Innovation Claims of FOSS — for Software Innovation and against Non-inventive Software Patents

Siu-wai Leung and Dave Robertson
Centre for Intelligent Systems and their Applications
School of Informatics, The University of Edinburgh, U.K.

4 August, 2004

Abstract

This paper proposes the adoption of innovation claims for some Free and Open Source Software (FOSS) to promote software innovation and refute non-inventive software patents. We also show how our research on automated synthesis of conceptual models could help the implementation of innovation claims in FOSS repositories.

GNU/Linux together with other FOSS constitute the current Linux desktop and server distributions. While FOSS is getting support from more users and developers, they are facing tougher challenges, e.g., greater software functionality demands (in competition with well-packaged commercial software) and growing numbers of software patents. FOSS communities must collaborate to improve their product integration to gain more user support and find evidence to dispute the inventiveness of particular software patents. Our research in automated synthesis of conceptual models provides some automated facilities for conceptual integration of different FOSS. The conceptual models of a target software package (new or competitive products) serve as references for establishing/experimenting new FOSS integration projects. If the resulting automatically synthesised conceptual models are similar to target patents these can serve as formal evidence against those patents.

1 Much Ado About Innovation

Free and Open Source Software (FOSS) encourages software (and its knowledge) sharing, re-use, and integration. It provides a pool of freely accessible knowledge, in addition to a resource of software tools, in many fields of applications. Even though FOSS developers must live with the restrictions of copyrights and licences, people may think similarly without copying and expresses the shared ideas differently, so copyrights do not unduly impede the FOSS development. However, like most people, FOSS developers find it difficult to examine every

bit of their ideas too see whether these are the same as (or similar to) existing software patents. Software patent databases are difficult to search. The original international patent classification did not include software because software was not considered patentable. Software patents could be categorised under any of their many application fields [Bessen, 2004], so it is difficult even for patent search experts to locate relevant software patents. In a random sample of 400 patents granted in 2002, it was estimated that 15% of the patents contains software patent claims [Bessen and Hunt, 2004]. FOSS communities want to get rid of software patents. Not only the Asian governments but also the US government prefer FOSS to proprietary software because of cost and security and e-voting considerations [Evans and Reddy, 2003]. However, it is likely that software patents will be around for many years with the argument being that patents, including software patents, play significant and sustainable roles in techno-economic innovation. As software patents aim to encourage more software invention, it is sensible for FOSS communities to demonstrate alternative ways of encouraging invention/innovation. This article will discuss some innovation issues addressed by software patents in the hope that FOSS communities will find specific ways to cope with these threats. We propose the use of innovation claims in FOSS repositories to describe the innovative features of particular pieces of FOSS. We believe that the FOSS innovation claims could refute non-inventive software patents and promote the software innovation in general. Ultimately, we hope that more software developers would contribute their knowledge in the form of FOSS rather than software patents.

2 Innovation Incentives

To encourage non-altruistic people to publish their new and useful knowledge, society establishes incentive schemes, such as patents and funding, to provide the necessary incentives that new and continued innovation will be conducted and disclosed. The incentives aim to motivate people to innovate and/or cover the cost of innovation processes. As an incentive, patents are granted to those who have done significant innovations, so-called inventions. There is always a dispute about how to give the right incentives (not necessarily patents) in the most economical way [Cohen, 1999, Maurer and Scotchmer, 20004].

Is our software industry innovative enough? We can list some points (non-exhaustive) for discussion. In an innovative industry (a kind of optimality), we should see

- appropriate competition or successful models for learning;
- open and efficient communication;
- fostered improvements;
- little duplication of creative efforts,
- re-use, combination, and integration;
- appropriate rewards/incentives, etc.

Some symptoms and signs of deviating from the optimality comprises:

- complaints of innovation impediments (inefficiency);
- too few similar products (monopoly);
- too many similar products (over-competition);
- complaints of inadequate incentives, etc.

In our opinion, FOSS has no lack of innovation but it seems that a lot of FOSSs share similar features. There may be a duplication of creative efforts.

FOSS developers are often self-motivated or motivated by the incentives other than direct financial returns [Lerner and Tirole, 2002], e.g., from selling software. We would like to see FOSS contribute new knowledge to the public domain. If it happens and FOSS demands a lower price for the society to procure new knowledge, the society will rely less on other incentive schemes such as patents.

There are alternative incentive schemes to motivate innovators to develop and publish their new knowledge, including

- research grants and awards given by the governments, charity organisations, and private companies;
- publications as achievement recognition;
- idealistic beliefs to which people can commit; and
- secondary incentives (e.g., income from providing services, consultancy).

3 Incentive Theories of Patents

Among many rationales of intellectual property rights such as moral (natural) rights, economic incentives, and market protection of entrepreneurial talent [Andersen, 2002], incentive theories of patents are the most explanatory and popular. In incentive theories, patents serve as the incentives to motivate people to disclose (publish) the details of their useful inventions, which would remain secrets or get lost otherwise. The laws and practices of patents are different from country to country but the main idea is to grant protection of the inventions which are

- statutory subject matter,
- novel,
- non-obvious,
- useful, and
- well-specified.

Non-statutory subject matter comprises laws of nature (scientific discoveries), mathematical formulae, logics, etc. In some countries,

things for human survival such as food, diagnostic and therapeutical methods of diseases, and things for immoral/unethical use are not patentable. Software was seldom thought to be patentable until recently. The statutory subject matter should be novel (there exists no similar thing), non-obvious (it is not simply a combination of things or it has an unanticipated effect or it is not easy to achieve), and useful (it has practical use or potential commercial value), and well-specified (so that ordinarily skilled people in the field can implement it accordingly) in order to receive a patent. The novelty and non-obviousness are relative to the state-of-the-art. Relevant pieces of knowledge (often in form of publications) are called prior art. A patentable invention should be novel and non-obvious over the prior art. A patent is not valid forever. It is only valid within a period of time, i.e., a patent term, usually 20 years if well-maintained. During a patent term, the patentee enjoys exclusive rights to protect a specific invention from the manufacture, sales, and use by others. After the patent term, the invention goes to the public domain. Besides serving as rewards, patents prevent others from doing redundant things and drive other inventors away to develop different things.

To help people understand and stimulate socio-economic research, incentive theories such as prospect theory [Kitch, 1977] and rent dissipation theory [Grady and Alexander, 1992] have been proposed. In the prospect theory of patents, patents are regarded as mineralisation prospects, which prevent others to conduct mining work in the same registered area although other businesses are allowed in that area. In this way, people will try to find other sites and register them to be mineralisation prospects. Eventually, more mining sites will be discovered. It also helps maintain the returns attractive to keep people interested in mining. As regarded as mineralisation prospects, patents direct the allocation of R&D resources away from patented areas and towards more pioneer innovations.

Rent dissipation theory of patents makes an analogy between doing R&D for obtaining patents and seeking rents from the government [Grady and Alexander, 1992]. In the rent seeking model to explain rent dissipation in economics, people compete with one another to secure their share of rents (profits). As the total rents are fixed and competition employs resources, some resources would be wasted in stiff competition and thus increase the total social cost to obtain the fixed rents. To avoid such rent dissipation in patents, patent seekers would try to do major improvements over a patented area or invest resources in new areas in order to secure some patents [Dam, 1993]. In the long term patent races can filter out inferior innovators [Judd *et al.*, 2003].

While prospect theory provides general rather than concrete guidelines to determine patentability, rent dissipation theory suggests not to grant the patents to the inventions which are close to prior art. In a mathematical economic models of rent dissipation theory of patents guided R&D [Dorsazelski, 2002], it was shown that the changes of rent dissipation in R&D races correlates with the degree of patent protection. Based on this model, it was argued that the patent system can be

used to modulate the R&D effort. However, further economic studies, particularly experimental economics, must be done to see whether the rent dissipation model is applicable to the software industry R&D with a significant contribution of FOSS.

4 Invasion of Software Patents

Software should not be granted patents because its novelty is about mathematical algorithms (abstract ideas) rather than tangible things. According to [Cohen and Lemley, 2001], in 1970s software patents in the U.S.A. were in form of mechanical devices and processes that happened to include computer programs. Disputes of software patents [Perelman, 1996, Sachs, 1996, Burk, 2000, Cohen and Lemley, 2001] have never stopped since then. Economics research of patents, which were moderately active a century ago, was recently revived [Lunney, 2001, Kortum and Lerner, 2002, Lerner, 2002, Bessen, 2004]. Software patents pretended to be something other than pure software (e.g., embedded in hardware devices or stored in disks) until 1998, when a software patent won a court appeal. A simple conceptual shift such as "software can turn any generic computer into specific machinery" overcame the rejections of software patents in the U.S.A. courts although the software patents need to have the following restrictions:

- restriction by the utility of the invention;
- directed to a utilitarian object;
- method written in English language, not in programming languages;
- must not "pre-empt" the field (cannot be the only method of accomplishing the result or cannot be a scientific discovery – law of nature); and
- cannot cover unknown uses of the algorithms.

Until 1998, software patents in the U.S.A. cannot be claimed as pure algorithms but programmed apparatus, data structures, or programmed storage devices [Cohen and Lemley, 2001]. Therefore, software patents may appear in various categories of the patent databases. No wonder software patent search is a big problem.

What does a pure software method claim look like? A pure software method claim can be in the following structure:

> A method embedded in a set of computer-translatable instructions comprising:
> <algorithm/procedure>
> so that <input> is <transformed> in conjunction with a computer into <output>.

where [algorithm] represents the steps of the algorithm, [transform] represents a specific keyword to describe the action of the algorithm, [input] specifies the information about the input, and [output] describes the expected output results of the algorithm.

Alternatively, the claim can be written in the following pattern:

> A method embedded in a set of computer-translatable instructions comprising:
> <algorithm/procedure>
> in order to <transform> <input> into <output>.

In common with other kinds of patents, software patent claims are usually broader than what has actually been implemented. Such broad patent claims hinder further improvement by others, rather like the reasonable claims that are supposed to allow just enough breathing room for the inventors to invest in development without fear of others to steal the idea and to do something quite similar.

Besides the problems of patent search and broad claims, there are problems due to the non-inventiveness of some software patents. For example,

- the royalty fees of software patents are disproportionate to actual R&D costs and
- 20 years of patent term is too long for trivial software patents (the same technology can be available as FOSS in a short time, even if it is not earlier than the patented technology).

It seems that the society is paying too high a price for unworthy software patents. Either shortening of the term for software patents or raising the inventiveness standards of software patents would be preferable to rectify the shortcomings of the current practice of software patents.

5 Innovation Claims of FOSS

While FOSS communities need to protest and urge for abandoning or rectifying the software patents, we think innovation claims for FOSS would also help to reduce non-inventive patents and provide breathing room for FOSS developers to develop their innovative software or improve the software technology already claimed by FOSS. Our proposal is to help software developers register the innovation claims for the innovative features of their FOSS in public FOSS repositories. The innovation claims and a certificate of registry would serve as an incentive (for those who really need some recognition) to motivate the development and publication of FOSS . Such innovation claims do not restrict the software industry but serve as prior art to deter software patent applications. For ease of search and checking against the software patent applications, the innovation claims can be drafted in the same language and structure of patent claims. For simple automated reasoning about the innovation claims and patent claims, restricted languages or formal representations can be developed. Such improvement in automation will facilitate the prior art search and FOSS reuse/integration. In principle, the public (particularly the software developers) would know the innovation claims and that knowledge will

encourage the public to work for different and/or better innovations for the betterment of society.

5.1 Innovation Incentives and Forces

FOSS with innovation claims can be recognised as something like "Free, Open Source, and Innovative Software" (FOSIS). A certificate of registry could be issued. As FOSIS would represent the state-of-the-art of FOSS, it should be an honour to FOSS developers. Unlike patent claims, innovation claims do not claim legal rights to solicit royalties but only the knowledge contributed to the community. Depending upon the repositories, registration should require no fees and formal examination. Search reports can be run in the FOSS repositories to demonstrate the innovative (or state-of-the-art) nature of the claims. While FOSS developers got incentives for innovation, patent seekers and commercial software developers, who usually got much better resources, have to produce better innvoations to compete with FOSS. This would reinforce the incentive schemes to prevent the duplication of creative efforts and encourage inventive resources to be used more efficiently.

5.2 Reasoning about Claim Scopes

The registration of FOSIS does not need rigorous examination which is required in patent examination. The only test is searching for similar innovation claims which have already been registered. The claimed is assumed to innovative if no similar software can be found. Even if there are similar software with overlapping innovation claims, the search result of the overlapping claims will be shown and the result would not restrict FOSS development. Restricted languages and simple automated reasoning are desirable to facilitate the determination of overlapping claims.

5.3 Blocking Patent Applications

Hopefully, FOSIS would block non-inventive software patent applications rather than be blocked by software patents. Patent seekers would not waste their time and money to file a software patent application if they can easily find one or more pieces of work having similar claims in public FOSS repositories. The task of Patent Examiners to find sufficient cases of prior art to reject applications can become easier. Probably, public FOSS repositories would evolve into FOSS software knowledge bases in complement with the computer science bibliography databases, including the database of software technologies maintained by Software Patent Institute (SPI). As long as similar prior art can be easily found, it is less likely non-inventive software patent applications will be even filed and more likely such applications will be rejected by the Patent Office.

5.4 Software Re-use and Integration

Ordinary FOSS users would not have time to try out all software available in the repositories. If possible, they just want the most useful ones and prefer all the useful pieces integrated in one package for ease of installation, configuration, and use. To gain more user support, integrating reusable FOSS components would be an important task. As innovation claims are written in a restricted language like a subsumption with types and subtypes to describe the salient software features, we can search the innovation claims to find suitable FOSS components for the development or re-design of integrated software. This would increase the possibility to generate project ideas for the re-use and integration of FOSS components in an automated manner.

5.5 Specification and Documentation

FOSS often does not provide quality specification and documentation. The innovation registration of FOSS may be an incentive to encourage project participants (not necessarily the core developers) to write basic specifications of their software. This would enable all people to study, use, and improve the FOSS. Innovation claims require certain clarity of specifications of a piece of software and would alleviate the specification and documentation problems of FOSS.

6 Inspirations and Challenges

Overturning software patents needs better strategies to demonstrate that software innovation can flourish without patents as social incentives. These strategies and evidence should be based on a lot of studies across many disciplines. For informatics research, the adoption of innovation claims in FOSS repositories would stimulate the development of some related technologies. Examples of those related to our research [Robertson and Agusti, 1999] are:

- automated reasoning of the innovation claims and patent claims;
- artificial artisans as patentability reasoners;
- easier anticipation of software re-use and integration; and
- automated software synthesis or transformation of software specifications.

Although there are other interesting R&D issues for the software industry with a significant contribution of FOSS, we concentrate on these in the discussion below.

6.1 Synthesis of Project Ideas

The innovation claims of FOSS (and software patent claims) are very high-level software specifications of the claimed software method. The innovation claims also provide chances to find compatible software

methods/components and generate some ideas/concepts of integrative software. For re-use and integration, the ideas/concepts will serve as collaborative project ideas among numerous groups of FOSS developers.

6.2 Artificial Artisans

A kind of imaginary persons, so-called skilled artisans or the persons who are ordinarily skilled in the art, are very often used to determine whether the patent application can be easily anticipated in the field. All patent specifications must be written with clarity to enable skilled artisans to implement the claimed technology. The reasoning result of artificial artisans might serve as the evidence what the existing technology can perform and thus help determine the obviousness of software. We might advocate a "doctrine of artificial artisans" that any innovation (or patent) claim should be refuted if the corresponding specification can be synthesised by artificial artisans.

6.3 Representation and Reasoning of Claims

The structure of innovation claims is similar to patent claims which we have described earlier. A claimed scope of innovative software is specified by a set of innovation claims (C). One of these claims should be independent of other claims in the set and is called the independent or main claim. The other claims are dependent upon or subsumed by the main claim or other dependent claims in order to provide further definitions. The scope of prior art can be defined by a set of similar claims (P). Suppose each claim with an identifier c or p for defining a transformation procedure consists of the following information:

- a set of input defintions $\{i_1, ..., i_n\}$;
- output definition o;
- a transformation relation r; and
- a set of detailed transformation steps $\{t_1, ..., t_k\}$.

Hence a context of innovation claims may be defined as (C, R, I, O, T), where C, R, I, O, and T are sets, and $c \in C$, $\{i_1, ..., i_n\} \subset I$, $o \in O$, $r \in R$, and $\{t_1, ..., t_k\} \subset T$, where T is a set of possible transformation. Each transformation relation r represent a relation among $t_1, ..., t_k$ to transform the input $\{i_1, ..., i_n\}$ into o. A context of prior art claims may be similarly defined as (P, R, I, O, T), except there is P instead of C and $p \in P$. Each claim can be decomposed into dependent claims to further define the individual transformation steps.

This simple definition is not the only way to represent innovation claims and software patent claims. An alternative representation is available for generic product and process patent claims [Dave, 2003] but it is less informative to represent software claims.

To reason about the claim scope, we can consider the transformation steps $\{t_1, ..., t_k\}$ as the elements contributing to the innovation. To evaluate the inventiveness of a claimed software method as defined

by an innovation claim (or patent claim), we can see if there is a known claim (e.g., in the prior art) sharing similar transformation steps and their relation. The novelty of the claim c_1 consisting of r_i, i, o, and $\{t_1, ..., t_k\}$) will be refuted if there exists a prior art claim p_i consisting of r_i, i, o, and $\{t_1, ..., t_k\}$) or a claim p_j consisting of r_j, i, o, and $\{t_1, ..., t_k\}$) where r_j is equivalent to r_1. We can also evaluate the equivalence (so-called "doctrine of equivalence" in patent law) of each element of the claim to any known claim. The non-obviousness of the claim c_1 consisting of r_i, i, o, and $\{t_1, ..., t_k\}$) will be refuted if there exist more than one known claims having all the equivalent elements of the claim (even when no single known claim contains all the equivalent elements), i.e., $c_1 = \bigcup_{H \subseteq P} H$. The refutation of non-obviousness may also need other information in the description of prior art, such as any suggestion or motivation to perform the innovation and any mention of a reasonable expectation of success [Kieff, 2003].

We just provide a naive representation and simple reasoning about the innovation claims in this article. To ensure the success of logical reasoning of the claims, we also need a software ontology. It seems to us that FOSS repositories are in the best position to refine their current software classification into a community-wide FOSS ontology, which would enable better prior art search and the automated reasoning of software features for refuting the patentability of similar software. The use of Semantic Web languages to describe the ontology and facilitate the automated reasoning processes seems appropriate.

6.4 Experimental Research

There were controversies in economic literature before 1870s on whether patents are necessary. Since then, in more than a century of time, only occasional papers were published on the economical aspects of patents. There has been a surge of economics papers on software patents in recent years, particularly those focus on mathematical modelling and experiments of rent dissipation, patent races, and R&D investments [Baye and Hoppe, 2002, Kortum and Lerner, 2002, Kähkönen, 2003, Judd *et al.*, 2003, Bessen, 2004]. There have been some considerations of FOSS economics [Lerner and Tirole, 2002]. It is interesting to study how the software industry is influenced by FOSS. We need empirical research to explore more possibilities and study the socio-economic roles or effects of FOSS and software patents in technological innovation. Informatics techniques such as web surveys and automated experimentation might help expedite the experimental economics investigations. Once a testing hypothesis is given, an experiment synthesised from conventional experiment protocols could be executed, e.g., to do automated patent search for a certain kind of patents and then perform a statistical analysis.

7 Concluding Remarks

FOSS can be seen as a rich and freely accessible resource of software and its knowledge. The proposal of innovation claims of FOSS is not the only solution to promote innovation of FOSS and prevent interference by software patents but we believe it is worth consideration. Innovation claims will reinforce their contribution in software innovation. The research into the technologies such as claims reasoning, automated synthesis of software project ideas, and artificial artisans would be stimulated in order to argue against non-inventive software patents and to facilitate the re-use and integration of FOSS components. The establishment of free, open source, and innovative software (FOSIS) repositories would become a strong defense to deter the invasion of software patents. It is expected that more innovation efforts in FOSS would be made in order to promote ordinary FOSS to FOSIS by registering innovation claims. If these can happen, software patent seekers would need more R&D efforts in order to secure some patents because of many easily accessible prior art and innovation claims of FOSS. For greater benefits to the the public, FOSS should go to the public domain, i.e., abandon the restrictions of copyrights or licenses, as soon as possible.

References

[Andersen, 2002] Andersen, B. (2002) If 'intellectual property rights' is the answer – what is the question? Draft paper, Department of Management, Birkbeck College, University of London.

[Bagley, 2001] Bagley, M.A. (2001) Internet business model patents: obvious by analogy. Michigan Telecommunication and Technology Law Review 7:253-288.

[Baye and Hoppe, 2002] The strategic equivalence of rent-seeking, innovation, and patent-race games. Games and Economic Behaviour.

[Bessen, 2004] Bessen, J. (2004) Patents and diffusion of technical information.

[Bessen and Hunt, 2004] Bessen, J. and Hunt, R.M. (2004) An empirical look at software patents. Manuscript in preparation.

[Burk, 2000] Burk, D.L. (2000) Patenting speech. Texas Law Review 99:1-55.

[Cohen, 1999] Cohen, S.A. (1999) To innovate or not to innovate, that is the question: the functions, failures, and foibles of the reward function theory of patent law in relation to computer software platforms. Michigan Telecommunication and Technology Law Review 5:1-33.

[Cohen and Lemley, 2001] Cohen, J.E. and Lemley M.A. (2001) Patent scope and innovation in the software industry. California Law Review 89(1):1-57.

[Dam, 1993] Dam, K.W. (1993) The economic underpinnings of patent law. Chicago Working Papers in Law and Economics (2nd series) No. 19.

[Dave, 2003] Dave, R.J. (2003) A mathematical approach to claim elements and the doctrine of equivalents. Harvard Journal of Law and Technology 16(2):508-559.

[Dorsazelski, 2002] Dorsazelski, U. (2002) Rent dissipation in R&D races.

[Evans and Reddy, 2003] Evans, D.S. and Reddy, B.J. (2003) Government preferences for promoting open-source software: a solution in search of a problem. Michigan Telecommunication and Technology Law Review 9:313-394.

[Grady and Alexander, 1992] Patent law and rent dissipation. Virginia Law Review 78:305-382.

[Jaffe and Lerner, 2004] Jaffe, A.B. and Lerner, J. (2004) Innovation and Its Discontents: How Our Broken Patent System is Endangering Innovation and Progress, and What to Do About It. Princeton University Press. ISBN 069111725X.

[Judd *et al.*, 2003] Judd, K.L., Schmedders K., and Yeltekin, S. (2003) Optimal rules for patent races.

[Kähkönen, 2003] Kähkönen, A.T. (2003) Patent race experiment revisited: an experimental exploration of R&D competition. ??

[Kieff, 2003] Kieff, F.S. (2003) The case for registering patents and the law and economics of present patent-obtaining rules. Harvard Law School Discussion Paper No. 415. (also Washington University School of Law Faculty Working Paper No. 03-4-3).

[Kitch, 1977] Kitch, E.W. (1977) The nature and function of the patent system. Journal of Law and Economics 20:245-280.

[Kortum and Lerner, 2002] Kortum, S. and Lerner, J. (2002) Assessing the contribution of venture capital to innovation. Rand Journal of Economics 31:674-692.

[Lerner, 2002] Lerner, J. (2002) 150 years of patent protection. American Economics Review Papers and Proceedings 92:221-225.

[Lerner and Tirole, 2002] Lerner, J. and Tirole, J. (2002) The simple economics of open source. Journal of Industrial Economics 52:197-234.

[Lerner and Tirole, 2004] Lerner, J. and Tirole, J. (2004) The scope of open source licensing. Manuscript in preparation.

[Lunney, 2001] Lunney, G.S. (2001) E-obviousness. Michigan Telecommunication and Technology Law Review 7:363-422.

[Maurer and Scotchmer, 20004] Maurer, S.M. and Scotchmer, S. (2004) Procuring knowledge. Advances in the Study of Entrepreneurship, Innovation and Economic Growth 15:1-31.

[Perelman, 1996] Perelman, M. (1996) Software patents and the information economy. Michigan Telecommunication and Technology Law Review 2:93-102.

[Robertson and Agusti, 1999] Robertson, D. and Agusti, J. (1999) Software Blueprints – Lightweight uses of logic in conceptual modelling. Addison-Wesley and ACM Press. ISBN 0-201-39819-2.

[Sachs, 1996] Sachs, R.R. (1996) Comments in response to the Patent and Trademark Office's proposed examination guidelines for computer-implemented inventions. Michigan Telecommunication and Technology Law Review 2:103-125.

[Schumm, 1996] Schumm, B. (1996) Escaping the world of "I know it when I see it": a new test for software patentability. Michigan Telecommunication and Technology Review 2:1-55.