Practice Tip

In virtually any case involving computer software, one of the discovery requests will be for the source code for the relevant computer software. But awareness before the discovery request of how the source code will actually be used can result in a more focused discovery request, says author Lee Hollaar, a professor of computer science who has been a technical expert in a number of intellectual property and antitrust suits. Knowing how the source code will be used will give a better chance that what is needed will be produced in a timely fashion, and be a more efficient use of the software experts.

Based on his experiences as a technical expert in computer software copyright, trade secret, and patent suits, and a number of the Microsoft antitrust suits (where he examined Microsoft source code from MS-DOS 1 through Windows XP), Hollaar gives practitioners advice on how to make their examination of source code more effective.

Requesting and Examining Computer Source Code

By Lee A. Hollaar

How Source Code Is Used

Before making a discovery request for computer source code, an attorney should consult with the case's technical experts to decide what to request, and in what form. Your discovery request will turn on what claims are at issue: In different kinds of cases you will have different specific purposes for examining source code. In a patent infringement case, for example, you are looking for the particular sections of source code that implements each claim element. There is no need to compare it to the patent owner's implementation of the invention.

In contrast, in a copyright infringement case, you are looking for substantial similarity between the copyright owner's source code and the alleged infringing software, after filtering out aspects of the source code that are not protectable by copyright, such as things dictated by external considerations or efficiency. However, this is not a simple literal comparison, since there still may be copyright infringement if the alleged infringing work is a derivative work written in a different programming language.

Trade secret misappropriations are similar to patent infringements, in that you are looking for portions of the source code that implement the trade secret. But trade secret suits are more complicated than patent suits, since there are not the claims of a patent to guide the examination of the source code. Rather, the plaintiff in the trade secret case must specify the particular trade secrets at some point in the litigation, preferably (at least for the defendant) before access to the source code so that it cannot be mined for possible trade secrets.

The need for source code in antitrust litigation depends on the nature of the claims. In *Caldera v. Microsoft*, I used the source code to determine whether the tie between Windows and DOS in Windows 95 was necessary to produce the benefits that Microsoft had claimed. It was also used to determine the nature of predatory code that Microsoft had inserted in a beta test version of Windows 3.1 that printed out an error message when run with DR DOS, and the nature of a problem that prevented the Windows 3.1 setup program to

Dr. Lee A. Hollaar is a professor in the School of Computing at the University of Utah in Salt Lake City, where he teaches networking and computer law. He is the author of Legal Protection of Digital Information, published in 2002 by BNA Books and available at no cost at http://digital-law-online.info. He was a committee fellow at the Senate Judiciary Committee, where he worked on patent reform legislation and what became the DMCA, and was a visiting scholar at the Federal Circuit. He has been the technical expert in a number of intellectual property and antitrust cases, and assisted the states in their settlement negotiations and the remedies phase of the Microsoft case. Professor Hollaar is a member of the Advisory Board of Expert Evidence Report, and can be reached at hollaar@cs.utah.edu.

  EXER ISSN 1536-1896

run with DR DOS. In contrast, in *Bristol v. Microsoft*, I used the source code to determine the portions of Windows NT4 that would not be supplied to Bristol under a change in its contract with Microsoft and the effect that would have on the viability of Bristol's product allowing the migration of programs from Windows to Unix.

What Needs to Be Provided

What source code needs to be provided depends on more than the type of action. While it may be possible to request only selected portions of the overall source code for a system, such as those that may correspond to the elements of a patent claim or that implement a particular capability, that requires a great deal of trust in the producing side. It is far better to request the complete source code, which in most cases can be stored on one or two compact discs.

In fact, it may be necessary to examine a number of versions of the source code if there is a claim by the defense that the software has been changed over time, such that it might not have infringed before or after a particular version. In that case, the plaintiff's expert would want to compare the source code of the different versions to determine how it has changed and whether such changes affect the possible infringement of copyright or patent. This comparison often can be done using a utility program such as Microsoft's WinDiff, which can compare all the files in two different directory trees and indicate which ones have changed. WinDiff also allows the examination of the changes to the individual files, highlighting added, removed, or changed lines in color.

It is clearly important to ask the other side, either in interrogatories or in a deposition, whether there are differences in the versions of the source code for a program that affect whether it may infringe a copyright or patent, and what those differences are.

Sometimes, the source code is stored in a source code control system (sometimes called a version control system) where a programmer can check out a file to work on and then check it in when finished. Many source code control systems can show the state of a file at any specified version, and provide a history of the changes. Because the source code control system may store interesting information not contained in the individual source code files, such as who made a particular change, when the change was made, and even a comment about why a change was made, a discovery request for source code should always specify that if a source code control system is used, all the information from the source code control system (and in particular, any control or database files used by the source code control system) must be produced.

But unless one has a running copy of the source code control system, having its information may not be enough. Perhaps the best way to handle this is to ask in an initial interrogatory whether a source code control system is used in the maintenance of the source code and, if so, what system is being used, including the name of the supplier and the particular version. If that source code control system is readily available, then requesting a copy of its stored files should be sufficient. However, if a "home-brewed" source code control system, or one not generally available, is being used, then it will be necessary to request the source code control system along with the source code, with information necessary to configure and use the source code control system.

It is sometimes desirable to compile the source code, rather than simply examine it. This can assure that the production is complete and corresponds to a particular marketed version, and allows the expert to examine its execution using a debugging tool.

It is sometimes desirable to compile the source code, rather than simply examine it. This can assure that the production is complete and corresponds to a particular marketed version, and allows the expert to examine its execution using a debugging tool. If you plan on compiling the source code, information about the particular compilation tools and their configuration should be requested in an interrogatory. As with a source code control system, if the particular tools are not generally available and are necessary for the examination of the program, they should be requested.

Sometimes the software to be examined is not just conventional source code, but has been developed using something like Microsoft's Visual Basic. The software contains not only the sequence of instructions familiar in conventional software, but also information that configures the runtime environment to create forms or reports based on predefined controls. To see the effect of that information, it is necessary to view the software in the same development environment as it was developed. Again, that environment should be determined by interrogatories before the source code production request, and if it is not something that can be easily recreated (say, because it uses tools that were developed in-house or a particular software development system not readily available), a request should be made for the development tools in addition to the source code.

Material to be produced can also include more than source code. If a database system, like Microsoft's SQL Server, is being used as the foundation of a system, then the database schema and any other information used to configure the database system may have to be reviewed.

Finally, if the program uses any libraries, database systems, or similar things supplied by others, it will be necessary to get those, and information regarding their configuration and use, if the program is going to be compiled and run with a debugger.

Based on the response to well-crafted interrogatories, or the deposition of a software developer familiar with the tools and procedures used, it should be possible to frame a discovery request that gets all the necessary material without being burdensome.

How It Should Be Provided

Obviously, you want the source code in a form that can be best used in the particular type of action. In no

case will this be a paper listing of the source code, because it is simply too hard to examine. For example, it does not permit the expert to search for the point where a particular message is generated (which is a great way to find a section of code that performs the operation related to the message), all uses of a particular variable or subroutine in the program, or similar things. A machine-readable version is necessary.

And in some instances, a paper listing of the source code would be too large to reasonably handle. In the *Bristol* case, the source code for Windows NT4 required four compact discs, and for Windows 2000 required seven compact discs. Microsoft initially supplied about two dozen compact discs of source code, because of different versions of Windows, and about 40 compact discs by the end of discovery. The source code for even a simple system may fill more than a compact disc if different versions must be examined.

The side producing the source code will be justifiably concerned with its protection, and will likely insist on special conditions regarding its use and handling. Many of these concerns may already be addressed by the protective order in the case for other highly-confidential information. If that is so, then the framework of the protective order should be used, rather than providing new conditions, simply so that mistakes are not made because of special rules. The treatment of source code needs to be considered in the development of any protective order in a software-related case.

Care must be taken to protect the source code. It should never be stored on a machine where it can be accessed by unauthorized persons. Because peer-to-peer file sharing program or computer viruses may give unintended access to others on the Internet, the computer should never be connected to a computer network when storing the source code. That can be achieved by using a special computer for the source code (since computers are so inexpensive today) or by storing the source code on removable media (Zip disks or external hard drives, for example) and only connecting the computer to a network when the source code has been removed.

If source code control systems, database systems, or particular development systems are necessary for the proper examination and understanding of the source code, it may be easier to request that the party producing the source code provide it on a portable personal computer, with all necessary database systems and development tools properly configured. The cost of such a portable computer is small compared to the time that the requesting party may spend getting the development environment running, and the producing party answering questions.

In a software trade secret case where I was a court-appointed expert, one of the potential trade secrets was the database organization. Each party set up a portable computer with their database systems running (as well as with their source code and its development tools) and sent it to me. That worked really well and substantially reduced time I had to spend and the cost to the parties.

What to Do After You Have It?

Of course, what you do with the source code after you have it depends on what you wish to prove in your case. For a copyright or trade secret case, your expert will compare the defendant's source code against the plaintiff's to determine if they are substantially similar or contain the trade secret elements. For a patent case, your expert will look for portions of the source code that implement the specific claim elements.

But if the source-code-produced-runs fill a compact disc or more, how do you go about finding the needles in a haystack of code? In many cases, the expert can use software tools to do much of the work.

But if the source-code-produced-runs fill a compact disc or more, how do you go about finding the needles in a haystack of code? In many cases, the expert can use software tools to do much of the work. For example, you can produce a list of files, database tables, or similar items from two systems, sort the lists, and compare them to see what items are in one system but not the other. You can search for a particular subroutine name to find all the other routines that use it. You can look for the character string of a particular message that occurs when the program is doing an operation of interest, and then search for all locations that output that string.

In patent cases, you examine the source code as added confirmation that the program infringes the patent. For example, in a case regarding patents on a method for repartitioning a hard disk of a computer, I could determine that the claims at issue were infringed by creating a test configuration on the disk, running the program, and seeing the result. Unless the changes were being done by magic, the defendant's program infringed the patents. But it was useful to indicate the sections of the source code where the various steps were performed, at least to the level of identifying the particular subroutines if not the specific lines of code, so that no argument could be made that the software didn't really perform the patented method.

One problem that comes up when you have only the source code to examine—rather than actually running the program with test data and observing the infringement—is that code that appears to implement one of the claim elements isn't actual run (so-called "dead code" that exists as an artifact of the code development), or is not run in the proper relationship to other claim elements. In that case, it would be necessary to actually run the code (and perhaps compile it with a debugging program) to show that it is actually executed as necessary to find infringement. Alternatively, the defendant could stipulate that anything in the produced source code will be executed when the obvious conditions are present. But the plaintiff in a patent infringement suit should either be able to actually compile and run the code in a debugging environment to show that it is actually executed, or get such a stipulation. Otherwise, they could be unfairly asked whether the code is actually executed.

Another reason for compiling and actually running the source code is to assure that the source code is complete so that it can be searched for the claim elements. But especially in a patent suit, where the expert is look-

  EXER ISSN 1536-1896

ing for particular sections of code that implement claim elements, there is a better alternative than having the plaintiff's expert searching through unfamiliar code. First, the defendant needs to make a representation that the source code production is complete and that for purposes of possible patent infringement, it is representative of all the versions of the allegedly-infringing program.

Second, the defendant can be asked, either in interrogatories or (preferably) a 30(b)(6) deposition, to identify all portions of the source code that perform particular operations corresponding to claim elements. That way, the code developer provides the initial road map to the code examination, and the source code can be used by the plaintiff's expert to confirm the answers and see whether anything was missed or misrepresented. (Knowing how easy it is to confirm the testimony tends to make it very accurate.)

By understanding what you intend to prove using source code, and working with your experts on the discovery request based on what they will need, an attorney in a case that involves computer programs can efficiently review source code, even when the system is as large as Microsoft Windows.