SOFTWARE PATENTS: AN ECONOMIC PERSPECTIVE

First Supplementary Submission
By The League for Programming Freedom
To The Patent and Trademark Office
On Patent Protection for Software-Related Inventions

(by Gordon Irlam and Paul Rubin)

Version 1.0

This document is the League for Programming Freedom's response to issues raised by the San Jose hearings of January 26-27, 1994. The League's submission on software patents is contained in a separate document.

Speakers at the San Jose hearings presented many conflicting arguments for and against software patents. To evaluate these arguments, we must organize them within a systematic framework and answer the questions that they raise. Since the patent system is an economic system, economics is the best framework. We therefore use an economic perspective to evaluate the arguments we heard. We first describe our approach to the overall question of software patents, and then move on to further questions.

What is the Main Question That Needs to be Answered?

The goal of the patent system is to promote progress. Whether software should be patentable is therefore a question of whether software patents promote progress. The economic interpretation of this question is whether granting software benefits the economy by making the software industry more efficient. More explicitly, this question asks:

> Does the transfer of economic resources which software patents represent constitute a transfer whereby the resources are going to be employed more productively?

Most broad economic transfers produce both winners and losers, so we need to look beyond the fact that a transfer occurs, and instead look at the larger picture. Sometimes this can be difficult: few economists believe agricultural subsidies are a good idea, yet the agricultural industry is nearly universally in favor of them. A government hearing on agricultural subsidies would be likely to receive almost overwhelming testimony as to their beneficial effect.

We can't simply classify public testimony on the effect of software patents and measure its volume to determine its validity. We need to carefully evaluate and investigate the issues.

We need to seek advice not only from those directly affected, but also those best equipped to answer the question: economists, especially economists familiar with the software industry. At stake is the future efficiency of a $50 billion a year industry that prospered and progressed quickly without patents. There is a basic conflict between patent monopolies and a free-enterprise economy based on competition, so free-enterprise principles demand that people who want to impose bureaucratic restrictions on a productive industry must show a clear need for them.

Is it Possible to Legislatively Define Software?

This issue was raised frequently at the San Jose hearings. We were very surprised that such an argument would be made to justify risking the future efficiency of a $50 billion a year industry.

Since this argument is apparently one of the key arguments in favor of the continued granting of software patents, we would like to subject it to intense scrutiny.

It is true that many things in this world form part of a continuum. Nonetheless we are able to legislatively differentiate between them. The Post Office is able to distinguish between a letter and a letter packet. The FDA is able to distinguish between a cheese spread and a cheese flavored spread. There is no way to draw a perfect line between "drunk" and "sober", but the law does draw a line, and it works.

On a larger scale, the IRS classifies capital goods into many different categories to determine depreciation rates, while the Customs Service is able to classify things to apply duties. Considerable financial incentives exist to try to circumvent these classification systems, yet they work. There is little problem with them being circumvented, or with their complexities imposing great financial burdens. The legal system effectively handles disputes over occasional borderline cases.

A legislative definition of software need not embody absolute truth. It need only work effectively and efficiently. Searching for absolute truth makes no more sense than determining the exact definition the IRS should use for wood pulping machinery.

Here's the definition we propose:

> Software is composed of ideal infallible mathematical components whose outputs are unaffected by the components they feed into.

We are confident the PTO and courts could readily distinguish between software and hardware using this definition. The PTO is already skilled at administering a classification system that deals with far more subtle distinctions.

Appendix A clearly shows that it is possible to classify patents based on their software characteristics. We examined 2,000 patents issued during a one week period and analyzed the details of every software related patent. We were able to readily identify such patents using any of a number of different techniques.

Do New Companies Need Software Patents to Attract Startup Capital?

At the San Jose hearings Tom Cronan of Taligent forcefully suggested startup companies require software patents to attract venture capital. He described Taligent as a recent startup that had succeeded in attracting a large amount of venture capital, and for whom software patents were considered as vital. He failed to mention Taligent is an IBM - Apple joint venture, staffed by transferring surplus personnel from these two companies. Taligent is quite different from most other startups.

The numerous "two kids and a garage" stories demonstrate that successful software ventures require relatively little capital. It isn't necessary to attract large amounts of capital to produce software. Or at least it was not necessary — defending against patent threats may increase the expense.

All the software companies spawned by the microcomputer revolution gathered sufficient starting capital without any software patents. Microsoft, Borland, Novell, Adobe, Symantec, Oracle, and WordPerfect are just a few examples.

Other speakers besides Tom Cronan also claimed startups need software patents to attract capital nowadays. It doesn't however follow that the existence of software patents helps startups. An explanation can be found via simple economic reasoning: if software patents exist, then having software patents cannot harm a startup; they can only help or do nothing. The outcome depends on whether your competitors also have them. The payoff matrix is:

		YOUR COMPETITORS HAVE S/W PATENTS	
		YES	NO
YOU HAVE S/W PATENTS	YES	you lose somewhat	you win somewhat
	NO	you lose big	neutral

Economists will recognize this matrix as a textbook example of the Prisoner's Dilemma from game theory. In either case, having software patents are to your advantage, if such patents are available. A rational investor therefore must insist on them when they are available, even though this strategy is bad for the software industry and the public as a whole. Government policy should aim to alter Prisoner's Dilemma situations so that the parties involved will prefer choices that give a high total benefit.

Will an Improved Prior Art Database Solve the Problem?

At the San Jose hearings, several companies called for the Patent Office to improve its prior art collection. Many of these companies had been on the receiving end of software patent litigation. Reinvention is commonplace in the software industry, and in many cases the patent office had erroneously granted software patents because it was unaware of prior art.

If the patent office was better aware of prior art, would the problem of software patents go away? No. What these companies are experiencing is merely a side effect of the transition from a software industry where software patents are uncommon to a software industry where they are widespread.

When a company finds itself accused of infringing a software patent today, it is frequently able to look back 5 or 10 years — to a time before software patents were widespread — and find prior art that invalidates the patent. What will happen in the future? The company will look back to today — to a time when software patents are being granted in great numbers — and may again find the technique it is accused of infringing was invented earlier. This time, though, the company will not get off the hook. It will just find instead that it is infringing a different software patent than the one in question.

Companies now being harmed by erroneous software patents issued in ignorance of prior art, will be harmed in the future by an ever-growing number of valid patents.

Will the U.S. Benefit Internationally From Software Patents?

Some speakers claimed software patents may benefit very large American companies on the international stage. Such arguments tended to be wrapped up in patriotism.

> Of course, nothing is easier than to excite people on the argument that everything should be made in this country and not imported: though what would happen if anybody really tried to carry that out is the same as would happen if everybody expelled his breath from his body and never drew any breath in.
>
> —Viscount Simon, Parliamentary Debates, House of Lords, 1949.

What is good for General Motors, or other very large companies such as IBM and AT&T, may not be good for America, or for the American software industry. Patents are permitted by the Constitution to promote progress, not to promote the financial interests of IBM and AT&T.

In evaluating the international aspects of software patents it is important to remember that economics is not a zero-sum game. It is also important to remember that trade between two nations, when it occurs, is beneficial to both nations.

There are a number of international issues that were not raised. The economic powerhouse of the American economy is the small business. The Japanese economy, by contrast, is driven by extremely large conglomerates. Software patents appear to favor large corporations with their extensive research facilities at the expense of the small entrepreneur. This might be acceptable if large corporations were equally adept at bringing innovative products to market, but they have often proven ineffective at doing so.

Between 1989 and 1992, Hitachi acquired 30 times more software patents than even that giant of the American software industry, Microsoft. Yet the value of the software Hitachi contributes to the U.S. economy is negligible. Similarly, despite having developed the graphical user interface, Xerox failed in its attempts to bring this important technological software advance to market.

Does Software Progress Quickly or Slowly?

One of the reasons patents are harmful to software is that progress is very fast. Successful software products normally make all their profits in a year or two; then they are replaced by newer versions. Programs a few years old are considered obsolescent.

Yet one supporter of software patents claimed this was false — he claimed software progresses slowly. He illustrated this by mentioning techniques that were known two or three decades ago and are still very useful today.

How can we reconcile these contradictory claims?

There's not really a contradiction. When we say that software progresses rapidly, we mean that new techniques and features appear at a rapid pace; this makes software a few years old very obsolete. At the same time, some software techniques and features do remain useful for decades. We expect natural order recalculation, Quicksort, and public key encryption, to be used for as long as people develop software.

So software moves fast in one respect, and sometimes slowly in another. Which one is the real measure of progress in software? Both, or neither — it depends on your purpose. The purpose at hand is to determine how much harm software patents do and how much good they do.

Because techniques remain useful for decades, programmers five, ten and fifteen years from now will need to infringe today's software patents. This is why software patents can impede development so much. An up-to-date software product will use many new techniques as well as many old ones. The only software you can write without infringing patents is software two decades obsolete; because so many new techniques have appeared since then, few users want such software.

We may also ask how much good software patents do. The answer to this question depends on many factors — such as, whether most of the techniques would be developed and published without patents. Experience until the early 80s, when the software field operated largely without patents, suggests that they would be; that software patents do little good, and we do not need them to encourage progress in software.

If Hardware is Patentable, Shouldn't Software be Patentable too?

The argument has been made that it would be contradictory if an invention was patentable when implemented in hardware, but not patentable when implemented in software.

The purpose of the patent system is to promote progress. If software patents do not promote progress, then permitting them in the name of some sort of legislative uniformity is contrary to the Constitution. Declaring software non-patentable actually provides clarity to intellectual property law. It helps neatly divide between creations having a physical embodiment, secured using patents, and creations purely of the mind, secured using copyright. A division similar to this is apparent in the wording of the Constitution. We therefore do not detect any contradiction in making software non-patentable.

Anything Under the Sun Made by Man

Arguments in favor of the continued granting of software patents made reference to the following quote on patentable subject matter used by the Supreme Court.

> . . . anything under the Sun that is made by man . . .
>
> — Diamond v. Chakrabarty, 447 U.S. 303.

The Supreme Court used this quote with regard to how to interpret the current patent law. Notwithstanding these words, some subject areas remain unpatentable under current law: mathematical algorithms, methods of doing business, and so forth. But when we ask what the law should be — whether the granting of software patents constitutes sound economic policy — reference to the law as it currently stands doesn't help answer the question.

Copyright is Effective and Efficient

Patents are used in other industries to prevent companies from using, but not paying for, the results of their rivals' research and development. Permitting this would be a serious disincentive against R&D investment.

Unlike every other industry subject to patents, the software industry is unique in that its products are also subject to copyright. Copyrights ensures that to be commercially successful, a company choosing to follow another must spend as much to develop a program as the original firm. Indeed, the history of spreadsheets, word processors, and virtually every other software product suggests that it is actually more expensive to follow than to lead. A product that seeks to displace the market leader can only do so by incorporating new features, thereby making it more expensive to develop than the original product.

Copyright is effective because it protects precisely the product that has been developed. It prevents other companies from benefiting by copying your product, while at the same time permitting them to reap the full benefits of anything they develop.

Copyright is efficient because it enables firms to compete on the basis of rival implementations. This competition is vital for the efficient allocation of economic resources. The traditional literal aspects copyright doctrine is also efficient because it has negligible administrative overhead and presents no uncertainties. A small startup has certainty in the knowledge that they control what they create.

Given that copyright law effectively and efficiently achieves the economic aims of the patent system, there is simply no need for software patents.

A Summary of the League's Position: Why Software Should Not be Patented

Appendix B provides an example of why the economic effects of patents vary from one industry to another as a result of industry specific economic factors. A consideration of the economic factors associated with the software industry suggests that the the granting of software patents will harm both the most economically productive sector of the software industry, and the American economy as a whole. This is because the software industry is highly competitive and software patents stifle this competition. Software patents will take profits from firms developing software that solves real customer problems, firms adding significant value to the American economy. Software patents will transfer wealth to firms that add little value to the software industry.

Unlike every other industry subject to the patent system, the software industry is unique in also being subject to copyright. Copyright in its traditional literal aspects form provides an effective and efficient form of intellectual property. There is simply no benefit to be gained from granting patents. Moreover, software patents, by sending the industry the wrong economic signals, are likely to significantly reduce the industry's efficiency.

The League for Programming Freedom believes that the government must take action to prevent software patents seriously reducing the economic efficiency of the software industry.

What Needs to be Done

Software patents appear economically damaging. A number of possible solutions exist. It might be desirable to declare software non-patentable. Or it might be preferable to the limit the scope of the patent monopoly in such a way as to permit the production and use of software intended for a general purpose computer.

It is not worth arguing over which of the many possibilities is most desirable at this stage. A final conclusion regarding the effects of software patents on the software industry should first be reached. Then, if it turns out the effects of software patents are indeed negative, attention can be focused on how to best solve the problem.

Appendix A: Example of How Software Related Patents can be Classified.

By examining all the patents granted during the week of March 19, 1991, we seek to show how it is possible to identify and classify software related patents using any of a number of different characteristics. Verification this was possible was important in determining that there are unlikely to be any problems in formulating legislation to prevent software patents.

The importance of the table below has less to do with the definitions employed, than the fact it shows it is possible to identify and classify software related patents in a number of important and useful ways. The definitions we employed were:

- "Infringed using software": Does the most likely method of infringing the patent involve the use of computer software? Software is defined as being composed of ideal infallible mathematical components.
- "Applicable to general purpose computers": Is it likely that the patent may be infringed by running software on a general purpose computer?
- "Applicable to embedded systems": Is it likely that the patent may be infringed when running software in an embedded system? An embedded system is a computer residing in a special purpose hardware environment, such as an automobile engine control system.
- "Mental process": A mental process patent that can be infringed by mere thought — if you think hard enough, are given suitable input values, and ignore any post solution activity.
- "Standard technique; special domain": A patent granted for the application of a well known technique to solve a problem in a particular software related domain.
- "Claim on generic functionality": A patent that includes a claim worded to cover all possible solutions to the problem being faced. One speaker at the San Jose hearings described such patents as representing the difference between patenting a particular mouse trap and patenting the concept of catching mice.

In compiling this table, we examined only what we felt was the most significant claim of each patent.

patent number	infringed using software	applicable to general purpose computers	applicable to embedded systems	mental process	standard technique; special domain	claim on generic functionality
5,000,030	yes	yes	yes	yes	yes	no
5,000,039	yes	no	yes	yes	yes	no
5,000,041	no	no	no	yes	yes	no
5,000,042	yes	no	yes	yes	yes	no
5,000,147	no	no	no	yes	yes	no
5,000,148	yes	no	yes	yes	no	yes
5,000,149	yes	no	yes	yes	no	yes
5,000,150	yes	no	yes	yes	no	yes
5,000,188	yes	yes	yes	yes	no	yes
5,000,189	yes	no	yes	yes	no	no
5,000,276	no	no	yes	no	yes	no
5,000,281	yes	no	yes	yes	yes	no
5,000,381	no	no	no	yes	yes	no
5,000,382	no	no	no	yes	yes	no
5,000,592	yes	no	yes	yes	yes	no
5,000,711	yes	yes	no	yes	yes	no
5,000,924	yes	no	yes	yes	no	yes
5,001,067	yes	yes	yes	yes	no	yes
5,001,344	yes	yes	no	yes	no	yes
5,001,418	yes	yes	yes	yes	yes	no
5,001,429	yes	yes	yes	yes	no	yes
5,001,447	yes	yes	no	yes	no	yes
5,001,471	yes	no	yes	yes	yes	no
5,001,472	no	no	no	yes	no	no
5,001,476	yes	no	yes	yes	yes	no
5,001,477	yes	yes	no	yes	yes	no
5,001,478	yes	yes	no	yes	no	yes
5,001,479	no	no	no	yes	no	yes
5,001,489	no	no	no	yes	yes	no
5,001,490	yes	yes	yes	yes	no	no
5,001,549	no	no	no	yes	no	yes
5,001,559	yes	yes	yes	yes	no	yes
5,001,560	no	no	no	yes	no	yes
5,001,561	no	no	no	yes	no	yes
5,001,568	no	no	yes	yes	no	no
5,001,569	yes	yes	no	yes	no	no
5,001,573	yes	yes	yes	yes	no	no
5,001,575	yes	yes	yes	yes	no	no
5,001,628	yes	yes	no	yes	no	yes
5,001,630	yes	yes	no	yes	no	yes
5,001,631	yes	yes	no	yes	no	yes
5,001,632	no	no	yes	no	yes	no
5,001,633	yes	yes	yes	yes	yes	no
5,001,634	yes	yes	no	yes	no	yes
5,001,635	yes	no	yes	yes	no	yes

5,001,636	yes	no	yes	yes	no	yes
5,001,637	yes	no	yes	yes	no	yes
5,001,638	no	no	yes	yes	yes	no
5,001,639	yes	no	yes	yes	no	yes
5,001,640	yes	no	yes	yes	no	yes
5,001,642	no	no	yes	yes	no	yes
5,001,648	yes	no	yes	yes	yes	no
5,001,650	yes	no	yes	yes	no	yes
5,001,651	no	no	no	yes	yes	no
5,001,653	yes	no	yes	yes	no	yes
5,001,654	yes	yes	no	yes	no	yes
5,001,660	yes	no	yes	yes	no	yes
5,001,662	no	no	no	yes	no	no
5,001,666	yes	yes	no	yes	no	yes
5,001,677	yes	yes	no	yes	no	yes
5,001,689	yes	no	yes	yes	no	yes
5,001,696	no	yes	no	no	no	yes
5,001,697	yes	no	yes	yes	no	yes
5,001,702	yes	no	yes	yes	no	yes
5,001,706	no	no	yes	yes	no	yes
5,001,707	yes	no	yes	yes	yes	no
5,001,710	yes	yes	yes	yes	no	yes
5,001,714	yes	yes	no	yes	no	yes
5,001,715	yes	yes	yes	yes	yes	no
5,001,724	yes	no	yes	yes	no	no
5,001,727	yes	no	yes	yes	yes	no
5,001,729	yes	no	yes	yes	yes	no
5,001,730	yes	yes	no	yes	yes	no
5,001,736	no	no	no	yes	yes	no
5,001,740	yes	no	yes	yes	yes	no
5,001,742	yes	no	yes	yes	no	yes
5,001,744	yes	no	yes	yes	yes	no
5,001,745	yes	yes	no	yes	no	yes
5,001,747	yes	no	yes	yes	yes	no
5,001,750	yes	no	yes	yes	no	no
5,001,752	yes	no	yes	yes	yes	no
5,001,753	yes	no	yes	yes	no	no
5,001,754	yes	yes	no	yes	yes	no
5,001,755	yes	yes	yes	yes	yes	no
5,001,758	yes	yes	yes	yes	yes	no
5,001,759	yes	yes	yes	yes	no	no
5,001,760	yes	yes	yes	yes	no	no
5,001,761	yes	yes	yes	yes	no	no
5,001,764	yes	yes	yes	yes	no	yes
5,001,765	no	no	no	yes	no	no
5,001,766	yes	yes	yes	yes	no	no
5,001,767	no	no	no	yes	yes	no
5,001,768	no	no	no	yes	yes	no
5,001,769	yes	yes	no	yes	no	yes
all others	no	no	no	no	varies	varies

Patents that apply to software intended for use on general purpose computers are likely to have greater ill-effect than software intended for use on embedded systems. Granting software patents for the application of a well known techniques to particular software fields is also particularly hard to justify on economic grounds. Combining suitable legal definitions for terms used in the above table shows how it should be possible to formulate a definition of statutory subject matter under-pinned by economic rationalism.

Appendix B: Example Showing Varying Economic Effects of the Patent System

As an example of how the patent system is dependent on economic factors that vary from industry to industry, we consider just one factor, the overall size of an industry.

Let's imagine that there are 5,000 people employed by the candle-making industry in the U.S. and that it has been determined based on sound economic principles that the optimal life for a patent in the candle-making industry is 20 years.

Suppose the demand for candles were twice what it actually is. The candle-making industry would be almost twice its earlier size, employing close to 10,000 people. Under a set of economic assumptions reasonable for the candle-making (or software) industry, economics would then dictate a cut in the length of patents for the candle-making industry. Cutting the length of patents by one half would yield roughly the same financial incentive to invent, and thus the same rate of progress as existed earlier. Alternatively we might consider cutting the length of patents by only one quarter. In so doing we are sending a signal to the candle-making industry regarding the increased net economic value of improvements in the candle-making process. This signal however has to be traded off against the negative effect on industrial efficiency caused by the increased lack of competition. When the size of the industry increases the optimal lifetime for patents needs to be shortened. Without knowledge of various factors relating to the inventive process in the candle-making industry the new length for patents is a matter for debate.

It isn't fair to directly compare the software industry to the candle-making industry; the software industry is far larger, but it is also far broader. From the candle-making example it should be possible to understand how the traditional 17 year patent grant may in some industries conceivably hurt progress by stifling competition more than it helps progress by encouraging innovation. The software industry employs some 6 million people. A significant fraction of them develop software. More people are probably engaged in software development than in all other branches of engineering combined. As a result in the software industry reinvention has become common place, and software patents seriously harm competition.