can vary greatly depending on the size, type and IT history of your organisation. TCO is currently a source of hot debate between the commercial and OSS camps. A quick search on the Internet will turn up plenty of TCO studies, some putting Linux with the lower TCO, whilst others will rate Windows as the cheaper of the two. TCO studies are just like other surveys; the results depend on what you choose to count, and it is important to consider who is running the survey and what vested interests they might have. Unfortunately there is no definitive answer – it all depends on what your business is and what you want to do; sometimes Linux will work out cheaper sometimes it will be Windows.

**Conclusion:** On the server you could consider using Linux as a cost-effective alternative to commercial UNIX-like operating systems; this is where Linux has gained a lot of market share. However, the benefits of using Linux as an alternative to Microsoft Windows on the server are less clear-cut. It will depend very much on how much Windows specific functionality and software you rely on for your current infrastructure and applications (e.g. Active directory, .NET, SQL Server). Choosing to run Linux servers in a few special cases alongside Windows servers could make sense (e.g. Linux for public web-servers and firewalls only) but a mixed infrastructure will probably make life more complicated rather than less.

On the desktop Windows still reigns supreme by virtue of the sheer momentum it has gathered over the last decade or more. If the applications that you need to use are available on Linux then moving your desktops to the OSS operating system may be an option for you. By carefully choosing the right version of Linux and application software you can escape the ongoing complications and costs of ensuring commercial license compliance for all of your desktop software – a non-trivial task if you have hundreds of PCs. However, if you already have a well-established Microsoft infrastructure then the task of changing over to Linux should not be underestimated.

## Web-servers

The web-server is another area where OSS is in widespread use by some very large businesses and organisations. The OSS Apache web-server hosts many business critical web-sites and competes head-to-head with Microsoft's IIS web-server.

**Market Share – the surveys:** The results of any survey depend largely on what questions you choose to ask and how you select your sample. The variability of surveys is no more evident than in assessments of the web-server market. The problem is that one server can run hundreds of web-sites with separate domain names, and conversely a single domain name can be served by many physical web-servers: so what do you choose to count? Most surveys seem to agree that the two main players in the web-server market are Microsoft IIS and open source Apache. The much-publicised Netcraft survey casts a wide net for its sample and gives Apache 67.1% market share and IIS 21.5%. The Port 80 Software survey takes a smaller sample of the websites of the top 1000 corporations and gets very different results giving IIS 54.1%, Apache 20% and Netscape Enterprise 15.1% (both sets of figures are quoted from their respective websites for May 2004). You can find out more about these surveys on the [port80software](#) and [netcraft](#) web sites.

What all surveys show is that the open source product Apache has significant market share and is used by some top corporations for business critical web infrastructure.

**Reliability:** Older versions of Microsoft's IIS had some real reliability problems that made Apache a much more attractive proposition. This undoubtedly contributed to the large inroads that Apache has made into the web-server market. With the release of IIS 6.0 and Windows Server 2003 Microsoft has completely re-designed the execution and security model of their product and addressed most of the reliability and performance problems in one go. Anecdotally the reliability of IIS 6.0 and Apache 2.0 are now on a par. Both systems have mechanisms to ensure that the failure of a single web request does not bring down the whole server.

**Configuration:** IIS offers numerous user-friendly GUI tools with which to configure your web-server. By contrast a basic Apache installation requires you to manually edit a text file to make configuration changes. There are, however, a few open source tools that provide a GUI interface to edit an Apache configuration. Whether the difference in configuration techniques matters is more a question of what your administrators are used to. Anyone who comes from a Microsoft Windows background is likely to prefer the familiar feel of the IIS configuration tools. Anyone from a Unix or similar background will probably be perfectly at home editing the Apache configuration file.

**Web-site Development:** If your website will consist of static HTML pages then as far as your content is concerned it won't matter very much which web server you choose. However, most business websites and intranets are likely to include some server-side dynamic content (that is to say web pages that are created or customised by the server at the time a user requests the page). IIS and Apache (sometimes with open source additions) both support a range of dynamic content technologies including: ASP, JSP, CGI, Perl, Python and PHP. In the context of open source web application development you may see the acronym LAMP, which refers to the combination of Linux, Apache, MySQL and one of Perl, Python or PHP. IIS also supports additional Microsoft technologies including .NET. The Microsoft specific technologies supported by IIS are very attractive to some organisations where a Microsoft based infrastructure is already in place. IIS allows developers to take advantage of Windows network authentication and a high granularity of access control using Active Directory due to its close integration into the operating system.

**Security:** Malicious attacks on IT systems are now commonplace. The Internet-facing web-server is, by definition, the most visible gateway into an organisation's IT infrastructure and so attracts constant attention from those intent on causing disruption and harm to systems. Security vulnerabilities in Microsoft's IIS web server and operating systems are identified and/or exploited frequently. Keeping up with the latest security patches is a regular task for system administrators. By contrast the Apache web server has had relatively few security problems in comparison to IIS. There are two schools of thought as to why this is the case; there are vocal supporters for both points of view:

- The first says that Microsoft products are the subject of more attacks because a) Microsoft products are more prevalent and b) Microsoft is specifically targeted because it is a large high-profile company
- The second says that Apache is inherently more secure because a) Apache is an open source product and so subject to wider scrutiny and b) Apache is self-contained whereas IIS is tightly integrated with Windows and so part of a more complex whole

Whichever explanation you prefer, it is widely accepted that IIS has to be patched for security vulnerabilities far more frequently than Apache.

**Support:** IIS comes as part of a Windows server operating system. As a commercially licensed product you can take advantage of the usual commercial support. If you want a commercial support package for Apache then there are some vendors offering this. Alternatively if you have suitably experienced staff you have the option of providing your own support using the resources and development community available via the Internet.

**Conclusion:** With the release of IIS 6 on Windows 2003 server then, anecdotally at least, there is now little to choose between IIS and Apache in terms of reliability. IIS does, however, still have the lesser reputation for security.

To a large extent your choice of web-server will probably go hand-in-hand with your choice of server operating system: IIS equals Windows, and Apache goes with Linux or UNIX. You may choose to run Apache on Windows (in some cases it comes bundled with other software like Oracle 9iAS) but, as IIS comes as part of the standard Windows server installation, you will want a compelling reason to use anything else.

Choosing a web-server isn't something that you do in isolation, it tends to be intimately connected to your choice of operating system, your technical policies and the skills that you have available. IIS and Apache are both capable of running business critical web sites and applications.

## Databases

Oracle, Microsoft SQL Server and IBM DB2 are the main players in the commercial database market. Two OSS alternatives are MySQL and PostgreSQL, both of which are widely used but account for a relatively small fraction of the market.

**Licensing:** MySQL and PostgreSQL are distributed under different licenses so it is important to assess how this would impact your use of either product. MySQL is developed by a commercial company called MySQL AB. MySQL is available under two different license arrangements: the Free Software GNU GPL and a commercial license. The GNU GPL is one of the more prescriptive FS/OSS licenses so you need to be fully aware of how using it may impact how you use and, especially, distribute any software based on or interfacing with MySQL. By contrast PostgreSQL is distributed under the open source Berkley BSD License, which is much more permissive.

**Speed and Capacity:** Database size and performance are usually inversely related. The big commercial database vendors put a lot of development effort into making their products perform well for the very largest databases, which can run into many terabytes. Both MySQL and PostgreSQL are known to perform respectably for databases sizes of up to 100 gigabytes, but for anything larger the commercial databases dominate the market. Of course 100 gigabytes is more than enough for many database applications.

**Features:** The commercial databases have led the way in database technology but the open source offerings are in the process of catching up, PostgreSQL more so than MySQL.

- Integrity - Both MySQL and PostgreSQL implement safe atomic database transactions to prevent changes being left in an indeterminate state. Both implement row-level locking and PostgreSQL also provides the alternative of Multi Version Concurrency Control
- Indexing - The right indexes can be a key factor in getting the best performance from a database. Whilst not being as flexible as, say Oracle, MySQL and PostgreSQL both offer primary key, multi column and full text indexing
- Development Features - Views, triggers, sequences and cursors are essential database features for anyone used to working with one of the commercial database products. PostgreSQL already offers all of these features and whilst not yet available in MySQL they are planned for a future release

**Administration:** The commercial database vendors put a lot of effort into providing nice graphical interface tools to allow you to administer their databases. If you can't face using a command line interface there are a number of open source administration tools available for MySQL and PostgreSQL.

**Backup:** Oracle, SQL Server and DB2 are mature products designed to support large and complex enterprise level systems. The commercial products come with some very sophisticated backup and replication tools, some provided by third party suppliers – data security and disaster recovery are big business. The open source databases are less well served when it comes to backups. Whilst MySQL and PostgreSQL do provide methods for making database dumps and hot backups, you will probably find less to choose from by way of third party enterprise level backup solutions.

**Application Development:** MySQL and PostgreSQL can both be accessed from applications developed using a variety of different development languages including C, C++ and Java. Both also have ODBC and JDBC capability enabling them to be used together with other software packages (e.g. Microsoft Access). Both databases are also frequently used in combination with the popular web application development languages Python, Perl and PHP. MySQL does not yet support stored procedures but PostgreSQL does. With PostgreSQL you have several language options for writing stored procedures including C, SQL, PL/pgSQL (similar to Oracle's PL/SQL), PL/Tcl, PL/Perl and PL/Python.

**Supported Platforms:** Both MySQL and PostgreSQL are available to run on a whole host of unix-type operating systems and Microsoft Windows.

**Support:** Commercial support can be obtained for both MySQL and PostgreSQL. The parent organisations MySQL AB and PostgreSQL Inc offer commercial support packages for their respective products and a variety of third party organisations also offer support. Of course with full access to the source code you also have the option of providing your own support should you wish.

**Conclusion:** Both MySQL and PostgreSQL are used for 'serious' database applications in commerce, industry and academia. For databases up to perhaps a few hundred GB they both offer a real alternative to commercial products like Oracle and SQL Server. Of the two open source databases PostgreSQL is more highly featured. An important consideration is also that PostgreSQL is distributed under the BSD license, which places hardly any restrictions on what you can do with the database. MySQL has a particularly good reputation for speed of access when used in a web application environment. Having said all of this, the commercial databases still lead the way in features, performance and scalability. In the end it comes down to what you want to do with your database. For some applications MySQL and PostgreSQL will provide excellent cost-effective solutions.

## Integrated Development Environments (IDEs)

IDEs exist to make software developers more productive. At its simplest an IDE may offer no more than a language sensitive text editor and a way of grouping source files into logical collections. At its most complex an IDE will offer many advanced capabilities including:

- Code organization into projects, libraries etc.
- Language and context sensitive editing
- Automatic code completion during editing
- On-the-fly compilation and syntax checking
- Automated code generation (e.g. for database wrappers)
- Integrated language reference material
- Code refactoring tools
- Multiple project views (e.g. by file, by class hierarchy)
- Drag and drop visual GUI builders
- Database schema browsing
- Source code repository integration
- Build tool integration
- Interactive source level debugging
- Integrated unit test tools

There has always been a wide choice of IDEs for the most popular languages like C, C++ and Java, including an increasing number of OSS alternatives. C# and .NET are also gaining attention from open source developers with new IDEs and extensions to existing OSS IDEs now appearing. There are far too many open source IDEs to list them all here so the discussion is restricted to a few of the more popular ones.

**To IDE or not to IDE?:** The reason for having an IDE is to make software development easier. An IDE achieves this by providing an array of tools that automate parts of the code development process and relieve the developer of having to do repetitive tasks and retain vast amounts of information about classes and libraries. The things that IDEs do for developers also form the principal objections that some have to their use. The argument goes that IDEs provide too much insulation from what is going on so that the developer gains less understanding of the fundamentals. Some IDEs create extra layers of code abstraction that hide the detail of the implementation and make it more difficult to understand what is going on. The arguments in favour of and against IDEs both have merit – ultimately it comes down to personal preference (or corporate policy). Bearing this in mind, a free open source IDE with perhaps limited functionality may be far more attractive than a more fully featured commercial product.

**The Commercial Offerings:** Before describing the open source IDEs we have to begin by characterizing the proprietary offerings. Commercial IDEs are numerous and include some very mature and highly featured products. For Microsoft technologies Visual Studio .NET offers an enormous number of productivity enhancing tools. People who move away from Visual Studio to other IDEs usually miss some of the rich features that it offers. The big drawback with Visual Studio .NET is the cost – it is one of the more expensive development tools. For the Java world there are a number of popular IDEs including Sun Java Studio (Sun), JDeveloper (Oracle), IDEA (Intellij) and JBuilder (Borland). Whilst open source IDEs are gaining in popularity the proprietary products still win out on features in some cases.

**Eclipse:** Eclipse was originally developed by IBM, who subsequently made the product open source. Eclipse is now maintained and developed by the Eclipse Foundation, enjoying the active support of many organizations including the likes of IBM, HP and Intel. Eclipse is perhaps best known as a Java IDE but is actually much more. Eclipse is an open and extensible platform for building software tools; the Java IDE is just one such tool. Eclipse tools are written as ‘plug-ins’, of which there are hundreds available. The list of official plug-in providers include Oracle, Borland, Red Hat and many others. Eclipse is written in Java and so is available to run on many different operating systems.

Eclipse is a very popular tool; the Java IDE is mature, robust and functionally rich. The Java IDE includes some powerful language-specific refactoring tools allowing you to make global modifications in a far more powerful way than a simple search and replace can. Eclipse also supports the SWT graphical toolkit, which is an open source alternative to the standard AWT and Swing toolkits.

Until now Eclipse has lacked a 'drag-and-drop' style visual interface editor, but as recently as May 2004 the first version of VE (Visual Editor) was released for Eclipse 3.0, which is currently in development and targeted for release in June 2004. The addition of VE and similar graphical editor tools should attract many more users to Eclipse's growing community. In common with many other IDEs Eclipse integrates with other development tools like CVS, ant and junit (source code control, build and test tools respectively). If you find that the base Eclipse IDE lacks something that you need then there is a good chance that a plug-in has been written to address that need. In addition to Java there are Eclipse tools for other languages including C/C++, Cobol, PHP, Python and C# to name a few.

**netBeans:** netBeans was acquired by Sun in October 1999 and after some additional development was made open source in June 2000. Sun is now the sponsor of the netBeans.org open source project. Like Eclipse, netBeans is best known as a Java IDE, but also like Eclipse, is also a platform for developing other IDEs and desktop applications. Other language plug-ins are available for netBeans including C/C++ and FORTRAN. netBeans also has the graphical interface design capability that Eclipse has been lacking until recently. Like Eclipse, netBeans is written in Java and will run on many different platforms and there are a large number of plug-ins available to extend its functionality. The features list offered by netBeans is similar to Eclipse and which of the two you prefer may come down to personal choice.

**Eclipse or netBeans:** Eclipse and netBeans are IDEs as well as being extensible platforms for developing other types of desktop application. As Java IDEs they are pitched head-to-head with the weight of IBM behind Eclipse and Sun behind netBeans. Both of these products are open source, freely available to use, extend and modify, but even so there are large commercial interests at stake. Some of the things that appear to be at stake are influence over and control of Java and IDE standards. Of the two, Eclipse seems to have the larger following. As a developer you have simply to try them both and choose the one that you prefer.

**#Develop (SharpDevelop):** #Develop is a GPL Free Software IDE for developing C# and VB .NET applications. Superficially at least, it looks just like Microsoft's Visual Studio .NET, but closer inspection reveals that #Develop does only a small sub-set of what Visual Studio does. In particular it lacks support for developing ASP .NET WebForms and the database productivity tools in the Microsoft product. Having said this, #Develop, now at its version 1 beta release, promises to be a good basic IDE for developing .NET Windows Forms applications, console applications and Windows services. The project also aims to support cross platform C# development, which is now becoming possible thanks to other open source projects. If the project continues to get the support of the open source community #Develop could be a real alternative to Visual Studio for some types of application development.

**Conclusion:** Your choice of IDE will probably be settled on grounds of cost, support and features. Some development tools are expensive, but since they will always provide big productivity improvements these costs are easily justified. Support for the open source IDEs is almost exclusively do-it-yourself, but, since IDEs are used by the most computer literate of all users, this need not be a problem with the resources provided by the large open source community. Which features you want or prefer is a matter of choice. High levels of automation in an IDE can provide great productivity to begin with but, if the cost is extra layers of abstraction and obscure auto-generated code, you can lose the benefits later if you need to refactor or investigate more obscure problems. Of course, the great thing about these open source IDEs is that you can download them and try them out for nothing.

## References

1. 'The Cathedral and the Bazaar' is an essay by Eric S. Raymond, presented at the Linux Kongress on May 27, 1997. It was published as part of a book of the same name in 1999.  
For more information, please see  
[www.catb.org/~esr/writings/cathedral-bazaar](http://www.catb.org/~esr/writings/cathedral-bazaar)
2. The Free Software Foundation  
[www.fsf.org](http://www.fsf.org)
3. Open Source Initiative  
<http://www.opensource.org>

### Web server market surveys on the web -

1. Netcraft - <http://news.netcraft.com>
2. Security space - [www.securityspace.com/s\\_survey/data/index.html](http://www.securityspace.com/s_survey/data/index.html)
3. Port80 Software - [www.port80software.com/surveys/top1000webservers](http://www.port80software.com/surveys/top1000webservers)

## **Tessella Support Services plc** **Creating Software for Science and Engineering**

Tessella's services range from feasibility studies, through system design, development, implementation and ongoing support. Our expertise includes:

Data Analysis Software  
Data Capture  
Simulation Software  
Advanced Graphics  
Systems Support  
Database Applications

### **Other Technical Supplements available include:**

- |   |  |
|---|--|
| <input type="checkbox"/> Active Server Pages            | <input type="checkbox"/> LIMS                          |
| <input type="checkbox"/> Archiving of Electronic Info   | <input type="checkbox"/> Linux                         |
| <input type="checkbox"/> Automated GUI Testing          | <input type="checkbox"/> Microsoft .NET                |
| <input type="checkbox"/> Bayesian Statistics            | <input type="checkbox"/> Object Oriented Programming   |
| <input type="checkbox"/> Beowulf Clusters               | <input type="checkbox"/> Open Source and Free Software |
| <input type="checkbox"/> C++                            | <input type="checkbox"/> Pocket PC                     |
| <input type="checkbox"/> COM                            | <input type="checkbox"/> Portable GUI Development      |
| <input type="checkbox"/> Computational Fluid Dynamics   | <input type="checkbox"/> Real Time Systems             |
| <input type="checkbox"/> Computer Image Processing      | <input type="checkbox"/> Regression Testing            |
| <input type="checkbox"/> Decision Support Systems       | <input type="checkbox"/> Security and the Internet     |
| <input type="checkbox"/> e-GIF                          | <input type="checkbox"/> Simulation                    |
| <input type="checkbox"/> Electronic Data Capture        | <input type="checkbox"/> Soft Computing                |
| <input type="checkbox"/> Electronic Lab Notebooks       | <input type="checkbox"/> Software Development Cycle    |
| <input type="checkbox"/> Evolutionary Computing         | <input type="checkbox"/> Software Documentation        |
| <input type="checkbox"/> Excel                          | <input type="checkbox"/> Software Portability          |
| <input type="checkbox"/> Extending the Life of Software | <input type="checkbox"/> Software Re-engineering       |
| <input type="checkbox"/> FDA 21 CFR Part 11             | <input type="checkbox"/> Software Specification        |
| <input type="checkbox"/> Formulation                    | <input type="checkbox"/> SQL                           |
| <input type="checkbox"/> FORTRAN 90                     | <input type="checkbox"/> UNIX Inter-Process Comms      |
| <input type="checkbox"/> Grid Computing                 | <input type="checkbox"/> UNIX Systems Performance      |
| <input type="checkbox"/> High Throughput Screening      | <input type="checkbox"/> Web Services                  |
| <input type="checkbox"/> Instrumentation                | <input type="checkbox"/> Windows 2000 Services         |
| <input type="checkbox"/> Integrated Lab Systems         | <input type="checkbox"/> XML                           |
| <input type="checkbox"/> J2EE                           | <input type="checkbox"/> X Windows                     |
| <input type="checkbox"/> Java                           |  |



INVESTOR IN PEOPLE

### **Tessella Support Services plc**

3 Vineyard Chambers, Abingdon, Oxon, OX14 3PX, England

Tel: (+44) (0) 1235 555511 Fax: (+44) (0) 1235 553301

E-mail: [info@tessella.com](mailto:info@tessella.com) Web Address: [www.tessella.com](http://www.tessella.com)