

The Patent Lawyer

The Magazine of Global Patent Law Issues

EITHER-OR NO MORE

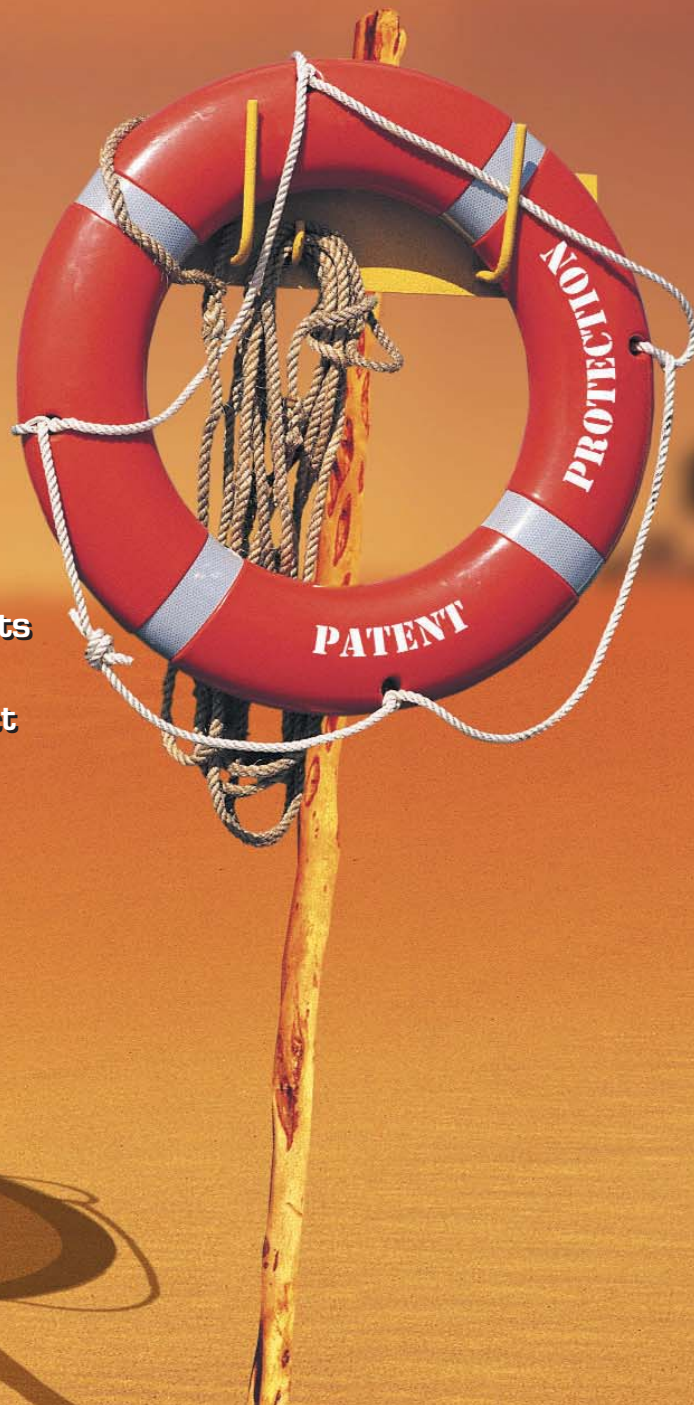
Using Copyright and Trade Secret Law
to Protect Patent Applications

PLUS...

Cutting the Cost of Software Litigation

An Integrated Approach to Software Patents

When Trade Dress, Utility Patents Intersect





An Integrated Approach to Software Patents

Unique technology calls for a unique approach to prosecuting and litigating

By Timothy R. Baumann

Historically, courts were reluctant to recognize software as being worthy of patent protection. A principal argument against the patentability of software technology was that software was merely a mathematical algorithm, which by statute cannot be patented.

In 1981, the Supreme Court rejected this argument and recognized the legitimacy of software-related patents. *Diamond v. Diehr*, 450 U.S. 175 (1981). Subsequent Federal Circuit decisions have also recognized the legitimacy of these software patents. See, e.g., *In re Alappat*, 33 F.3d 1526 (Fed. Cir. 1994).

For any type of patent, the costs associated with prosecuting and litigating are not minor and have increased in recent years. Software prosecution and litigation, because of the type of technology involved, often raises its own issues.

These unique concerns can dramatically increase the costs associated with the prosecution and enforcement of software patents if not properly addressed. What is needed is an integrated approach to obtaining and enforcing software patents to gain positive results in the most cost effective manner possible.

Anticipate Litigation

The successful enforcement of a software patent begins with drafting the application, and successful drafting should anticipate the possibility of litigation down the road. Drafting the application should follow

practices common to patents in other technical areas. For instance, claims of having broad coverage and narrower coverage should be included. In another example, the specification should adequately support the claimed subject matter.

Whoever “makes, uses, offers to sell, or sells” a patented invention, directly infringes the patent.

Software technology, however, is different than other more traditional types of technology. Software programs are usually organized into modules or components that may be operated by different parties. For example, many software systems, particularly in data networking applications, involve the interaction of software applications among many actors. These actors are often in separate geographic locations.

Software is also flexible technology that is easily modified or changed in contrast to “older” technologies. The flexible nature of software encourages upgrades or changes over time. The changes frequently are made through different releases of the product and may quickly make earlier versions obsolete.

The community of software developers is different than many other technical areas. Rival developers communicate with each other and developers usually know the technical development of rival organizations. Software conferences are common and the number of publications and presentations made at these events is large. Many software applications implement industry standards and standards organizations, both of which have mushroomed over the past few years, particularly in the telecommunication industry.

Because of the nature of software technology and the software community, a patent drafter should consider certain issues when drafting software patent applications. By considering these options when drafting the application, time, money and frustration may be saved when the time comes to enforce the patent.

Consider Direct, Indirect Infringement

Whoever “makes, uses, offers to sell, or sells” a patented invention, directly infringes the patent. 35 U.S.C. § 271(a). On the other hand, whoever “actively induces” infringement of the patented invention or “contributes” to the infringement of the patent of the patented invention, is liable for indirect infringement of the patent. 35 U.S.C. §§ 271(b) and (c).

As noted above, in software systems, different elements are often linked together to form a complete working system. For example, different computers running separate software modules are often con-

In Practice *continued*

nected over computer networks and, frequently, these different components are owned or operated by different entities. An area of concern in software patent applications is that if the claims of the software application are drafted too broadly, it may be possible that the issued claims are only infringed indirectly only, by multiple parties.

Although indirect infringement results in the same liability as direct infringement, it is more difficult and costly to prove. In attempting to prove indirect infringement, a plaintiff will likely have to take substantial additional discovery from additional parties in order to prove their infringement claim.

To avoid these obstacles, claim sets having different scopes and focuses should be drafted. One claim set might be directed to one component of a system. Another claim set might be directed to all components of the system. If a single component cannot be



claimed, a minimum set of components should be selected for claiming. Thus, a single infringer might be implicated by one claim set and the job of enforcement is made a lot easier and cheaper.

Consider Changing Technology

When filed, the patent is by definition written in terms of the technology available at the time. However, after filing, the technology often changes. In one example, a patent covering software in a telecommunication system may be written in terms of a certain communication protocol or standard being used at the time of filing. Later, however, the protocol may become outdated and be

replaced by a new protocol. How can one cover or incorporate later-developed technology in the patent and have the later developed implementation still covered by the claims?

Properly drafted, claims are not limited to certain protocols or terms. Broad claims should be written to be as implementation-independent as possible. Using generic terminology avoids changing meanings and focuses on the invention. For example, using a telecommunication example, one might wish to claim “data unit” rather than “packets” or “frames,” since the later terms have specific meanings within the telecommunication field. In other words, a drafter should look to the function of an element rather than a technology-specific term when drafting claims.

The specification should also stress the elastic nature of the software technology. In

this regard, the text should describe examples rather than absolute approaches. Specifically, unless required for patentability, the invention should not be limited to certain formats, protocols, or architectures.

Document Key Dates

Software technology changes rapidly. What is considered new and inventive at one point in time may become quickly outdated. Moreover, engineers at many organizations are often working to solve the same problem. The result is that, more often than litigators would care to admit, similar solutions are often arrived at in roughly the same time by different people.

Because of these considerations, documenting the conception and reduction-to-practice dates become critical. Unfortunately, this area is often overlooked, particularly by small companies. In litigation, defendants often produce evidence of the prior art that predates the patent. Without written evidence, it becomes difficult to predate and overcome this alleged prior art. Obviously, this adds to the cost and expense of litigation.

To alleviate this issue, a formal procedure should be established to document conception and reduction to practice information. Lab notebooks are one example of an item that might be maintained. Code listings of the actual software should be stored and dated. By taking easy steps during development, the job of litigation becomes easier and less costly.

Carefully Consider Prior Art

The software industry is prolific in terms of the amount of documents produced. For instance, large numbers of conferences, trade shows and public product demonstrations occur at a great frequency. Slides, brochures and even sample programs are distributed at these gatherings.

A plaintiff does not want to be in a position where a piece of prior art is in the client's files (especially an inventor's file) but not disclosed to the U.S. Patent and Trademark

Office. Obviously, if such a situation develops, the expenses involved in litigation become a lot greater. An attorney must ask and re-ask the client and inventor what they have in their files to avoid these complex and potentially claim-killing problems.

Consider Continuation Applications

A continuation application is often used to obtain additional claims that have not been made in a parent application. The prosecution of the continuation application continues, and claims from the continuation will issue after the parent application has issued. Often, during a lawsuit, prosecution of a continuation application is transpiring during the litigation against an infringer.

The nature of software technology makes it more difficult to determine the extent of infringement compared with other technical areas.

In this situation, the very presence of a continuation has an impact on the lawsuit. The infringer cannot be certain claims will issue. Even if the infringer has viable defenses from the claims of the parent, these defenses may not apply to the continuation claims. As such, a plaintiff has valuable leverage in settlement discussions with the infringer. The continuation application must be filed during the pendency of the parent application. If technology or the direction of an industry changes and/or better ideas concerning claim scope become apparent, it is wise to seriously consider some sort of continuation application.

Pursuing Infringers

The nature of software technology makes it more difficult to determine the extent of infringement compared with other technical areas. For plaintiffs, it is important to realize this aspect of software technology, and evaluate infringement with these issues in mind so as to avoid problems later in litigation.

As noted above, the nature of software is different and more difficult to decipher than other technologies. Software components are often organized into modules of source code. The architecture, flow and other details of the source code are usually not visible or accessible to the public. How the individual modules communicate with each other is also difficult to determine. The claims of software patents often implicate these very sort of hidden details.

In contrast, patents involving other technology make it easier to determine infringement. For example, with patents that cover mechanical devices, one can purchase the product and examine it. You can take it apart and investigate its internals, if they are at issue.

How does one investigate software technology? One place to start is the Web. Web sites may have white papers or other similar documents written by engineers or scientists that describe in detail the inner workings of the software.

Another source of information may be articles or conference papers written by employees of the alleged infringer. These individuals may present details of their design at these meetings. Technical reviews of products in the technical press may also prove useful.

No matter the source of the information, a careful and realistic determination must be made by the patent holder and its counsel to avoid potential problems during litigation. Rule 11 sanctions are always a possibility if the plaintiff is dead wrong about the infringement allegations. Obviously, a defendant that unleashes Rule

In Practice *continued*

11 sanctions against a patent holder greatly increases litigation costs for the patentee. The bottom line is that careful and extensive research is required before a lawsuit is filed.

Modify Litigation Schedules

Traditional case schedules begin discovery with document production and other forms of paper discovery. Depositions often come toward the middle or end of the discovery period. As stated above, one of the difficulties in establishing infringement allegations for software technology is the lack of visibility of key technical details of the software. Paper documents often omit or gloss over these very details. If nothing special is done, months or years could pass with no perceptible advancement of the case.

In this situation, it often makes sense for a plaintiff to try to persuade the court to allow an early and aggressive discovery schedule. For example, the court could require a

statement of how the accused product works to narrow the issues involved. Also, Rule 30(b)(6) depositions relating to the workings of the accused products could be moved to the top of the schedule in order to discover the details of product operation.

This latter approach is especially effective in confirming accused product details since it is less apt to be successfully massaged by opposing counsel as, for example, interrogatory answers, may be. The general goal for any discovery acceleration approach is to “cut to the chase” and avoid costly, dragged out discovery. By doing so, litigation costs can be reduced and the case can be advanced in a meaningful way.

An important goal in patent prosecution is to obtain a patent that can be enforced. A corollary to this goal is to have property



that can be enforced in a cost-effective manner. With software patents, the nature of the technology necessitates taking steps both during and after prosecution that ensure that the software patent is strong and enforceable. By taking all or some of the steps above, software patents can be enforced successfully and in a more cost-effective manner. ■

Timothy R. Baumann is an associate in the Chicago office of the intellectual-property and technology-law firm Fitch, Even, Tabin & Flannery.



CUTTING THE COST OF SOFTWARE LITIGATION

Strategies for minimizing expense while maximizing success

By Timothy R. Baumann

The number of software-related patents has exploded in recent years. According to one study, 130,000 software patents were issued between 1976 and 1999. With the proliferation of software patents, the number of corresponding infringement lawsuits has also increased.

Software patent enforcement actions present unique challenges and characteristics that often make such litigation more expensive than other infringement cases. For instance, during pre-lawsuit investigations, software code listings – the heart of proving most infringement cases – are usually not available to potential plaintiffs, making the decision to initiate a lawsuit more difficult and increasing the possibility of launching a weak (and hence more expensive) case.

Even after the lawsuit begins, critical operational details of how the software operates (including the source code) are often the last items to be discovered, thereby prolonging discovery and increasing costs to the plaintiff. Furthermore, due to the nature of software technology, many software patent claims often involve multiple infringers, thereby necessitating multiple sets of discovery and further increasing costs for plaintiffs.

This article discusses strategies designed to minimize these obstacles and thereby reduce discovery costs by obtaining the most complete information at the earliest possible time. Simply put, the more time drags on, the more money is spent in discovery. Obtaining needed information as early as possible in the litigation ultimately reduces costs and leads to more positive results.

Modify the Discovery Schedule

Traditional case schedules begin fact discovery with document requests, interrogatories and other forms of “paper” discovery. Depositions often come toward the middle or end of the fact discovery period. One of the difficulties in proving infringement allegations involving software technology is the lack of visibility of key technical details of the software. Paper documents often omit or gloss over these critical details. If nothing special or different is done to alter traditional discovery schedules, months or years could pass with no perceptible advancement of the case.

In software cases, it often makes sense for a plaintiff to try to persuade the court to allow an early and aggressive discovery schedule that focuses primarily on obtaining the critical software operational details. For exam-

ple, the court could require a statement of how the accused product works be produced by the defendant so as to narrow the issues involved.

In another example, “special” Rule 30(b)(6) depositions relating to the workings of the accused products may be moved to the front end of the case schedule in order to discover the details of product operation early in the case. Early depositions are especially useful since they confirm (or illuminate) accused product details, and the testimony obtained is less apt to be successfully “massaged” by opposing counsel as, for example, interrogatory answers, may be.

By avoiding costly and dragged-out discovery, litigation costs can be reduced and the case can be advanced in a meaningful way. Procedurally, the request for an accelerated schedule can first be brought to the court’s attention at status hearings. The early service of deposition notices may also be an avenue to alter the traditional schedule. In some cases, both parties may be able to agree to an altered schedule to promote advancement of the case and early settlement.

Some district judges, such as some serving the Northern District of Illinois, have implemented this approach in previous

Strategic Counsel *continued*

cases and may be more receptive to altering traditional discovery schedules than judges with less patent experience. It is up to the plaintiff to educate the court as to the virtues of a revised schedule. In the end, requesting an accelerated schedule costs little in terms of money or the court's good will and may significantly reduce costs.

Obtain Source Code Early

The source code that operates an accused product is often the heart of proving a software infringement case. However, before the code can be produced, other issues often need to be settled.

For example, the early production of source code will almost certainly require that the court enter a protective order early in the case, because defendants are unlikely to produce any source code without significant protections (e.g. attorneys' eyes-only level of protection). The longer protective order issues drag out, the longer it will take to obtain the source code and the more expensive discovery becomes. Thus, a protective order needs to be advanced early in the case, not months into fact discovery.

In addition, counsel should consider the format of how the code is produced. There are several formats that are not electronically searchable (e.g., PDF files) while other formats (e.g., text files) are electronically searchable. This issue becomes important because, in source code analysis, the focus falls on particular modules, variables and routines. If tens of thousands of lines of source code need to be examined, having unsearchable format becomes both daunting and, for the client, expensive. If sections of the code needed to prove infringement cannot easily be found, much time is wasted trying to piece together how the code works. Make it clear to opposing counsel from the onset of discovery that it needs to produce the code in a searchable format.

Narrow the Number of Players

Software processes often require several actors to perform. For example, a software process where a transmitter sends information to a receiver via a transportation medium may implicate several actors. Specifically, one defendant may operate the transmitter, a second defendant the transportation medium, and a third defendant the receiver.

The number of defendants becomes important because each new defendant adds another party to the case, another set of discovery, and, therefore, more costs. Moreover, potential defendants may, in fact, be customers or business allies of the plaintiff, making it impracticable to add the party to the case. For these reasons, a plaintiff should carefully consider asserting patent claims that implicate as few defendants as possible, if costs are a major concern.

Selecting claims requires careful analysis at the inception of the case and must be based on something beyond simple guesswork. Knowledge of the technology becomes important. Here, the client can be of assistance.

The client and outside counsel should break the software process into portions and determine the identity of potential infringers of these parts. Sometimes it will become apparent that all claims of the to-be-asserted patent require multiple actors in order to be infringed. Involving multiple parties is acceptable if, at the beginning of the case, both outside counsel and client accept the need for additional parties as well as higher costs. Waiting until later in the case, at the conclusion of fact discovery, to decide that additional parties are required leads to complications and significantly increased costs.

Obtain Early Technical Help

Obtaining adequate technical assistance is important in software cases, especially in parsing through the documents such as the source code. Most attorneys will not be technical masters in the field, and, especially for small- and medium-sized clients, it becomes prohibitively expensive to train attorneys in the technology.

Beyond attorneys, there are only a few options available for obtaining the needed assistance. The client itself probably has employees who are steeped in the relevant technology. However, in most cases, protective orders will prohibit employees from reviewing the defendant's sensitive discovery, such as source code listings.

Another option is to hire an outside expert. The outside expert, if cleared under the protective order, should be able to see the discovery. An effective expert must be obtained and it is here that the client can be of help. The client likely knows individuals in the field who are respected and, one would hope, charge modest rates. These experts typically have worked in the industry for a long period of time and charge lower rates than "academic" experts.

To reduce costs, the use of an expert needs to be carefully monitored and managed. Experts should not have open-ended assignments, but should be focused on specific tasks. One example is to have an expert focus on analysis of the source code and portions of the code that implicate the claim limitations.

Obviously, using an expert costs money. But effectively managed, focused experts

can aid in all phases of discovery, such as the review of the source code and the formulation of deposition questions. Although experts are a significant expense, they can end up saving money by helping to obtain needed information at the earliest possible time.

Reduce Production Costs

Software litigation produces reams of paper (and non-paper) discovery. This discovery needs to be stored so that it can be reproduced easily. Copying tens of thousands of pages of documents (often on multiple occasions), indexing and other processing steps add significantly to the client's costs.

My experience has been that using front-loaded discovery scheduling can cut overall discovery time by months, thereby saving tens of thousands of dollars.

Although document production services may disagree, technology has advanced to the point where law firms can often process documents in-house, without employing expensive imaging and indexing services. Documents produced by opposing parties can be scanned and stored on a computer system using inexpensive scanners. Once stored in electronic form, the documents can easily be sent to experts or placed on a laptop for work out of the office.

Manual indexing can also be done upon initial attorney or expert review of the discovery. For instance, an outside expert familiar with source code can produce a "quick" index, often in a few hours. This may negate the need to use expensive scanning and indexing services when client costs are a critical concern.

As for documents to be produced to opposing parties, these can easily be produced on CD-ROM or in some other electronic

format. This saves money at several points in the discovery process.

For instance, during the initial gathering of documents for production to the defendant, the client can quickly put potential documents to be produced on a disc and thereby save the time and expense of having outside attorneys physically search for the documents at the client's facilities. The documents can be reviewed by outside counsel for potential production and be produced to the other side in an electronic format (e.g., CD-ROM) with one production number. Keeping documents in electronic form from the beginning of the case saves money.



Conclusion

These suggestions often require spending more money up front than is typically done in infringement actions. However, experience indicates that a loaded and focused front-end schedule ultimately reduces costs in the case. Settlement options may also be advanced because the relative strengths and weaknesses of the parties are highlighted sooner than would normally occur.

My experience has been that using front-loaded discovery scheduling can cut overall discovery time by months, thereby saving tens of thousands of dollars. Performing in-house indexing and using electronic formats for production saves even more resources. Finally, limiting claim assertions to single parties saves still more time and money.

The nature of software technology requires some tweaking of traditional litigation approaches. By taking all or some of these steps, counsel can enforce software patents successfully and in a more cost effective manner. ■

Timothy R. Baumann is an associate in the Chicago office of the intellectual-property and technology-law firm Fitch, Even, Tabin & Flannery.