Advice report

EUROPEAN COMMISSION

Enterprise Directorate General

IDABC/GPOSS

Encouraging Good Practice in the use of Open Source Software in Public Administrations

PATENTS AND OPEN SOURCE SOFTWARE:

WHAT PUBLIC AUTHORITIES NEED TO KNOW

Prepared by:

MERIT

UNISYS

Short report for IDA/Open Source Observatory
Senior consultants at MERIT: Rishab Ghosh, Reinier Bakels
Senior consultant at Unisys: Patrice-Emmanuel Schmitz

1. Executive summary

While patents on computer programs are not allowed in European law “as such”, in practice, patents have been granted that cover functionality in many common software applications. Such patents may appear trivial or invalid, and may sometimes eventually be overturned after legal costs. However, they pose a risk to the development of software.

The risk is primarily to developers and vendors of software. Users are rarely subject to legal action for patent infringement. Developers and vendors must almost always be notified first, prior to legal action seeking damages, thus giving them an opportunity to replace or remove functionality that is possibly patented. The risk to users is therefore more likely to be related to higher costs for specific (patented) functionality, or reduced competition in the marketplace, rather than the direct costs of being subjected to patent lawsuits.

Patent law does not discriminate between free/open source software and proprietary software. The theoretical risk is therefore equal. The risk as demonstrated in present practice would appear to be higher for proprietary software, as no free / open source software has yet been subject to legal action for patent infringement, although open source software development, like proprietary software development, has been halted in specific areas of technology with known software patents, such as MP3 audio or the LZW data compression present in the GIF graphics file format. The fact that source code is available does not make free software significantly more at risk to patent infringement action than proprietary software, as patents cover functionality, not source code. Indeed, the availability of source code is an enormous documentation resource that can be used to invalidate software patents by proving the prior existence of patented ideas, though this is an expensive process.

Vendors rarely indemnify users against legal costs and awarded damages relating to patent infringement. The ability and likelihood of vendors providing such legal protection is related to the size of the vendor rather than the type of software licence. Small vendors (open source or proprietary) are far more vulnerable to legal action relating to patents. Users however, even customers of small vendors, are highly unlikely to be sued for patent infringement.

In conclusion, the risk to public administrations using free / open source software from *legal action related to patent infringement*, while not zero, is very low. The EU Directive (EU Council version) is unlikely to increase such risk insofar as it confirms the existing practise of the European Patent Office of granting patents on software.

2. Legal protection of software: trade secrets, copyright and patents

The prime means of legal protection for computer software is copyright. All software is protected automatically by copyright as soon as it is created, until 70 years after the death of the author. Copyright essentially protects against direct copying of the software itself without the permission of the author, but not much else.

Due to the limitations in copyright protection, some software developers felt the need to protect underlying techniques using patents. Patents are granted for inventions, typically after a costly and complicated procedure, for a time period of up to twenty years. Patents are only granted for inventions that are novel, and that are "not obvious to a person skilled in the art". The invention is represented by the description that is submitted for the patent application, not by actual software code. Thus, a single software patent can protect multiple implementations.

As an alternative for patent protection, software techniques can be kept secret. Such "trade secrets" usually are also protected by law, typically by penal law. Secrecy may be contractually agreed with technology transfer agreements. Not all techniques can be protected with secrecy. Techniques that can be "reverse engineered" (if only by using the product or reading the manual) cannot be kept secret. Also, once a secret is broken, it is lost. However, trade secrecy can be a useful means of protection. It is free, and it lasts forever, unless it is broken. That is one of the reasons smaller companies in particular do not apply for patents but rely on secrecy whenever possible.

The demand for software patents is not new. In the early computer days, there were no independent software products. Systems software packages (operating systems) were typically bundled with the machine, and application software was written by the users themselves. However, from 1970 on, software was increasingly traded as a product by itself. That increased the demand for legal protection. Another development that increased the needs for software patent protection was the rise of programmed microprocessors in various branches of traditional technology, varying from digital signal processing to control engineering.

In the early 1970’s some software patents were actually granted, but generally patent offices and courts had reservations about granting software patents. A practical reason was the lack of documented knowledge on software techniques. In order to assess the novelty of a patent, comprehensive knowledge of the state of the art is required. But there were also more fundamental reservations. Is software really an industrial product, or is it just a logical construct?

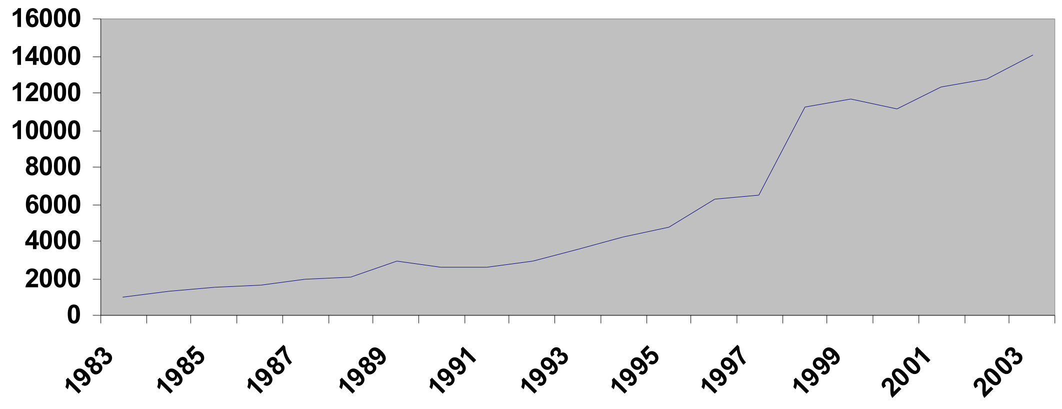
In order to avoid lengthy discussions about the patentability of software, patents on computer programs “as such” were excluded in the 1973 European Patent Convention (EPC). Unfortunately, this provision did not bring the intended clarity, and led to numerous court cases.

What is meant by the exclusion of computer programs “as such”? The history of the negotiations on the EPC gives no clue about this. Critics of software patents argue that this exclusion ought to be construed as a categorical exclusion of software patents, and some even argue that all software patents granted by the European Patent Office (EPO) until now are illegal, and thus cannot be enforced in court.

As the purpose of patent law is supposed to be the furtherance of technology, only technical inventions are supposed to be patentable, even though this is not explicitly required in the statutes. But what is “technical” in conjunction with software? In the 1970’s, the German courts equated “technology” to applied natural sciences, and found hardly any software invention qualified for patent. But the European Patent Office from the very beginning was more flexible and adopted a “dynamic” definition of technology. This caused the German courts eventually to follow, from the late 1980’s. In the Netherlands, the Patent Council in the mid-1980’s already accepted a wide patentability of software inventions. The perceived need for international harmonisation was given as one of the reasons. Typically, courts tend to follow each other, usually increasing patent protection. In theory, everybody is aware that patents have disadvantages too and must judiciously be granted – but in practice that hardly ever leads to a decrease in patent protection (except in the time of “abolition” movements in the late 19th century that caused some countries such as the Netherlands and Switzerland to suspend patent protection entirely for some time).

The EPO interprets the exclusion of *software as such* as a *conditional* exclusion. Only *technical* software inventions are supposed to be patentable. The rules to decide whether software is technical, in a legal sense are rather vague, and subject to change. But among lawyers, there is little doubt that some software can be patented indeed. The subject decisions have been challenged in court on numerous occasions[1], and it was always confirmed that at least some software is patentable. As neither the European Patent Convention nor the related national statutes has been changed in this respect, realistically, with the current state of the law some software patents are allowed. But what software patents is not entirely clear.

In the U.S., there were similar rules to draw the line between software inventions that were supposed to represent mere algorithms and those that represented truly "technical" inventions. But in the landmark 1998 *State Street Bank* decision a U.S. Court decided that inventions that lead to something "concrete, useful and tangible" qualify for a patent, if the other statutorial requirements are met such as novelty and nonobviousness. The court stated that the nonobviousness test was more important than the test on "statutory subject matter". This court decision caused a flood of software and business method patents (see diagram below).



Software patent grants (class 7xx) in the U.S. over the years.

2.1. "Computer Implemented Inventions"

Lately, the term "Computer Implemented Inventions" is proposed in conjunction with software patents. Proponents of this term argue that patents always only relate to inventions - that may or may not be implemented in software. In particular inventions that relate to conventional technical systems using programmed microprocessors are supposed to represent "Computer Implemented Inventions" rather than software inventions. Still, even the patentability of such inventions is questioned. In a famous early case in this area, the German

[1] EPO Board of Appeal for instance decisions 15 July 1986, T208/84, *Official Journal of the EPO* 1987,14 (*VICOM*); 21 May 1987, T26/86, *OJ* 1988,19 (*Koch & Sterzel*); 6 October 1988, T6/83, *OJ* 1990,5; 20 April 1994, T59/93; 4 February 1999, T935/97 and 1 July 1998, T1173/97, *OJ* 1999, 609 (*IBM I and II*); German Federal Supreme Court (BGH) for instance decisions 13 December 1999, *GRUR* 2000, p. 498-502 (*Logic Verification*); 11 May 2000, *GRUR Int* 2000, p. 930-933 (*Speech Analysis Device*).

Supreme Court decided that an anti-skid system using a programmed microprocessor to control the operation of brakes was indeed a patentable technical invention[2].

There are cases where the distinction between such an invention and a "software invention" is less obvious, or perhaps nonexistent. For instance, the same image enhancement technique may be implemented in a specialised harware (computer-controlled) device such as a television set and also in a software application running on a general-purpose computer (to improve the quality of your pictures). Therefore, a meaningful distinction between "Computer Implemented Inventions" and "pure" software inventions seems to be impossible. Patents are granted for inventions, not for implementations.

2.2. Business methods

Probably the best known questionable business method patent is the *Amazon.com* "One Click" patent. In the US many business method patents were granted, in particular after the 1998 *State Street Bank* decision that effectively abolished the "business method exception".

The European Patent Convention does not allow patents on "schemes, rules and methods for doing business", but – again – only "as such". Similarly to software, in Europe business methods are assumed to be patentable only if they are technical. A technical software implementation could make a business method technical. However, the interpretation of technical for business methods, as for software, appears to be broad. While the Amazon "One Click" patent does not exist in Europe, the Amazon "Gift ordering" patent covers the technology of allowing consumers to ordering goods on-line and have them delivered to a third party (as a gift); similarly, the "shopping cart" patent granted by the EPO to Sun Microsystems covers a virtual shopping cart on a website.

2.3. Confusion and the SCO lawsuits

The lawsuits launched in 2004 by SCO against Linux-related firms, in particular IBM, has led to widespread confusion over the legal risks of using the Linux kernel. In fact, SCO does not claim to hold patents over Linux. It initially claimed to own some Linux copyrights, but it appears not to be claiming that any more. The lawsuit mainly relates to SCO's claims that IBM has violated contracts with SCO. SCO's initial lawsuit against an end-user of Linux (DaimlerChrysler) was dismissed by the court.

[2] German Federal Supreme Court (BGH) 13 May 1980, *GRUR* 1980, p. 849-852 (*Antiblockiersystem*)

3. Software patents in Europe: Law and practice

Originally, patents were only issued by *national* governments, and applied only to the territory of one nation. With the increase of international trade, patent protection had to be obtained again and again for each individual country. That is the reason talks started in the early 1960's for the establishment of a European patent system.

The adoption of the European Patent Convention (EPC) in 1973 and the subsequent institution of the European Patent Office (EPO) in 1977 are only steps in the realisation of a truly European patent system. The current implementation, still in force today, is a compromise. The European Patent Office only handles the *grant* of European patents: the technical examination and the related legal formalities (including appeal and opposition procedures). However, after grant a European patent is treated as a bundle of national patents in (selected) EPO member states. Infringement and nullity cases are handled by the national courts. While national statutes have been adapted to the European Patent Convention, courts may decide differently, in particular in borderline cases. So potentially a "European" patent may be upheld in one European country and nullified in another one. Since the EPC is not EU legislation and the EPO is not an EU institution, this situation cannot be easily resolved in the current framework.

Therefore, the European Commission would like to set the rules for a uniform interpretation by means of a European Directive. While this approach still leaves patent adjudication after grant to the national courts, the interpretation of a European directive is subject to the supervision of the European Court of Justice, whose decisions related to EU directives are binding on all EU member. Furthermore, while the EPO and EPC would still remain outside the EU institutional framework, the preparation of any directive actively involves EU institutions including the European Parliament.

After a long series of consultations, in February 2002 the European Commission eventually issued its proposal for a "Directive on the Patentablity of Computer-Implemented Inventions", allegedly with the purpose of clarifying the boundary between patentable and non-patentable software inventions, by a codification of the current EPO practice.

So far, this directive has not yet been adopted. Software patent adversaries are opposed against a codification of current "liberal" practice, and they believe the directive will obscure rather than clarify the boundaries of patentable subject matter, effectively allowing patents on *all* software inventions. Proponents of the directive argue that it is not about software patents, but only about "computer implemented inventions" such as telecommunication equipment and television sets containing programmed microprocessor chips.

The provisions of the directive may not necessarily exclude pure software patents, if only because the same patented invention can often be implemented both in a microprogrammed device and in plain software that is delivered on e.g. a CDROM. In sum, the boundaries between patentable and nonpatentable software inventions are unclear, and they will probably remain so if and when the directive is adopted, in whatever wordings.

By requiring a "technical contribution", the directive also aims at preventing "business method" patents. No definition of the word "technical" is given however, so effectively the "technical contribution" criterion is openended, and the interpretation by courts and the EPO has seen a gradual expansion of what is "technical" over the past several years.

During the first reading of the directive draft in September 2003, the European Parliament proposed many amendments to the proposed directive, including a strict definition of "technology". The European Council of Ministers found most of these amendments unacceptable. At the time of writing, it is unclear where the Directive Proposal will end. It seems unlikely that wording in the Parliament amendments that prevent software patents altogether would be adopted in the near future, if at all: while software patent adversaries argue that such wording reiterates the letter and intent of the law – i.e. the EPC – it would certainly be a significant change from current legal practise, i.e. the interpretation of the EPC by the legal boards of the EPO and some national courts.

4. How many patents do software packages infringe?

4.1. Potential infringements by software in general

Doing something that is only permitted to the patent owner without his consent is called infringement. In the current European patent system, the actual acts that constitute infringement are not listed in the EPC but in national laws, and infringement is dealt with by national courts. For this purpose, a distinction is made between inventions that are described as *products* and inventions that are described as *processes* in the patent application. Without the consent of the patent owner, nobody else is allowed to make, offer, dispose of, use or import a patented *product*, or to keep it whether for disposal or otherwise. In case of a *process*, nobody except the patent owner is permitted without the consent of the patent owner to use the process or to offer it for use to anybody else.

There are some exceptions to these rules. Private non-commercial use and experimental use of a patented invention are permitted without the consent of the owner. However, these exceptions must be construed restrictively. Use in a not-for-profit organisation such as a government agency for instance is not considered private use. And only experimentation *on* the patented invention is allowed, experimentation *with* the patented invention is not permitted. The purpose of the experimentation exception is to allow other inventors to build on a patented invention. It is not meant to be a general exception for research institutions.

A distinction is made between *substantive* infringement and *contributory* infringement. Delivering something that may help other people to infringe upon a patent may be contributory infringement.

This distinction is important for software patents. Traditionally, software patents are either drafted as product patents (the product being a programmed computer or other device) or as process patents (the program or method itself). In both cases, delivering the program by itself may constitute only contributory infringement. As this may be harder to prove than direct infringement, software patent owners prefer direct product protection for the software by itself. Both the EPO and the U.S.P.T.O. have been responsive to this need and allow "program product claims".

If a "program as such" cannot be patented, can a "program by itself"? It is one of the controversial issues of the proposed European Software Patent Directive. Following EPO Board of Appeal case law, supporters of the Directive argue that there is no fundamental difference between a software invention drafted in such a way and the same invention drafted traditionally as a programmed machine. Adversaries of software patents however argue, that "program product claims" tend to specify – and patent – entire problems rather than specific solutions. Such "wish list" patents could have an excessive scope of protection because they cover any implementation that solves the described problem. According to current law, a patent application does not need to include a concrete ("reference") implementation of the software solution. An abstract description is sufficient, covering potentially a wide range of implementations.

4.2. Quality and triviality

Strong concerns have been raised against trivial software patents. Many of the software patents that were issued are believed to be trivial, or not even novel. Patents on inventions that were not novel at the time of grant are invalid and can be nullified in court.

Novelty checking of software inventions is difficult because there is less documentation about such inventions than e.g. about inventions in the area of chemistry, and most software inventions are not properly documented in professional literature at all. This problem is believed to be even more serious with business methods. Business people do business, they rarely document their methods, especially when these methods are not implemented in software. Patent examiners rely on databases of "prior art" – previous examples similar to the invention being claimed for a patent – but it takes a long time to create and search such databases.

Some patents appear trivial by their lack of novelty, but real triviality is something else. Traditionally, the "non-obviousness" threshold for patents is low, in any area, and software patents are no exception in this respect. Patents on inventions that appear to be "obvious" actually can be legally valid, based on current legal views. A patent office or a court can not withdraw or nullify a patent that was granted in agreement with the rules that applied at the time of grant, even if those rules are deemed inadequate in hindsight. The fact that some software technique, such as an online shopping cart, may seem obvious *today* has no bearing on the validity of a patent granted a year ago.

4.3. Litigation

Unlike copyright law, patent law does not allow an "independent discovery defence". If an invention is patented, the patent owner can even sue someone who made the same invention independently and applied it unknowingly. This risk of "hitting" a software patent inadvertently is supposed to be considerable, due to both the low threshold of the patent system and the overly wide scope of protection of some "wish list" patents. Small software companies cannot afford legal advisors that check every piece of software for potential patent infringement. Nor can most end-users, including public authorities.

Typically, patent infringement is handled as a civil law case, even though several European statutes contain penal provisions as well. A typical counter-claim is to challenge the validity of patent, due to lack of novelty at the time of application or other legal shortcomings. Conscious infringement may have to be proved in order to claim damages (as opposed to just an injunction). In the Netherlands for instance a prior notice is required to claim damages, allowing the alleged infringer to cease his activities.

Litigation is expensive. An U.S. organisation, the Open Source Risk Management (OSRM) has evaluated the average cost of patent litigation to about $3m per lawsuit. While it may be less expensive before a European court than in the U.S., in Europe potentially legal action is required in each EPO state. A unique feature of the European patent system, much envied by the Americans is the *opposition procedure*, allowing anybody (in practice, mainly competitors) to challenge the validity of patents during the grant procedure.

In a concrete infringement case, it may be advisable to avoid a long and expensive court procedure and to arrange a settlement, even in questionable cases. It may be helpful to build a “defensive” patent portfolio in order to deter competitors from infringement charges, or otherwise to be prepared for a settlement by a cross-licensing agreement. This is affordable only for large companies. For end-users, settling a case for relatively small amounts of money may be cheaper than a legal defence, but is still expensive, and has the disadvantage that even the holder of a bad patent can make money out of it through out-of-court settlements with several end-users, whereas a single court case could invalidate the patent and save all end-users a lot of money.

4.4. Potential infringements by open source, including Linux

Is Free/Libre/Open Source Software more affected by software patents than other software? Indeed, it is not inherently more likely to infringe software patents than proprietary software. As described above, a lot of software faces potential patent infringements for the simple reason that software is extremely complex, a single small software package can involve concepts and ideas that could be covered by several hundred patents (in contrast to a drug, which is typically covered by under 10 patents). Furthermore, software appears especially vulnerable to “trivial patents” with more such patents apparently being issued and many potential infringements of trivial patents. Trivial “inventions” in software tend to be independently “invented” over and over again in different software packages, but patent law equates use of independently invented ideas with infringement.

This problem affects all software equally. There has been no evidence showing that free software is any more vulnerable to patent infringement than proprietary software. There have been several cases, however, of free software development being affected by software patents. Such cases usually lead to a software development project being terminated in its early stages. There has been no case so far of a publicly released “user-ready” open source software application being the subject of a patent infringement lawsuit.

There is an argument that free software/open source may face a slightly higher risk of attracting attention from patent holders, since source code is available and can therefore be easily inspected for possible patent infringement. Even if proprietary software is infringing a patent, it can be somewhat harder, according to this argument, for a patent holder to detect.

However, software patents relate to functionality, not source code, and most “high-value” patents that would be worth the patent holder pursuing legal options cover functionality that would be perceived by the user, rather than hidden somewhere within the software application. The unavailability of proprietary software source code is no protection against patent infringement threats. Indeed, there have been numerous patent infringement actions against proprietary software, and none (so far) against open source software.

The availability of source code does allow independent organizations to make a careful assessment of potential patent infringement. A well known study for Open Source Risk Management by Dan Ravicher of the Public Patent Foundation examined the Linux kernel version 2.6 and concluded that it could potentially infringe as many as 283 US patents. However, 98 of these are owned by Linux allies who would presumably not sue, including IBM (60 patents), Hewlett-Packard (20) and Intel (11). Speaking to MERIT, Mr Ravicher was not able to estimate the proportion of these 283 US patents that have equivalent patents in Europe, as these were not studied. However, there are between 60000 and 140 000 US software patents, depending on the definition of “software patent” and between 20 000 and 30 000 software patents granted by the European Patent Office using similar definitions[3]. This may suggest that there are European equivalents already granted for about 30% of US software patents.

While documenting the potential infringement risks for the Linux kernel, the OSRM study only demonstrates that it may be easier to calculate infringement risks for open source software; the risks for proprietary software are not inherently different but may be harder to calculate by independent researchers.

[3] Note that European patent law tends to avoid the term “software patent”, choosing instead to refer to “Computer Implemented Inventions”.

The OSRM study also shows that open source software is not without defences against threats of legal action for patent infringement. On the one hand, open source developers themselves are unlikely to be large enough to participate in “cross-licensing” arrangements with other patent holders (who compare the size of their respective patent portfolios and agree not to sue each other). It has been argued that open source developers may therefore face a higher risk of lawsuit: in case of infringement, large proprietary software companies, owning many patents, tend to have already “exchanged” rights to one another other through cross-licenses rather than going to court. Free software / open source developers cannot do this, as their “organisations” do not own large portfolio of patents that are worth cross-licensing for other patent holders. It should be noted that small proprietary developers face the same problem, and this risk of open source developers being sued has not so far materialised.

On the other hand, open source is backed by large companies who are accustomed to using patents as strategic weapons. IBM, for instance, responded to SCO’s lawsuit – see section 2.3 – with a countersuit accusing SCO of infringing IBM’s software patents. Moreover, several open source licences such as the GPL which covers the Linux kernel include defensive clauses that offer some protection against patent infringement threats.

5. Who gets sued for software patent infringement?

As described above, open source software is not especially at risk for software patent infringement. Indeed, patent infringement lawsuits so far have dealt with the use or development of proprietary software. One reason for this could be that open source is either developed by individual developers or very small companies, who may not worth suing, or by very large companies such as IBM, who would probably be too expensive to sue. However, it is not only software developers who can be held liable for patent infringement – users are liable too, and as open source software gets more widely used in large companies or large public administrations, it is increasingly likely that such large users of open source will face action for patent infringement.

Patent infringement suits can be launched against developers, vendors and support providers, or users. Most lawsuits so far have been against developers and resellers – after all, with proprietary software the developers and vendors tend to be the same. With open source, since individual open source software developers are unlikely to be sued, and vendors may be small companies, large users or large vendors may be likely to face more lawsuits. However, this is unlikely for a number of reasons. While it is true that in the case of open source, users, especially large enterprises and large public sector authorities are a more appealing target for patent infringement lawsuits than individual open source developers or small supporting companies, in most jurisdictions the patent holder is obliged to provide notification of infringement and an opportunity for remedy prior to suing for damages. Thus, patent holders may in practise have to notify the original developers or at least the suppliers to a user organisation of infringement, providing an opportunity to implement remedies before suing a user organisation for damages. In the worst case, such remedies could be removing functionality in the software, or working around the patents to achieve similar functionality. The costs to users in such cases will be the requirement to update their existing software with new software unaffected by the patent claim, rather than to face the patent claim themselves.

Thus, while a very small threat of infringement of user organisations may exist, it is more a threat for developers and small suppliers / open source companies who would be forced to either rewrite software to avoid patents after they receive notice of infringement, or avoid developing software altogether for certain functionalities where the potential infringement threats are well known[4]. The result is a reduction in business for small companies and open source developers, and user organisations are far more likely to suffer indirectly, due to a lack of competition in the marketplace, than any direct threat in terms of legal action against them for patent infringement.

6. Who covers these risks?

Most vendors who provide support for open source software, including HP, IBM and Novell, do *not* indemnify customers against legal risks related to patent infringement[5]. Red Hat provides an "Intellectual Property Warranty", promising to replace purchased software or functionality in the event of any legal issue relating to "intellectual property", which presumably includes patents. Sun provides extensive indemnity to customers against patent infringement threats in its own (but not third-party) open source software. The majority of open source software comes from other, smaller, vendors who rarely provide indemnity. Similarly, Microsoft is almost unique among proprietary software vendors in providing most of its customers indemnification against legal costs and damages related to patent infringement, since November 2004. Prior to this date, in common with most proprietary software vendors, Microsoft limited its liability to the licensing cost of the software paid by the customer.

[4] For instance, open source software has difficulty providing support for MP3 audio, TrueType fonts, and various video and compression systems, which are covered by well known software patents.

[5] HP and Novell provide legal indemnity for copyright infringement claims.

This brief survey shows that vendor-provided indemnification for legal costs relating to patent infringement is uncommon. This is perhaps a reflection of the low risk to customers, especially in relation to vendors, who will almost invariably face notification of infringement and any possible legal action well before users. The Microsoft indemnity programme is a good example of this – it is exceedingly unlikely that a patent holder will sue an end user rather than the world's richest software company, and indeed patent infringement actions relating to Microsoft's software have so far been targeted at Microsoft, not its users.

Some insurance firms do provide patent litigation insurance, and OSRM specialises in insurance related to infringement risks of open source software. It is unclear whether such insurance provides any significant value to user organisations given the relatively low risks they face.

In case of an infringement claim, the defence is often to challenge the validity of the patent, e.g. by challenging the novelty or at least the non-obviousness based on the state of the art at the time the patent was applied for. Free / open source developers as well as others in the software community have established a number of documentation centres that time stamp the state of the art with regard to all conceivable areas of software technology. Users and vendors of open source software may be advised to co-operate with such documentation efforts, but the same goes for all users and vendors of all software.

7. What steps can Public Administrations take to contain risk?

As can be seen from the discussion above, the risk to public administrations using free / open source software from *legal action related to patent infringement*, while not zero, is very low. The EU Directive (EU Council version) is unlikely to increase such risk insofar as it confirms the existing practise of the European Patent Office of granting patents on software. While not in the form of direct legal threats, there may well be some negative effects of patents related to software for public administration users of free software. As described in section 4.4 and section 5, these relate mainly to the effect on the market of reduced competition, and possible termination of free software support for specific features due to patent threats against the *developers*.

So what should Public Administration *users* do? In brief, when considering free software, just as with proprietary software, it is important for users to conduct a careful selection and evaluation of alternatives, costs and benefits of each specific application or solution being considered. Such an evaluation should include an assessment of the origin and popularity, both of which indicate the extent of interest there is in the software among a community of supporting organisations. Given the extent of support for popular free software / open source solutions, there would seem at this point no reason in general for public administrations to stop using or considering the use of such software. But the discussion around software patents serves to reiterate the importance of being aware of the legal aspects of software. Users should not only consider the software itself , but also the community of support that is built around it and the arrangements made with any organisations providing, installing or implementing the software, whether it is proprietary or open source.