Software Patents: An Industry at Risk

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

(by Gordon Irlam and Ross Williams)

Abstract

Software patents pose a serious threat to the software industry. The League for Programming Freedom comprises over 600 individual software developers, business people, professors, students, and computer users concerned about this threat. The League favors the introduction of legislation to eliminate the threat posed by software patents.

Caveat

Most companies are now being forced to apply for software patents for defensive purposes. The League does not advocate that such companies cease applying for such patents; nor does it advocate that they unilaterally dismantle their software patent portfolios.

Note

This document is large because it contains many appendices containing supporting material. The main text can be read in less than twenty minutes.

Table of Contents

1. The Threat Posed by Software Patents

We can best illustrate some of the problems that software patents cause, through the use of an imaginative scenario written in the style of the well-known computer exploration game "Adventure":

Welcome to Patent Adventure V3.4. --------------- All Rights Reserved 1994.

You are the CEO of AcmeSoft - a fairly successful software company. You are at your desk. Suddenly the fax machine rumbles...

Command> **READ FAX**

It's from SharkTech! They're claiming that your company's product "Acme Professional" violates their software patent "Distinguishing Nested Structures by Color". They want 1% of your wholesale price in royalties.

Command> **GIVE ROYALTIES**

Now they want 5%.

Command> **GIVE ROYALTIES**

That satisfied them. Hmmm, the fax machine is humming again. It's from ParaTech! They're claiming your company's product "Acme Professional" assigns clients to whichever server process is least busy, and as a result want 3% of your wholesale price in royalties.

Command> **IGNORE THEM**

ParaTech have decided to take you to court. Do you want to settle for 10%, pay $800,000 in legal fees, or circumvent the patent?

Command> **CIRCUMVENT**

Your programmers say they can't circumvent the patent without hurting performance --- causing you to lose you 30% of your customer base. Do you want to circumvent?

Command> **CIRCUMVENT**

You've lost 30% of your customers! The fax machine is going again. This time it's from MeanTech. They're claiming your company's product "Acme Professional" violates their software patent on storing document images on a CD ROM along with an automatically generated index, and because they are a competitor, they do not want a royalty. They want you to remove the violating code or stop shipping the product. What do you wish to do: 1) Ship 400 floppy disks instead of one CD ROM. 2) Go to court. 3) Stop shipping the product.

Command> **GO TO COURT**

Legal fees are $600,000. Current funds are $400,000. You've gone broke!

Do you want to play Patent Adventure again?> **DEFINITELY NOT**

While the above scenario is fictional, it is far closer to the current situation than many in the software industry realize. In reality:

- IBM holds patent #4,965,765 which covers the use of different colors to distinguish the nesting level of nested expressions.
- Patent #5,249,290 covers assignment of client requests to the server process having the least load.
- Patent #4,941,125 covers using a digital camera in conjunction with character recognition software to store and index documents on a CD ROM.
- And IBM really does charge small companies 1% of royalties to license a single software patent, and 5% for its entire portfolio.

The software patent system is clearly out of control and if not caught soon will change the face of the software industry forever. Two thousand new software patents are granted each year. They are being granted on software technologies as diverse and mundane as file servers and word processors. It is already dangerous to create products containing data compression or public key encryption, and recently major inroads were made in the field of multimedia. Here are some more examples to help give you a feel for the scope of the problem:

- A spreadsheet in which each cell has a "next cell" attribute defining the next cell to advance to after having entering data into the current cell. [#5,121,499].
- A spreadsheet in which a single cell can contain multiple (possibly optional) fields. [#5,247,611].
- A word processor that has a feature that allows you to specify that a portion of the text should be shaded - such as may be useful when revising a manual - by enclosing the relevant text within commands that turn shading on and off. [#4,924,411].
- Use of a host independent network byte ordering. [#4,956,809].
- A parallelizing compiler that estimates the execution time for each of a number of different parallelization conversions and then selects the one that it thinks will be the fastest. [#5,151,991].
- Simulating the access times associated with a CD ROM by slowing down a hard disk. [#5,121,492].

These are just a few of hundreds of software patents that pose a critical threat to all software developers, large and small. Many more examples are contained in Appendix C. The fact that these patents all cover trivial ideas is significant, but not the only reason for the difficulties. In this document, we argue that the nature of the software industry makes it an inappropriate subject for the granting of patents.

2. What Makes Software Different?

The patent system has served us for more than 200 years. (See Appendix A for a brief explanation of the patent system). Over the years, the patent system has adapted to all sorts of emerging technologies. So why is it failing now?

The reason is that software isn't just something new. Software is something fundamentally different. Abstract and slippery, it doesn't conform to the ordinary constraints of the real world of objects. The nature of software has created not only a different kind of intellectual challenge, but a different kind of industry with its own particular economic structure. To this is being applied a 200 year old patent system. The following section describe the special properties of software that make the application of the patent system inappropriate.

2.1 Software is More Complicated

At the top of the list is the complexity of software. Because software is largely free from physical constraints, complexity has grown to the current state where a single large computer program cannot be completely understood by any one person. As the highly acclaimed computer scientist E. W. Dijkstra has stated:

> *In computer programming our basic building block, the instruction, takes less than a microsecond, but our program may require hours of computation time. I do not know of any other technology than programming that is invited to cover a grain ratio of 10^10 or more. The automatic computer, by virtue of its fantastic speed, was the first to provide an environment with enough 'room' for highly hierarchical artifacts. And in this respect, the challenge of the programming task is without precedent.*
>
> --- E. W. Dijkstra, A Discipline of Programming, 1976.

This capacity for complexity is a great strength because it permits the creation of highly sophisticated products. But it also means that most products, simply by their very complexity, are dependent on a vast range of software technologies.

In most other industries, a product will contain perhaps twenty parts. In the case of sophisticated consumer goods, such as video cameras, we could raise this to 1000 parts. Nevertheless, the constraints of the real world ensure that the complexity of the product cannot become too great. Software, however, is essentially free from these constraints. A major computer program can comprise anywhere from 100,000 to 10 million lines of code. In most other industries a product will involve technologies covered by just a few patents. In the software industry, a product can contain thousands of inventions, any of which might be patented.

For instance, even when buying something as mundane as a word processor, you might be able to choose between a word processor with built-in spelling checker, ability to format multi-column text, and an outline editor; a word processor with proportional fonts, an equation editor, and kanji capabilities; and a word processor that has style sheets, a page previewer, and document interchange facilities. And this is only the start. When you look closely you will find that each word processor actually incorporates thousands of different user visible features. Tens of thousands more features exist inside that are visible only by a programmer. The total number of features contained in something as simple as a word processor is enormous. Thus, patents make the legal risks and expenses associated with developing even well understood software frightening.

2.2 Software is More Abstract

While software's complexity makes a typical computer program dependent on many different software technologies, software's abstraction makes it difficult to partition these technologies.

In most industries patent searches are fairly easy to perform and provide fairly solid results. Patents are typically targeted at a particular product in a particular industry, and as such can be readily classified. For example, a typical patent title might be *Method for increasing grain throughput in a combine harvester by means of an air-forced hopper.*

In contrast, the nature of software means that much of it is very abstract. As a consequence, software patents are often abstract even though their titles can sound specific. For example, patent #5,175,857, *System for Sorting Records Having Sorted Strings Each Having a Plurality of Linked Elements Each Element Storing Next Record Address* has a rather specific-sounding title, but is in fact a rather broad patent covering a well-known algorithm called "Quicksort" when implemented using a linked list. Sorting is a fundamental building block of software, and its implementation using linked lists could be performed by any programmer working in any area of software development --- without the programmer even being conscious he had accidentally "invented" anything. The "Quicksort" algorithm and linked lists both appear in one form or another in many undergraduate Computer Science textbooks.

The complexity of software means that it is dependent on many technologies. The abstraction of software means that it is hard to classify these technologies, so there is a combinatorial explosion of potential patent coverage which removes any kind of certainty about what is patented and what is not. The result is that:

- Software patents are expensive to search.
- Software patents are expensive to analyze.
- Software patents are expensive to fight over in court.

In short, because of their broad coverage and complexity, software patents introduce far more uncertainty than do their non-software cousins. And uncertainty is bad for business. Uncertainty makes it difficult to decide the best strategy to pursue. Which patents might you be in violation of? Will the patent owners take any action? What royalties will they request? Will they sue? Will you be able to get the patent overturned? What damages might be awarded?

These are not questions that can be incorporated into the smooth everyday running of a business. They are not questions comparable with concerns about tuning advertising or production inefficiencies. Rather these are issues that can kill products stone dead and destroy companies.

The penalties for patent infringement can be severe. The most famous case was *Polaroid v. Kodak* in which damages amounted to $900 million - with a further $500 million reportedly being spent by Kodak buying cameras back from consumers. More recently:

> *... a US District Court jury in California awarded $1.2 billion in damages based on the company's claim that Honeywell, which developed the first laser gyros in the 1960s, had subsequently and "willfully" appropriated a special process Litton patented in 1978 for coating the instruments' high-precision mirrors.*
>
> --- Photonics Spectra, October 1993.

2.3 Software Technology Evolves Rapidly

As if complexity and abstraction were not enough, the software industry is developing much faster than other industries --- even the computer hardware industry. Conventional industries typically produce a new generation of products every ten to twenty years. This has been reduced in recent decades thanks to "lean production" and "time-based competition", but even so the rate of product generational change in the software industry is far higher than that of other industries. The presence of patents that last for 17 years is therefore extremely alarming. Think back 17 years. Graphical user interfaces were virtually unknown. Desktop publishing didn't exist. MSDOS didn't exist. Neither did PCs and Macintoshes. With microprocessors doubling in speed every 2 years, this qualitative change in the nature of software is likely to continue. Compare this rate of progress to that of other industries such as the aircraft industry. In software, 17 years is a very long time. The existence of patents 17 years ago on what might then have seemed non-obvious or esoteric technologies would be extremely damaging today. Likewise, much of what may be considered non-obvious today, will be seen as being fundamental and obvious tomorrow.

One consequence of this rapid change is that research is galloping ahead of development. Most industries could be considered to be "waiting" for new ideas. In most industries it costs more money to come up with and evaluate suitable ideas than it does to bring them to market. In contrast, the software industry is completely overloaded with new ideas and innovations. In software, the costs are reversed: it's easy to come up with new ideas, but very difficult to develop products. The idea behind most software patents can be coded in just 20 lines of code, but any program incorporating that idea - along with many others - will be a thousand times larger. It is the writing of a program that takes all the time, not coming up with ideas. The result is that the software industry is awash with innovation and it will take many many years for the industry to catch up, if ever.

This rapid rate of evolution means that those who are investing time creating and lodging patents are vastly outpacing those who are investing effort bringing such ideas to market. By the time an immature technology develops to the point where it can be incorporated into products, it has a dozen or more patents on it that render it commercially intractable.

The consequence of all this is that it is now difficult or impossible to produce new products in the software industry without violating numerous patents. The uncertainty that this introduces into the product development process has to be seen to be believed. The sight of personnel of massive software companies scrambling to rework their software so as to circumvent patents on trivial ideas that were in use twenty years ago, but not documented because they were too obvious, is now a sad reality.

2.4 Software Doesn't Wear Out

A traditional argument for patents in general is that they encourage innovation in fields that have stagnated. Experience shows that most major industries go through a "sunrise" phase of rapid growth during which the first 80% of good ideas are discovered and incorporated into products. Finding these good ideas is usually easy in these early stages, and even some exponents of patents accept that patents may retard an industry during this first phase of chaotic growth. However, they argue that once an industry matures, good ideas are harder to find, and there is a tendency for mature industries to operate complacently (and non-innovatively), merely satisfying their customers' need to replace products that have worn out. There may be scope for innovation, but it might require too high a capital investment in research to be tempting. In mature industries, it is argued, patents encourage innovation by rewarding capital investment in research with a monopoly in the market for the results of the research.

However, this argument doesn't apply to software because software doesn't wear out. A computer program that is fully debugged will perform its function forever without requiring maintenance or modification. What this means is that unlike socks that wear out, and breakfast cereal that is eaten, a particular software product can be sold to a particular customer at most once. If it is to be sold to that customer again, it must be enhanced with new features and functionality.

The inevitable conclusion is that, even if the software industry approaches maturity, any software company that does not produce new and innovative products will simply run out of customers! Thus, the industry will remain innovative whether or not software patents exist. The need for a patent system to encourage innovation in mature industries doesn't apply to the software industry. A mature software industry is not going to gain any benefits from the patent system. And even exponents of software patents, such as Paul Heckel, admit that during the early stages software patents will retard the industry.

2.5 Software has Different Economics

Previous sections have shown that software patents, at the very least, introduce considerable uncertainty and impose a significant business overhead. This would not matter so much if the software industry naturally provided a financial infrastructure that could support a patent system. Software is not the only complex product around: a jumbo jet probably incorporates over 100,000 different parts, many of which will be covered by patents. However, in all such industries the physicality of the parts, and the cost of their mass production, means that their cost dwarfs the legal overheads. A typical jumbo jet part (e.g. a small panel) might cost $100 to manufacture. A typical software component (e.g. a line of code) deployed in a mass market application costs approximately $0.00001 to manufacture.

To see the effect of the patent system on the economics of the software industry, let us model the product cycle in three stages: research, development, and production. The cost of each stage in relation to a product's total cost varies by industry:

	Research	**Development**	**Production**
Pharmaceuticals	Medium	High	Low to Medium
Automobiles	Medium	Medium	Huge
Aircraft	Medium	High	High
Software	Tiny	Huge	Low

Software has a very low research cost because development has not been able to keep up with research, and there are thousands of ideas just waiting to be exploited. Also, a core idea often requires just a few lines of code to implement.

Software has a high development cost because it takes a lot of human effort to write production-quality software. An important aspect of the software development cost is that it mostly consists of people's time. This means that somebody with nothing but a computer and a lot of time on his hands can develop software even though they do not have a lot of capital. Thus, while development is expensive, it is still accessible to the individual - at least for small products.

Depending on how much support is provided, software production costs are low. Each product consists of just a few floppy disks at $0.50 each and a manual which can be printed for less than $5. Typically software sells for in excess of $100.

Now consider the impact of the patent system on these various industries. The cost of patents is proportional to the development cost because it is the amount of stuff that you actually put in your product that determines how many different patents may be involved. In other industries, production costs dwarf development costs, and so the overhead of the patent system (on the development cost) is a minor component in the entire enterprise. However, in software the entire cost is development, and so the patent system represents an enormous cost to the industry. The auto industry would scream if the government affected production margins by just 1%. The software industry is being progressively slugged with what will be a far greater impediment, but so far has not reacted to the threat coherently.

The effect on large companies is that they will have to incorporate the patent process into their software development process, set up bulky legal divisions, get into the business of cultivating defensive patent suites, and perpetually negotiate royalty payments and settle lawsuits. For most big companies that focus on developing software, such action will for a time allow them to survive, for with enough broad and trivial patents in their suite they can threaten virtually anyone who threatens them. But they will also probably encounter companies **that do not develop software**; that are demanding royalties with the gloves off! Because such companies have a distinct advantage when negotiating royalty licenses, it is likely that corporate evolutionary selection pressures will make them more numerous in the future.

Big companies will also experience difficulties with small companies that decide to use broad, but trivial, patents to defend market niches against legitimate competitors. A recent example is Stac Electronics, a small company making data compression software, who apparently bought a software patent from Ferranti in England so that they could prevent Microsoft from including a data compression feature “Doublespace” in MSDOS V6.0. This lawsuit was launched over a year ago, is still going, and has cost both sides huge amounts of money.

The effect of software patents on large companies is bad enough, but to a small company it can be crippling. Large companies may already have a legal infrastructure, but most small companies must rely on the advice of external professionals who charge what seem high rates. Large companies may for a time be able to accept patent lawsuits in their stride, but small companies can be wiped out by a single one - fair or not.

For many small companies, the prospect of being sued over a patent infringement **even if the case is ungrounded and would ultimately fail** is so terrifying, that many companies choose to give all patents they know about a wide berth rather than risk the possibility of any kind of patent challenge. Patents and patent laws are so complex that even an ungrounded lawsuit may take a year to resolve, simply because it may be hard to prove quickly that the other side does not have a case. Meanwhile hundreds of thousands of dollars in legal fees will be spent, crippling the target software company.

Thus, whereas most large pharmaceutical and aerospace companies can afford to conduct ongoing patent battles to resolve the scope of various patents, the small players of the software industry cannot. As a result, they will attempt to steer well clear of patents, making the patents even more powerful than they were ever intended to be.

In summary, the marginal cost to produce software is very low. The value of that software in the marketplace is often very high. Therefore, if sufficient volume is attained, profit margins will also be very high. This is the main reason the software industry is able to attract venture capital, and is a reflection of the value the industry is delivering to society. Software patents, by introducing uncertainty and requiring the payment of unavoidable royalties, have the potential to destroy this leverage.

2.6 Software is Successful Because of Market-Driven Properties

As we have seen, the software industry has a surplus of new ideas and technologies. What happens to them all? Well, of course, eventually they appear in products. However, because of the complexity of the products, the process by which this happens is more involved than in other industries. Typically, the technology is so complicated that bringing it to market as a simple, workable product is a greater challenge than performing research to create the enabling technology. Because of this, today’s successful software companies have become successful largely because they are market driven, rather than technology driven companies. These companies didn’t build software because they thought it would be fun to build something no-one else had built before; rather they set out to build products that would meet the real needs of the market place. As Oracle Corporation clearly states:

> *Whether a software program is a good one does not generally depend as much on the newness of a specific technique, but instead depends on the unique combination of known algorithms and methods. Patents should not protect such methods of innovation.*
>
> --- Oracle Corporation Policy on Software Patents (See Appendix F).

Borland didn’t invent compilers. Microsoft didn’t invent operating systems. Novell didn’t invent networking. Sun didn’t invent Unix. Apple didn’t invent the graphical user interface. Oracle didn’t invent the database. It turns out that nearly all successful software companies have concentrated on constructing better implementations of already existing technologies. The market rewarded these companies because they provided the market with what it wanted: products, not ideas. These companies didn’t have a horde of researchers working in obscure fields in the hope that one of them would discover something useful. In the software industry ideas are like air. The hard part is deciding which ideas to choose. The focus of these companies was on “doing it right” rather than on “doing it first” or “doing it differently”. By allowing other companies to monopolize new technologies, patents strike at the very essence of the software industry’s business philosophy.

All of the above mentioned companies have relatively few software patents (see Appendix D). Furthermore the patents they do hold tend to have relatively narrow claims. For instance something like *enhanced error recovery in a network file system cache* rather than *image storage and retrieval system*. This is a result of their focus on developing the right product at the right time, rather than on trying to be the first to leap into every new technology. Opportunities for mainstream software developers to obtain important software patents will be somewhat limited. Most companies don’t spend their time thinking about what they can do that no one else has done before, but on which techniques to turn into products. Software patents will hurt successful companies because they will prevent them from doing what they do best: bringing technology to market, not because it is new, but because the time is right.

In summary, it is easy to be the first to develop a new software technology (such as desktop video). The hard part is to transform that technology into a useful product that solves a real customer need. By rewarding research companies rather than development companies, software patents harm an industry whose value is largely a result of development.

2.7 Conclusions

The software industry is significantly different from other industries that are subject to the patent system:

- Software is More Complicated.
- Software is More Abstract.
- Software Technology Evolves Rapidly.
- Software Doesn’t Wear Out.
- Software has Different Economics.
- Software is Successful Because of Market-Driven Properties.

These differences make application of the patent system at best ineffective, and at worst potentially disastrous. The complex nature of software patents, their abstraction and broad scope, their excessive longevity, and their capacity for detrimental economic transformation, make software patents a potentially paralyzing problem for the software industry.

3. The Problem of Software Patents

3.1 Problems Developing Software

As a result of software patents, many areas of software development are simply becoming out of bounds. A good example is the field of text data compression. There are now so many patents in this field that it is virtually impossible to create a data compression algorithm that does not infringe at least one of the patents. It is possible that such a patent-free algorithm exists, but it would take a team of patent attorneys weeks to establish this fact, and in the end, any of the relevant patent holders would be able to launch a crippling unfair lawsuit anyway. For the small company, even tiptoeing through the mine field is not good enough. The mines do not need to go off to be damaging.

A similar situation probably exists for most other relatively new fields of software development: neural networks, hypertext, public key encryption, pen based computing, multimedia, “groupware”, and so on.

This pattern is set to continue. Until they are eliminated, software patents will likely jam up development of all future new areas of software technology.

3.2 Problems in the Courts

> *I’m not familiar with any type of litigation that is any more costly than patent litigation.*
>
> --- R. Duff Thompson, VP and General Counsel, WordPerfect.

Numerous software companies now find themselves facing threats or lawsuits relating to software patents. Indeed a large software company might face perhaps 5 or 10 such threats at any one time. As with most other legal matters, there is a tendency to try and keep such matters quiet, but occasionally they do spill over into public view. Examples of some known software patent disputes are contained in an Appendix C.

IBM has an extremely large software patent portfolio. Very little is known about how much different organizations have to pay to license it, but a large software company should probably be prepared to budget somewhere around $10 million per year. A small software company may need to budget anywhere up to 5% of gross revenues.

In addition to the threat from large companies, there is also the risk of threats from “independent inventors”. Take Roger E. Billings, founder and first graduate of his own “International Academy of Science”:

> *Novell Inc. is bracing for a battle that could be fiercer that anything the company has faced in the network software market. This battle will be in the courts, where both Novell and one of its largest customers will try to prove that a patent they are charged with infringing is not valid.*
>
> *In a suit filed in U.S. District Court, Northern California, last December, Billings claims Novell’s NetWare network operating system infringes his patent when used for distributed computing. Billings applied for the patent in February 1982, nearly a year before NetWare hit the market; he won approval in 1987 after twice being rejected by the U.S. Patent Office. Billings wants Novell to fork over royalties representing 8% of total NetWare sales through the trial date, or about $220 million.*
>
> --- Information Week, March 16, 1992.

The case is still in the courts. So far Billings has met with some success. Billings claims his patent #4,714,989 is for the concept of a “file server”, however, the wording of the patent makes it difficult to know exactly what it covers. Irrespective of the merits of this case --- who invented what when ---just the possibility of 17 year patents on enabling technologies such as this will have a chilling effect on both the software and hardware industries.

3.3 Problems in the Patent Office

There are numerous problems in dealing with software patents within the patent office:

- A lack of computing expertise.
- An inability to properly search for prior art.
- Difficulties in classifying software patents.
- An excessive period of pendency - typically 2 years. This time period is unacceptably slow in comparison to the rate at which software technology advances. Investment decisions are having to be made in the presence of great uncertainty over whether a competitor holds a key patent covering the technology being considered.

Other problems that are sometimes blamed on the patent office, but are not entirely its fault:

- The application of very low criteria for judging whether a patent is non-obvious and novel.
- The 17 year patent term is totally inappropriate to software.

These problems are more properly problems that reflect problems in the underlying legislation, or at least the way the courts have chosen to interpret the legislation.

While problems in the patent office may be interesting to talk about, we feel that fundamentally it is a secondary issue. The real problem with software patents has now become a legislative one. A legislative approach is now called for to resolve it.

3.4 Conclusions

The software industry is significantly different from other industries. As a result software patents are causing numerous problems for the software industry:

- Problems Developing Software.
- Problems in the Courts.
- Problems in the Patent Office.

The problem of software patents is already having a direct effect on the software industry. But this is only the start. If software patents continue unchecked, the underlying structure of the software industry is going to be significantly altered.

4. The Effect of Software Patents

4.1 Current Corporate Behavior

> *"Lawyers are running around our industry asking people if they'd like to patent something," said Ken Wasch, executive director of the Software Publishers Association in Washington, D.C. "It's gotten worse than ambulance chasing, and we don't think it's a positive development."*
>
> --- Chicago Tribune, March 20, 1989.

Most large software companies are by now well aware of the threat that software patents can pose to their business interests, and as a method of protection are attempting to build up "defensive" patent portfolios that can be cross-licensed with other large corporations.

This provides protection against other large companies, but there seems little possibility for any sort of guaranteed protection against people like Roger Billings, who are probably not interested in signing a cross-licensing agreement. Against such people the future of these companies is very much at the mercy of the courts, and the amount of damages that they choose to award.

For the small software company it obviously is not possible to build up any sort of serious defensive portfolio. The small company has to be prepared to pay whatever license fee the big company demands --- or find a business other than developing software. It likewise needs to be prepared for the possibility of a bloody fight against another small competitor.

One anonymous vice president of a major software company crystallized his corporation's dilemma and consequent decision to register software patents, despite widespread unease within the company, by saying: "How does a just man live in an unjust world?"

It is a common nostrum that patents "protect" small companies from competition from bigger ones. In reality, it usually doesn't work that way. Normally, the largest companies own most of the patents, and use them to force other companies, both large and small, to cross-license with them. Roger Smith, Assistant General Counsel for IBM, explains how this works:

> *The IBM patent portfolio gains us the freedom to do what we need to do through cross-licensing --- it gives us access to the inventions of others that are the key to rapid innovation. Access is far more valuable to IBM than the fees it receives from its 9,000 active patents. There's no direct calculation of this value, but it's many times larger than the fee income, perhaps an order of magnitude larger.*
>
> --- "Think" magazine, #5, 1990.

Thus, if a small company tries to use a patent to "protect" itself against competition from IBM, IBM can usually find patents in its collection which the small company is infringing, and thus obtain a cross-license. Besides which, if you are a small company, do you really want to try taking IBM to court?

The need to take out "defensive" patents is likely to be detrimental to the overall profitability of the software industry. As shown in Appendix C, there is a strong negative correlation between a company's propensity to patent and its ability to bring to market innovative software --- software that proves itself capable of filling a real customer need.

4.2 Patents as a Selection Effect in Corporate Evolution

Strong parallels can be drawn between the functioning of the market and the notion of survival of the fittest from evolution theory. The successful software company of today has succeeded because of its ability to develop and bring innovative products to market. However, software patents will make these traits far less desirable. The desirable traits will be those embodied in IBM, Hitachi, and AT&T: an ability to produce software patents without producing software products. These traits will be rewarded. Some companies may be able to adapt to these changes. Those that can't will become extinct.

To survive, software companies will need to de-emphasize developing new products to solve real customer needs and emphasize activities more likely to result in obtaining broad patents in newly emerging fields.

Companies that choose to develop and market significant products should expect and plan for suits alleging patent infringement. Having a large patent portfolio will prevent threats from competitors, but will only provide a limited defense against those that produce little in the way of products.

The resulting changes in the make up of the software industry won't happen overnight, but rather, over a period of perhaps ten years. A large company that chooses to continue to concentrate on producing only software will be able to keep doing so for quite a while. Eventually, however, it will find most future areas of technology restricted. Then, either as the result of a single calamitous award for damages, or the cumulative costs of the patent system, it will find it has become uncompetitive.

4.3 The Future

A vision of patents entrenched in the software industry is a vision of stagnation. A vision of IBM once again calling the shots. A vision of companies like Xerox and AT&T who have proven incapable of bringing innovative products to market stealing profits from those companies can.

Such a vision is particularly alarming to the successful software company of today, the company that is skilled in building and bring products to market. See Appendix E for a list of some of the organizations and individuals that have voiced concern about the threat to the software industry posed by software patents.

Those that have few or no products to sell are likely to pose a serious threat to those that do.

An example of a successful software company of the future might be Public Key Partners. Instead of building and marketing a real product, it purchased the patent rights to a technology. It now collects royalties from companies capable of integrating and marketing products containing this technology.

Being property, patents can be bought and sold. Some companies specialize in acquiring and litigating patents. Such companies present another example of the software companies of the future.

Lastly we might see the software equivalent of Gilbert Hyatt. He files very broad patents relating to some emerging technology, contests the claims with the patent office for a significant period of time, and when the patent finally issues, attempts to collect sizable royalties for the next 17 years. See for example patent #4,942,516 originally filed in 1970, finally issued in 1990, and titled *Single chip integrated circuit computer architecture*:

> *North American Philips Corporation ... today announced the signing of a license agreement for two portfolios of Hyatt's patents.*
>
> *The Computer-Related Patent Portfolio covers technologies including: fundamental single-chip microprocessor architecture; dynamic random access (DRAM) chip memory refresh techniques; intelligent keyboards for computers; techniques for creating random access memory (RAM) pages or blocks (for memory management purposes); computer-to-computer communication and serial communication, such as in networks or automobiles; and, control of machines by micro computers. In the case of the latter, machines include disk drives, printer control in PCs; tape control in camcorders and VCRs; and control of automobiles.*
>
> *The LCD-Related Patent Portfolio covers technology including: LCD television displays; projection LCDs; shades of intensity and color for LCDs; high intensity illumination and thermal control for projection LCDs; and other related inventions.*
>
> --- PR Newswire, November 6, 1991.

Perhaps disturbingly, such people often tend to be viewed in the media and by juries, as being in some sense unsung heroes who have had their inventions misappropriated by "evil corporations". (Ford was recently ordered to pay $5 million dollars in damages to Robert Kearns for the patent he has on the intermittent windshield wiper.)

4.4 A Question of Economics

Consider the following equation which every software developer now faces. **Software patents are harmful to you if:**

royalties you receive	+	profits gained by impeding competitors with patents	<	royalties you pay	+	profits lost by being impeded by competitors' patents	+	legal and other administrative costs

For companies whose focus is on building and bringing innovative software products to market the first and second of these five terms will be relatively small. The third term will be quite large due to the likes of IBM. It also has a large positive uncertainty because some individuals with patents but with no product of their own requiring cross-licensing may negotiate crippling royalty contracts. The fourth term is not yet large, but as more and more fundamental technologies are patented it will rise rapidly. Some areas of technology, such as data compression, are already intractable. The fifth term is relatively small for large companies, but a crippling overhead for small developers whose entire capital outlay may be less than $1 million. This last term consists entirely of destroyed wealth.

4.5 Conclusions

The current problems in the software industry caused by software patents are really just a symptom of what, if left unstopped, is going to constitute a far more fundamental change. Resources are already being diverted away from developing software and towards building up defensive patent portfolios. However the overall influence of this in changing the software industry will be relatively minor.

The major influence on the software industry will result from the competitive disadvantage suffered by those firms that choose to build software products to fulfill market needs, relative to those firms that choose to de-emphasize direct product development in favor of attempting to monopolize emerging software technologies.

A vision of the likes of Hitachi and IBM being in control of the software industry is particularly disturbing to those companies that are able to successfully build software products to meet customer needs. In addition, the industry is likely to be haunted by firms and individuals that produce little, but demand much.

On the flip side, of course, there are going to be benefits for those who register particular software patents. But given the almost nonexistent amount of research required to develop a patentable software idea, ownership of a software patent is more akin to winning a minor (or major!) lottery than a reward for years of research. Directing money towards these people will only be bad for the industry.

If software patents continue to be issued, software development will become expensive and dangerous, something like a trip through Jurassic Park. Action must be taken soon if corporate dinosaurs from the past are not to rule the earth again.

5. Options for the Government

Software is different!

Software patents are causing problems!

Software patents will significantly harm the software industry!

5.1 Options

So what are the options?

- **Make No Changes:** This document shows why this would be disastrous.
- **Modify the Patent System:** Proposed modifications include:
  - Tighten up the requirements for awarding software patents.
  - Reduce the duration of software patents from 17 years to, say, 3 years.
  - Significantly reduce the period of pendency.
  - Find a simpler way to determine if a piece of code is affected by a patent
  - Improve patent indexing so that software patents can be more easily searched.
  - Publish patent applications as soon as they are received.

  Ideally the patent system should be managed in a way similar to the Federal Reserve: as a carefully monitored dynamically balanced instrument of economic policy. This would involve using feedback of the rate and cost of innovation within each industry to determine the length of time for each patent. Unfortunately the Patent Office is largely bereft of economic experience, the economics of innovation are not well understood, and any changes to the patent system that affect other industries are politically infeasible.
- **Abolish Software Patents Completely:** This is the option we advocate. The industry thrived and prospered without software patents. There is simply no real need for them.
- **Set Up a New System of Patents Tailored for Software:** This idea has some merit. Some software ideas really do require a lot of research to arrive at and some kind of system to protect them could be a good thing. But the current system is completely inappropriate.
- **Have Copyrights and Patents be Mutually Exclusive:** This would allow software to be copyrighted or patented, but not both at the same time. It would be permissible in a copyrighted work to freely make use of a patented invention only if no attempt was made to patent any of the improvements made to that invention, or to patent any other inventions contained in that product. This would provide the software industry the freedom to choose the system best suited to the development of software.

Despite the merits of some of these proposals, **we advocate that software be made explicitly non-patentable**. Here's why:

- While in theory is may be possible to modify the current software patent system to be fairer, it is unlikely that this will be possible in practice. The Patent Office is a 200 year old institution with its own culture and history. It has been forced to apply the general patent system to software without being able to compensate for the special nature of software and the software industry. This has been evident particularly in the low threshold of acceptable inventiveness, which while acceptable in other industries, is a disaster for the software industry. Whether or not an acceptable patent system can be constructed for software, it remains true that the current patent system is so totally wrong when applied to software, that it is hard to imagine that it could evolve into something acceptable. A total break with the past seems the only solution.

- It is clear from the past. This solution works!
- Muddling about with vague proposals for alternative systems constitutes a complicated solution to what is really a very simple problem.
- Of all the thousands of software patents we have seen, the number that might be able to truly justify patent protection could be counted on one hand. From the simple perspective of numbers, **the most direct path to the best outcome is abolition**.

A secondary issue is what to do about existing software patents. If people have made decisions based on such patents, they may need time to adjust. One possibility would be to reduce the duration of these patents to, say, 3 years rather than 17 years, so as to reflect the pace of the software industry. This should be an acceptable outcome for everyone.

In the long run, past patents are a minor consideration. The growth rate of software patents means that the number of existing patents will eventually be dwarfed by the number filed in the future. The important thing is to stop any more software patents from being issued.

5.2 An Invitation

The League for Programming Freedom is concerned about the threat software patents pose to the future of the software industry.

The League will be happy to provide clarification of any aspect of its testimony, or to provide any additional information that may be requested of it relating to software patents.

The League would also be happy to meet with the Commissioner (or nominee) to better discuss various aspects of the issues which it has raised. Information on the League, including how to contact it, is contained in Appendix H.

Lastly, the League realizes that the legislative changes it feels are most appropriate to solve the problems associated with software patents clearly fall outside the direct powers of the Commissioner. However, the League feels that through contact and interaction with the Commissioner it will best be in a position to assist the legislature in formulating an appropriate solution to the problem.

The League looks forward to the possibility of being able to work with the Commissioner on these matters.

Appendix A: What is a Patent?

A patent is a monopoly right created by the government to a new invention. It provides the holder the legal right to prevent others from making, using, or selling the invention, or any product containing the invention for a period of 17 years. The holder has to go to court to enforce this right. Independent reinvention does not constitute a defense against the charge of patent infringement.

Patent holders typically grant third parties license to use the invention in return for an agreed license fee or royalty. Licensing a patent does not automatically permit the licensee to produce the named invention. Frequently an invention will be covered by several patents and it is necessary to license each one.

A patent is obtainable by a third party for an improvement to an already patented invention just as readily as on anything else.

The patent office interprets the notion of what constitutes an invention very broadly. Important inventions and trivial applications are equally patentable. There is a requirement that prohibits the obtaining of a patent if:

> *... the subject matter taken as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.*
>
> --- 35 USC 103.

However, this means exactly what it says: non-obvious. It does not mean the invention has to be in some sense above the norm, or in any way clever. The patent office takes the attitude that if something has never been built before, then presumably, the invention is non-obvious. This causes significant problems in the software industry due to the rapid rate of technological change.

Brian Kahin, at Harvard's Kennedy School of Government, states that the nature of what is considered to be an invention is "often at a level of abstraction that is shocking to the uninitiated". This point is central to any understanding of the workings and potential impact of the patent system on the software industry.

Patents are quite different from Copyright. Copyright merely covers a particular piece of writing. Patents cover the underlying idea. This document is copyrighted. This prevents you from being able to change or redistribute this document in ways the original authors dislike. If the authors were the first people to come up with the idea of writing a document on software patents, and it was possible to patent this idea, then they could prohibit you from writing and distributing your own documents on the subject of software patents.

There are at least three important vantage points from which the patent system may be considered:

- Law: as a matter of jurisprudence.
- Economics: as a public policy issue.
- Business: as a matter of financial self interest.

This document focuses on the direct business impact of patents, but first here is a brief discussion of the other two.

Most people who deal with the patent system have some sort of legal background. It is not surprising then that the legal technicalities of the patent system gain the most study. For our purposes however the precise legal details of the patent system are only of marginal interest. Suffice to say patent legislation has a long history dating back to the English Statute of Monopolies of 1623. The U.S. constitution granted Congress the right to issue patents subject to certain constraints.

The economic rationale for the patent system is that on account of the appropriable nature of inventions it is necessary to grant patents so as to provide an incentive to invent. Economists tend to be slightly uneasy about the patent system on account of its ability to stifle competition.

Taking a business point of view, all that really matters about the patent system is the bottom line. Will the existence of fewer patents increase or decrease your profit margins? This is a difficult question to answer, but we believe that the patent system is detrimental to much of the software industry. The reasons for this are discussed in much more detail elsewhere.

While a few earlier examples can be found, essentially beginning in the early 1980's and in response to court decisions, the patent office started to grant patents on inventions that included software. Different companies realized this at different times and started submitting applications accordingly. It is fair to say that by 1990 most of the computing industry was well aware of this policy and had at least started submitting software patent applications. Since it typically takes 2-3 years for an application to be approved, it is only relatively recently that the full effects of this policy are becoming apparent.

Roughly, 2,000 software patents are issued each year. The total number of software patents in existence is probably around 10,000.

As a technical point, the patent office maintains that algorithms per se are not patentable. This is indeed the case, although for all practical purposes algorithms may as well be. An algorithm in the abstract is not considered patentable. However, an algorithm when used to solve some particular problem is considered patentable. Thus the "RSA algorithm" is not patentable, but "use of the RSA algorithm to encrypt data" is patentable. If it turned out that you suddenly decided you could use the RSA algorithm to produce a stream of random numbers you would not be infringing the RSA patent. This is however a very fanciful occurrence. For all practical purposes, such patents can be considered patents on algorithms.

Appendix B: Examples of Software Patents

This appendix contains examples of currently existing software patents. We have chosen software patents from a range of fields to indicate their diversity and scope. Perhaps the most alarming aspect of this list is that these examples do not represent the worst cases; there are hundreds more patents that are as trivial and as broad in scope as the ones listed here. For all practical purposes, no invention is too trivial to be patented.

B.1 Word Processors

- Any word processor with a separate mode that the user selects when they wish to type in a mathematical formula. [#5,122,953].
- Any word processor screen layout that simultaneously displays the global page heading/footing and the contents of the current page, and permits you to edit either. [#4,984,162].
- Any word processor that has a feature that allows you to specify that a portion of the text should be shaded - such as may be useful when revising a manual - by enclosing the relevant text within commands that turn shading on and off. [#4,924,411].
- A word processor which marks and makes correction to a document using two additional different colors. [#5,021,972].
- A word processor that monitors the sequence of keys you type and tries to teach you about new features. If it notices you doing a particular sequence several times it will display information about a simpler command sequence that may help you do what you want. [#4,947,346].
- Use of different colors to distinguish the nesting level of nested expressions in computer programs. [#4,965,765].

B.2 Spreadsheets

- Any spreadsheet that can automatically collapse rows that are hierarchically subordinate to another row. [#5,255,356].

- Any spreadsheet in which a single cell can contain multiple (possibly optional) fields. [#5,247,611].
- Any spreadsheet in which each cell has a "next cell" attribute defining the next cell to advance to after having entering data into the current cell. [#5,121,499].

B.3 Operating Systems

- A file server that merges together multiple pending writes that require updating the same meta-data. [#5,218,695].
- Remembering file access behavior and using it to control the amount of read-ahead the next time the file is opened. [#5,257,370].
- Altering the working set of a process based upon its paging behavior and how its paging behavior changes in response to changes in its working set size. [#5,247,687].
- Creating variable size disk partitions comprising tracks residing on multiple disks. [#5,129,088].
- Assigning a client request to a server process by examining all the server processes not handling the maximum number of clients and then assigning it to the server process currently servicing the fewest clients. [#5,249,290].

B.4 Compilers

- Any parallelizing compiler that estimates the execution time for each of a number of different parallelization conversions and then selects the one that it thinks will be the fastest. [#5,151,991].
- Using condition code graph analysis in a CPU simulator to determine whether it is necessary to simulate the generation of the condition codes. [#4,951,195].
- Caching the most recent branch target code when simulating a procedure return instruction. [#5,167,023].

B.5 Miscellaneous

- Any document storage system that has a digital camera to scan in documents, stores the documents on an optical disk, and uses character recognition software to construct an index. [#4,941,125].
- Generation of random numbers by feeding the output of one random number generator into the input of another random number generator. [#5,251,165].
- The computer graphics representation of a surface using and array of dots, rather than the more traditional wire frame model. [#5,257,347].
- Quicksort implemented using a linked list of pointers to the objects to be sorted. [#5,175,857].
- A calendar tool that includes a bar graph of the duration of each meeting and a composite bar graph of all meetings. [#5,247,438].
- Simulating the access times associated with a CD ROM by slowing down a hard disk. [#5,121,492].

Appendix C: Examples of Software Patent Disputes

Most companies that are threatened over patent infringement probably prefer to keep the matter quiet. Therefore the current direct impact of patents on the software industry is not fully known. The following examples probably heavily understate the true effect.

The biggest news at Comdex this year was the announcement of the Compton's patent. Compton's, a spin off of Encyclopedia Britannica, claim their patent covers multimedia searching. Their announcement received a very hostile response from the press. Compton's had been threatening everyone in the industry with a 3% license fee before the Patent Office spontaneously decided to re-examine the patent. This decision must have been a result of all the media attention the patent was receiving. Other than that, there was nothing particularly unusual about this patent. Hundreds more equally broad software patents currently lie dormant in the Patent Office.

Lotus, Microsoft, and Ashton-Tate have all been sued by Refac, a litigation company, for a patent it acquired, #4,398,249, that contains a very broad claim covering "natural order recalculation" used in spreadsheets. Fortunately the case got thrown out on a legal technicality. The patent in question was filed in 1970, but wasn't issued by the Patent Office until 1983.

Paul Heckel has threatened Apple and IBM over patent #4,736,308 which he alleges is infringed by HyperCard and ToolBook respectively.

Cadtrak has collected large sums of money and successfully defended patent #4,197,590 on the concept of an "xor cursor".

XyQuest was forced to remove features from the latest release of the XyWrite word processor after being threatened by Productivity Software. Attempts to license the features proved unsuccessful as Productivity Software increased the fees every time XyQuest attempted to reach agreement.

Mark Williams Company has harassed various software companies over patent #4,956,809 on the (very fundamental) idea of a host independent network byte ordering.

AT&T is finding itself free to start exercising its muscle. It first threatened members of the MIT X consortium alleging that the X11 windowing system was in violation of patent #4,555,775 which it holds on the concept of backing store. AT&T is now suing MCI for alleged software patent infringement.

Novell is being sued for $220 million dollars by Roger Billings for infringing his patent #4,714,989 on the concept of a file server.

The fields of cryptography and data compression are essentially off limits to programmers on account of patents. Numerous companies have been forced to obtain licenses from Public Key Partners, which in turn purchased key patents from Stanford and MIT to create an outright monopoly on public key cryptography. Unisys has threatened people over a data compression algorithm that is also used in the popular Unix "compress" program.

Microsoft is being sued by Stac Electronics as a result of Microsoft's incorporation of transparent data compression in MSDOS 6.0. The main patent involved is #5,049,881.

Appendix D: Negative Correlation of Innovation and Software Patents

In what may be judged as either ironic or deeply disturbing, most software patents are held by companies that history has proven, and those within the computing industry judge, to be totally incapable of delivering innovative software products to market:

> *Interestingly, approximately 12% of all software patents are owned by IBM (roughly 1,000), and no other companies come close (the importance of this being that Microsoft just paid $20,000,000 to license IBM's software patents). Other American companies with many software patents include ATT and the Bell Laboratories, Xerox and DEC, while Hitachi has the most patents by a Japanese company (roughly 450), along with Toshiba, Fujitsu, Fanuc, Sharp and Mitsubishi.*
>
> --- Gregory Aharonian, Communications of the ACM, January 1993.

The following constitute our best estimates of the number of software patents granted to various companies between 1990 and 1992 (the results appear similar to the above, though they are not identical):

Software Patents Granted 1990 - 1992

IBM	500	Fujitsu	50	Lotus	7
Hitachi	400	HP	50	Novell	1
AT&T/Bell	150	Sun	50	Borland	0
DEC	150	Unisys	30	NeXT	0
Toshiba	150	Apple	20	Oracle	0
Sharp	100	Texas Inst.	20	Pyramid	0
Xerox	100	Microsoft	13	SGI	0
Canon	70	Intel	10	Sybase	0
Motorola	70	Matsushita	9	Symantec	0
Wang	60	Adobe	8	WordPerfect	0

Total software patents granted (1990 - 1992):	5000
Entities with fewer than 5 s/w patents:	1000
Entities with 5 or more s/w patents:	60

Because of the way patents are classified it is very difficult to gather accurate data on how many software patents exist. Also differences of opinion as to what precisely constitutes a software patent can also muddy things. The above data is indicative of the overall situation, but individual figures may have errors of anywhere up to 50%.

The above table tends to suggest a significantly negative correlation between the number of software patents granted to a company and its ability to bring innovative software products to market. Companies that form the backbone of the software industry: Microsoft, Adobe, Lotus, Novell, Borland, Oracle, and Sybase, have relatively few software patents, while companies that hardly market any software: Hitachi, AT&T, Toshiba, Sharp, and Xerox, have many.

As an example of this, consider Sun's Network File System, NFS, which Sun designed and developed, which was for its time a highly innovative product, and which went on to become the standard file service protocol throughout the Unix industry. Although far from conclusive, a search for the string "NFS" on a small database of some 2000 patent abstracts which one of the authors maintains turned up five patents assigned to IBM, one to Auspex, and none to Sun. This is despite the fact that Sun developed NFS, and the other two companies have engaged in no more than the most trivial of tinkering around the edges.

When asked to name some companies responsible for the production of innovative software, Hitachi isn't one of the companies most people immediately think of.

IBM has a very strong software patent portfolio. It is oversized even in proportion to the size of IBM itself. This is a result of IBM's patenting every single trivial idea every employee ever comes up with, rather than having any great propensity to be truly innovative. IBM has never been considered synonymous with innovative software. IBM even has a patent, #5,247,661, on a software application to permit employees to automatically document ideas for later patenting.

Fortunately, when IBM was being investigated for antitrust (some time ago) it issued a consent degree permitting the automatic licensing of its patent portfolio. As a result any one patent can be licensed for 1% of royalties, and the entire suite for 5%. In this regard the downsizing of IBM that is currently occurring is cause for considerable concern. If IBM ever feels free to start exercising its full powers, its patent portfolio could pose a considerable threat to the entire computer industry. It has already recently increased the fee to automatically license its entire suite from 3% to 5%. The possibility of IBM selling off various divisions or deciding to break up is also cause for concern. A worst case scenario as far as the rest of the computer industry is concerned would involve some or all of IBM's patents winding up in a company that produces few or no real products.

None of the hardware or software companies that collectively constituted the "microcomputer revolution" hold significant numbers of software patents. Companies such as Microsoft, Borland, Novell, Adobe, Lotus, NeXT, Intel, Apple, Sun, and SGI all have relatively weak software patent portfolios. These are the companies that have created wealth in the computer industry over the last ten years by developing new and innovative products. They are very much responsible for turning the industry into the vibrant place it is today. Without these companies, the software industry would be virtually nonexistent.

Between 1990 and 1992, software patents were granted to roughly 1,000 different people and organizations. This tends to confirm the theory that entities that individually play only a very small role in the overall software industry will be able to obtain patents on various key software techniques.

Appendix E: Opposition to Software Patents

Articles discussing the threat posed by software patents have appeared in numerous newspapers including the New York Times, Wall Street Journal, and Washington Post.

Pamela Samuelson, Professor at the University of Pittsburgh School of Law, and Brian Kahin, Adjunct Research Fellow at Harvard's Kennedy School of Government, are both speaking eloquently against the current patent office policy and practice of patenting software.

Surveys by organizations such as the Association for Computing Machinery show a strong opposition to software patents amongst its members. Many academic computer scientists are willing to speak out against software patents.

WordPerfect Corporation has expressed considerable concern regarding software patents. They currently receive an average of one letter a month alleging patent infringement and threatening legal action. This is probably not atypical for a large software corporation.

Mitch Kapor, the original founder of Lotus, recently attested before Congress as to the danger software patents pose (see Appendix G).

Phillipe Kahn, president of Borland International, is known to share similar concerns.

"I'm kind of scared about the climate for the next 10 years" says Dan Bricklin, coinventor of VisiCalc, the world's first electronic spreadsheet.

Jim Warren, founder of InfoWorld, is likewise equally strongly opposed to the patenting of software related inventions.

Oracle Corporation recently issued a detailed statement opposing the granting of patents on software (see Appendix F).

Autodesk is also about to issue a statement against software patents.

Appendix F: Oracle's Policy on Software Patents

The following statement on software patents was released by Oracle Corporation, a major database vendor:

Oracle Corporation opposes the patentability of software. The Company believes that existing copyright law and available trade secret protections, as opposed to patent law, are better suited to protecting computer software developments.

Patent law provides to inventors an exclusive right to new technology in return for publication of the technology. This is not appropriate for industries such as software development in which innovations occur rapidly, can be made without a substantial capital investment, and tend to be creative combinations of previously-known techniques.

Even if patent law were appropriate for protection of software, due to the large volume of recently-granted software patents and the rising number of new applications, the current patent process would continue to be troublesome for the software industry. Software patent examinations are hindered by the limited capability of searching prior art, by the turnover rate among examiners in the Patent and Trademark Office, and by the confusion surrounding novelty and innovation in the software arena. The problem is exacerbated by varying international patent laws, which both raise the cost and confuse the issue of patent protection.

Unfortunately, as a defensive strategy, Oracle has been forced to protect itself by selectively applying for patents which will present the best opportunities for cross-licensing between Oracle and other companies who may allege patent infringement.

COMPUTER SOFTWARE POLICY ISSUES

The policy rationale for patent protection in many industries is understandable. In exchange for making an invention available to the public, inventors are rewarded with a seventeen-year monopoly giving them exclusive right to the new technology. In such cases, this opportunity to monopolize the commercial application of the invention is justified as an appropriate reward given the capital resources dedicated by the inventor to the invention, including time and money spent in innovation, production, distribution, etc.

This policy, however, does not fit well with the software industry. Unlike many manufacturing-intensive industries, the development of software requires a minimum of capital investment. Producing and distributing a product is simpler, faster, and less expensive in the software industry than in manufacturing sectors. New developments influential to the software industry frequently emanate from individuals and small companies that lack substantial resources.

Software varies from manufacturing in another key aspect. The engineering and mechanical inventions for which patent protection was devised are often characterized by large "building block" inventions that can revolutionize a given mechanical process. Software, especially a complex program, seldom includes substantial leaps in technology, but rather consists of adept combinations of many ideas. Whether a software program is a good one does not generally depend as much on the newness of a specific technique, but instead depends on the unique combination of known algorithms and methods. Patents should not protect such methods of innovation.

The U.S. software industry has evolved to a multi-billion dollar industry that leads the world in productivity, and accounts for substantial portion of U.S. GNP. The software industry has advanced the efficiency of other industries through the proliferation of computing and computer-controlled processes. All of these gains have come prior to the application of the patent process to software, and consequently without patent protection for software. There is no justification for a policy that would not only drain capital resources (which are better spent on software development) into patent applications and other legal fees, and also actually serve to reduce innovation by limiting the availability of previously-developed techniques.

In sixteen years, Oracle Corporation has grown from a start-up company with a handful of employees to the world's third-largest independent software producer employing 8,000 people. Oracle filed its first patent application in November 1991, not because it felt that its software was suddenly worthy of patent protection; it filed that application because of concerns that other inventors, afforded patent protection by a flawed patent system, might find themselves in a position to seriously weaken the Company's competitive edge by alleging patent infringement. Even if Oracle had developed a certain invention first and could produce the appropriate prior art to prove its case, thousands of dollars in attorneys fees and other expenses would be spent in defense of its rightfully-owned technology. Oracle consequently believes that it must have a patent portfolio with which to respond to potential aggressors, so as to settle with them by cross-licensing to avoid litigation. Oracle is forced to channel a significant portion of its financial resources into patent protection of its assets, rather than using those resources in further innovating and expanding its computer software products.

Copyright protection for computer software is sufficient to preserve the rights of software developers, who rely on the unique combination of algorithms and techniques to produce successful software programs. Copyright law, including relief from those who copy or distribute copyrighted works without permission, in combination with careful handling taken to preserve trade secrets, has afforded adequate protection to software developers against the losses they may encounter from the wrongful use of their software. Compared to adequate copyright and trade secret protections, patent protection is excessively broad and enormously expensive.

CHANGING THE PATENT SYSTEM

Oracle has recommended that patent protection not be provided for computer software or computer software algorithms, for the reasons described above.

If software continues to be protected by patent law, however, we recommend the changes described in the following paragraphs. These recommendations in no way endorse the use of patents for protecting software, but rather serve to assuage the existing problems if patents must ultimately affect software development.

Patent law should be consistent throughout the world and, if it is to be applicable to software, should extend for much shorter periods of protection than exist now, unified prior art searching capabilities, equal standards of novelty, the elimination of patent rules that allow “patent flooding,” and identical standards for prior use restrictions (bar dates).

The evolution of software moves very quickly. The term of software protection should be cut back accordingly, from the current 17 years from grant date to three years from application date (the application period must be drastically reduced). A balance of fifty years protection for direct copying of code would continue to be provided by copyright law.

If the patent system is to remain an entrenched part of the software industry, then the following changes need to be made:

- The prior art capabilities of PTO records must be vastly improved to confirm effectively the novelty and non-obviousness of software patent that is the subject of applications. New classifications, as well as an effort to record the current state of prior art would be necessary.
- Because of the unusual speed with which software innovations are incorporated into products, the PTO’s patent review process must be made much more efficient so that it takes no more than six months from application to registration. In the software industry, if a patent application takes two years to process, the patented “invention” is often either widely used or obsolete by the time the registration is issued.
- Examiners skilled in computer science and software programming must be trained on the nature of software inventions, and the state of existing art. Qualified examiners must be hired and retained by the PTO at much higher rates than they are today. Compensation rates equal to those provided by industry are essential to recruit qualified personnel.
- The PTO, in conjunction with industry, must establish additional committees to clearly delineate the standards of novelty and non-obviousness that will be required for software inventions to receive patents.

Appendix G: Mitch Kapor’s Congressional Testimony

The following is an extract from the testimony of Mitch Kapor, founder of Lotus, to a congressional hearing:

I want to thank the Committee for this opportunity to testify on some of the intellectual property questions surrounding software. This is an area that I personally find fascinating and provocative - so please excuse me if my testimony ends up leaving you with more questions than answers.

With no joke intended, software has been very, very good to me. I was fortunate enough to find a collaborator to craft an innovative piece of software called Lotus 1-2-3 - and that software evolved into both an industry standard and turned Lotus Development Corp. into one of America’s most successful software companies.

Because it is impossible to know what patent applications are in the application pipeline, it is entirely possible, even likely, to develop software which incorporates features that are the subject of another firm’s patent application. Thus, there is no avoiding the risk of inadvertently finding oneself being accused of a patent infringement simply because no information was publicly available at the time which could have offered guidance of what to avoid. Please, require publication of patent application within a short period of their filing.

The period of patent protection, 17 years, no longer makes sense in an era when an entire generation of technology passes within a few years. My recommendation would be to consider substantially shortening the length of protection.

Most importantly, it is my heartfelt belief that many of the increasing number of recently issued software patents, concerning, for instance, fundamental techniques and artifacts of user interfaces, should never have been granted in the first place because of their failure to qualify as either novel or non-obvious. Some patents appear to preempt automation of common functions such as footnoting. This to me is like allowing a patent on the round steering wheel. The breadth of claims being allowed in these matters, is, in the words of Brian Kahin, Adjunct Research Fellow at Harvard's Kennedy School of Government, "often at a level of abstraction that is shocking to the uninitiated."

If some future litigant is successful in upholding rights to one of these "bad" patents It will require expensive and time-consuming litigation, whose outcome is frankly uncertain, to defend the rights of creators which should never have been challenged in the first place. If I speak very bluntly here, it is only because I am deeply concerned that a single bad patent court fight with a negative outcome, like a major environmental accident, could have catastrophic effects. I don't think we can afford the risk.

Appendix H: About the League for Programming Freedom

Ten years ago, it was possible to develop software using any techniques known, and providing whatever features were useful. This is no longer the case. New legal precedents have resulted in software patents and interface copyrights both of which now prevent this.

"Look and feel" lawsuits attempt to monopolize well-known command languages; some have succeeded. Copyrights on command languages enforce gratuitous incompatibility, close opportunities for competition, and stifle incremental improvements.

Software patents are even more dangerous; they make every design decision in the development of a program carry a risk of a lawsuit, with draconian pretrial seizure. It is difficult and expensive to find out whether the techniques you consider using are patented; it is impossible to find out whether they will be patented in the future.

The League for Programming Freedom is an organization of over 600 software developers, business people, professors, students, and users dedicated to restoring the preexisting freedom to develop software. The League is not opposed to the legal system that congress intended --- copyright on individual programs. Its aim is to reverse recent changes that run contrary to the public interest principles of the Constitution.

The League works to abolish these new monopolies by publishing articles, talking with public officials, working with companies, assisting in court cases, and serving as a point of contact.

The League may be contacted by phone: (617) 621-7084, by electronic mail: lpf@uunet.uu.net, or by post:

League for Programming Freedom
1 Kendall Square #143
P.O. Box 9171
Cambridge, MA 02139

The League is always happy to assist those threatened or concerned about the above issues, or to simply provide further information on these matters to any party that may be interested.

Appendix I: About this Document

Version 1.3 of January 25, 1994.

Assistance: James Salsman, Lile Elam, and Paul Rubin.
This document represents the opinions of the LPF.

Gordon Irlam
League for Programming Freedom
(415) 336-5889
gordoni@netcom.com

Ross N. Williams
Rocksoft^tm
+61 8 379-9217
ross@guest.adelaide.edu.au

Gordon Irlam is a software developer at Sun Microsystems. He is in the Systems Architecture and Performance group where he is currently developing a microprocessor simulator to gather data needed to help design future hardware. He is a member of the American Economic Association, has a personal interest in information economics, and has been observing software patent developments for over four years.

Dr Ross N. Williams is an independent consultant and software developer, specializing in the areas of data integrity and data compression. In addition to developing a data integrity product, Dr Williams has been working as a consultant to Microsoft on the Stac vs Microsoft software patent lawsuit.