DisputeSoft
Software Dispute Experts

Source Code Protective Orders, From the Perspective of a Source Code Examiner

Andrew Schulman
SLC / DisputeSoft

This is the first in a group of articles on source code and how it is handled, examined, and analyzed in litigation. Separate articles will define the term "source code" for litigation purposes, put source code in the context of e-discovery (the court-mandated disclosure to the other side of a litigant's electronically stored information), and discuss source code discovery disputes, the purposes behind source code examinations (including the types of questions they can and can't answer), how a source code examination relates to other ways of looking at software (particularly reverse engineering), how information gleaned from a source code examination can be correlated with other evidence (such as the vendor's internal emails, or its public documentation), and some of the necessary skills of a source code examiner in litigation (which may differ from the qualifications of competent programmers generally, and which relate to the qualifications of a testifying expert under Daubert). The focus of these articles is fact gathering and analysis, and how these may be impacted by civil procedure rules and mechanisms, rather than on the legal rules themselves.

=====

Consider a software copyright case in which source code from the defendant (D) and from the plaintiff (P) must be compared against each other to look for literal as well as non-literal similarity. As part of the discovery phase in pretrial litigation, in which the two sides must disclose information to each other, each side's experts will generally get access to the other side's confidential source code. While perhaps surprising to those new to litigation (especially when they learn that each side generally must bear the costs of disclosing its "crown jewels" to its opponent), such access is clearly necessary in a case such as this, but access needs to be restricted to the purposes of the litigation. Restricted access might be controlled via a Non-Disclosure Agreement (NDA); but instead it is generally governed by a court-approved

Protective Order (PO) which outlines restrictions on source code access, in some cases requiring that the experts can only view the source code at limited times in a particular place.

Sometimes the two sides will negotiate, and agree to, a PO in which each side's source code must be kept on a *separate* standalone computer without an internet connection and without USB access; the two source code computers will likely be at law offices in two separate cities, and the PO will prohibit copying between the two computers. In a copyright case, comparing the two bodies of source code under these circumstances is challenging. That such a PO clause was agreed to in the first place may indicate that the attorneys viewed the PO in the abstract, without considering how it would impact a crucial part of the case's fact finding: the source code examination. (By the way, it is possible for source code examiners to do minimal comparison under these circumstances, using hand-written notes and a tokenization/sampling method, but that is a last resort).

Attorneys working in software-related litigation (such as patent, copyright, and trade secret cases) sometimes agree to PO restrictions on source code access, without thinking through the implications for how source code will actually be examined by experts and consultants. Failure to negotiate appropriate PO terms may later hamstring your expert’s ability to conduct the type of source code examination necessary and appropriate for the type of analysis that the expert must undertake, and further, may dramatically increase source code examination time and expense. Therefore, before agreeing to stipulating to terms of a PO that will govern source code examination, it is advisable to first consult with your expert to gain an understanding of the type of analysis he or she plans to undertake, and the type of access required for such analysis.

The producing party's law firm, which hosts its client's source code for inspection by the other side (the receiving or requesting party), may well understand the PO's impact on the source code exam, and may insist on certain PO restrictions to adversely impact the receiving party’s source code examination time and expense, rather than to protect any trade secret or confidential nature of the source code. Restrictive POs may be insisted upon as a tit-for-tat response to the other side’s “fishing expeditions” or discovery abuse, especially when the other side is a so-called “troll” for whom mutuality of discovery burdens doesn’t hold. (The term “discovery” as used

here includes mandatory disclosures under Local Patent Rules, discussed in an earlier patent-litigation article. The term "production" as used here refers to one side's court-enforced disclosure of its source code to the other side, not to the production or creation of software.)

This article urges attorneys, especially those who will *receive* source code productions, to "stop and think" (as the FRCP committee notes put it in a different context) how a PO will impact the source code exam. If possible, have an expert or non-testifying consultant review, up front (while the terms are still fluid), a proposed PO, rather than simply accepting "this is how it's been done before," and later presenting your expert with a *fait accompli* (which one wit has defined as "French for 'the train has left the station' ").

While the author happens to also be a lawyer, he is mainly a source code examiner, and this article is based on about twenty years of experience in source code examinations for patent, copyright, trade secret, antitrust, and privacy litigation.

This article addresses some prominent impacts that typical POs can have on source code examinations. Note that any descriptions here of particular methodology for source code examination are exemplary, and not intended to describe strict requirements. The views given here are the author's alone, and do not necessarily represent the views of DisputeSoft or its experts.

Typical source code PO restrictions, and their impacts on the source code examination

Typical PO restrictions on source code access are listed below; this list includes both reasonable and unreasonable restrictions. Intermingled with the restrictions are discussions of their implications for the exam:

- "Attorneys eyes only" (AEO), including outside experts; "Outside attorneys eyes only" (OAEO): Source code access generally limited to outside *non*-employees retained by the other side; no access to source code by the opposing party itself, possibly including in-

house counsel, and especially including its in-house engineers; and no sharing even of quotations from source code.

    - Thus, a party's own expert reports and claim charts, when based on the other side's source code, may need to be redacted, or not shared at all, with the party itself.

- Patent prosecution bar: no source code access for litigation attorneys also helping the other side acquire patents, at least in related subject-matter areas (see Unwired Planet v. Apple; and what has been called "Geotag v. The Known Universe").
- Source code provider receives resumes (CVs) of the other side's proposed source code examiners, and may object if the examiner could pose a competitive risk (see Symantec v. Acronis); the examiner may even be "tainted" by access to source code in earlier cases (see BIAX case cited below).
    - Reviewing the CVs of proposed source code examiners lets the producing party probe what would otherwise (under FRCP 26(b)(4)(D)) be unavailable information on the identity and background of those of the receiving party's *non-testifying* consulting experts who will be examining source code. (Of course, the producing party must know who is going to be accessing its source code. And, of course, it is entitled to information about the other side's *testifying* experts.)
- Source code only accessible on a "standalone" protected computer, without an internet connection. This is intended to prevent opposing experts and consultants from leaking a company's "crown jewels" to the internet.
    - While a properly-crafted non-disclosure agreement (NDA), signed by professional source code examiners would generally provide many of the same protections, here the locked-down computer prevents inadvertent expert leakage through, *e.g.*, Google searches.
    - Lack of an internet connection prevents the examiner from comparing the produced source code with the public domain, *e.g.*, open source code (unless some particular open source has by prior agreement already been downloaded to the protected computer).
    - Lack of web access further prevents copying VeryLongNamesLikeThisOneHere from the source and then pasting into Google (instead, the examiner must write

down the name on paper, walk down the hall, type them into an internet-connected computer; hand-write any relevant hits, carry that back to the examination room, etc.); such a search is often important to determine if code corresponds to a known standard. However, even querying for a source code snippet can leak confidential information, and preventing even inadvertent leaks is largely the point of turning off internet access.

    - Lack of web access also blocks internet-based tools, such as JavaScript beautifiers, Java deobfuscators, and Google Translate (useful, *e.g.*, for source code comments and strings in Chinese or Korean). Again, this is consistent with the very purpose of the PO, but attorneys should be aware of the impact on the source code exam.
    - No, or limited, ability to employ software composition analysis (SCA) tools which may have the benefit of employing "Big Code" repositories (see, *e.g.*, Black Duck).
    - Similarly, this may impact a source code examination firm's ability to use its in-house proprietary tools, such as DisputeSoft's Code Examiner.
- Protected source code computer only accessible Monday-Friday 9AM-6PM at producing law firm site or escrow facility.
    - This drives up the receiving party's costs, especially as experts and consultants are often based in a different city from the site of the protected source code, requiring airfare and hotel bills. These restrictive POs are often also costly for the producing party, *e.g.*, for special rooms and for employees to monitor the rooms; see below.
    - This can also have a dramatic effect on an expert's ability to follow up on 2AM brainstorms.
- Examiner may not bring a laptop, mobile devices, etc. into the source code room; usually internet searches must be done in a separate "break-out" room, which is typically provided.
    - No laptop in the same room as source code means: only *handwritten* notes allowed; in some cases, the producing party may have an opportunity to inspect or copy the examiner's handwritten notes. See EPL Holdings v. Apple on note-

taking. It is important to preserve all handwritten source code notes, in case the producing party is later permitted to inspect or copy them.

- "No analysis of source code" outside the restricted source code room: no printing for later analysis; all analysis, *i.e.*, careful reading of the code, must be done *in situ* (in the designated room).
    - This limits the receiving party's time to analyze the code (especially as it likely must travel to a different city to do so), and restricts analysis to pen and paper (see note-taking above).
    - This also restricts how the testifying expert can read and analyze source code print-outs generated by non-testifying consultants. It is difficult to think of a legitimate reason for such clauses, except perhaps to limit printing (see "Forensic Software Analyst May Not Study Printed Source Code").
- Standalone locked-down source code computer has no accessible USB ports, DVD/CD drives, etc. Thus, no ability to copy *anything* (not just source code) to or from the protected source code computer.
    - Thus, no ability for an examiner to copy software examination tools *to* the source code computer; tools must be agreed upon beforehand, and attorneys unfortunately often agree to the set of tools, without consulting experts as to the tools suited for the particular job (see below on tools). The result is especially problematic when source code has been produced inside version-control files, but matching version-control software such as SVN is absent.
    - Further, no ability for examiner to copy notes, examiner's own script output, etc. *from* the source code computer; see below on handwritten note-taking. Non-adverse clients may permit their outside examiners to copy notes to an encrypted flash drive for removal from the source code review room. However, if the underlying reason for the restriction is a "reasonable security precaution" (RSP) to maintain trade secret (TS) status, it's possible that the same restrictions applied to the other side's examiners should also be applied to one's own *outside* examiners.

    - An examiner can type in small tools, if the computer is preloaded with scripting language facilities such as PowerShell or WSH on Windows, or (on the Mac, or on Windows with Cygwin) AWK.
- "No copying of source code" interpreted to mean no direct quotations from source code in expert reports, claim charts, etc.
    - One workaround is to cite proper nouns (function names, variable names, etc.) from source code, without transcribing lines of code, or any portions of lines that contain "logic" (including anything with an equal sign or a function call). But sometimes not even this is allowed, and the examiner must *paraphrase* all function/method names in its work product (taken to an absurd extreme, the result might look like some paraphrases of the DeCSS source code, including one as a Haiku).
- Any memo or other subsequent document which quotes from, or perhaps even refers to, source code will thereby come under the same PO restrictions as the source code itself.
    - As noted above, this may mean that a party can only see a redacted version of its "own" expert's report which references the other side's source code. Importantly, a party's testifying expert witness is not intended to be under the hiring party's control.
- Source code computer only has standard operating-system-provided tools (*e.g.*, findstr, grep, diff, AWK, cscript), or those explicitly agreed to in the PO (*e.g.*, SciTools Understand, PowerGrep, Apple XCode, Microsoft Visual Studio, dtSearch, WinMerge, Sigasi (for VHDL or Verilog code), Cygwin (Linux-like environment for Windows), text editors for printing, such as Notepad++, TextWrangler or EditPadPro, etc. For a PO spelling out such tools, see here.
    - Some parties decide (even when not addressed in the PO) to turn off command-line access.
    - This can seriously impact source code examination by some examiners whose regular practice is to employ command-line tools.
- Limitations on quantity of source code files, pages, and/or lines which may be printed; the PO may also restrict print-outs to source code files themselves, not allowing the (often important) printing of directory listings, script output, etc. (see *e.g.*, *Apple v.*

Samsung, 2013 WL 1563253, preventing Apple from taking printouts of output from the "diff" program: "... if the diff reports are Apple experts' notes, those notes necessarily include source code, and removal of notes with source code references is prohibited under the Protective Order").

    - Printouts are generally reviewed by the producing party, before production to the requesting party; this in some ways is the very purpose of a source code exam. The producing party may thus glean early inferences into the other side's case, and this in turn yields a perverse incentive for the receiving party to delay printing source code files until the last-possible minute.

- Once print-outs have been created from selected source code, these print-outs often are only accessible at the receiving party's law firm office, which is often in a different city from that of its own experts and consultants. If experts and consultants are permitted to have source code print-outs, they often must be kept in a lock box.
- Proctor often may sit in room with examiner, or just outside the door of the source code room; some POs address whether the proctor may actively monitor the source code computer screen, keystrokes, file accesses, or searches.
- Sometimes lengthy sign-in/sign-outs are required even for brief coffee or toilet breaks: sometimes this results from a genuine (if possibly false) mystique around the source code's value (see Citizen Kane below), but an additional goal may be sheer annoyance (see "we're really sticking it to them" below).
- Continued impact of the PO, past the life of the case, including not only continued protection for source code and for litigation-created materials based on that source code (including notes and print-outs), but also possible attempted restrictions on future work by experts and consultants who have been exposed to the source code (e.g., *BIAX v. Brother Int'l.* on party's "arrogant" attempt to restrict opposing expert from similar work for four years, based on exposure to source code; such a restriction would seriously limit the "pool" of experts needed by the patent litigation system).

In other words: no laptop in room; handwritten notes only; no direct quotations from source; no internet connection; limited printing; all analysis must be done *in situ*.

While some of these restrictions are likely reasonable under some circumstances, as a whole it is reminiscent of the Thatcher Library vault room scene in Citizen Kane.

An examiner should carefully read the PO before a source code examination, and should bring a copy of the PO with them to the source code exam, should questions arise regarding note-taking, printing, command-line access, etc.

Typical source code PO restrictions are also discussed in an excellent law review article by Professor Lydia Pallas Loren & computer scientist Andy Johnson-Laird, "Computer Software-Related Litigation: Discovery and the Overly-Protective Order," 6 Fed. Courts L.R. (2012); the article covers, *e.g.,* security for printed source code in transit; proscribed items in a source code room; requirement that examiner only take handwritten notes; restriction on "studying" source code outside source code room; proctors; etc.

Blanket source code POs, and open source and trade secret problems

Overly restrictive source code POs often reflect a confusion (or deliberate mystification) of source code, as though "The Source Code" were -- merely by virtue of being source code -- in and of itself and as a single whole, a trade secret (TS) and/or confidential business information (CBI).

However, much of the information in source code, though certainly not all, is often already public (most clearly in the case of open source), or readily ascertainable, *e.g.*, by reverse engineering a product on the market (apart from license restrictions on reverse engineering, which may not apply in litigation; see article on reverse engineering to investigate software patent infringement). Consider for example a PO on Java source code for an Android app, when the app itself is publicly available, has not been obfuscated, and thus can be readily decompiled (if nothing else, its decision not to obfuscate its publicly available code may be a factor in determining whether the producing party truly regards its code as a TS).

A blanket (or umbrella) PO is typically placed over the *entire* body of produced source code, even when that produced source code largely contains open source, albeit perhaps with significant vendor modifications. The blanket PO is typically in place even if the owner has not indicated the presence of any specific TSs (*e.g.*, server code only directly accessible behind a firewall) and/or of sensitive materials (*e.g.*, code relating to passwords, security, or encryption) contained within the source code. A colleague reports that in the *Bedrock Computer Technologies, LLC v. Google Inc.* case, "Whenever the source code was shown in the trial, the courtroom was cleared except for people directly involved in the trial. The source code in the case came from the [open source] Linux networking stack with some modifications by Google."

The most confidential portions of the source code are often "comments" by the programmers, which are generally completely removed before distribution of software to consumers (in contrast to source code itself, which is often compiled into object code, which in turn often retains some symbolic information such as API and other function names), and hence not susceptible to reverse engineering. But such comments may not hold up to a TS analysis of whether the owner derives economic benefit from others not knowing them.

Not surprisingly ("if everything is special, then nothing is special,"), blanket treatment of a company's source code as a single TS can easily lead to underappreciating the specific TSs which really can reside in source code. See below on TS compilations.

A company sometimes *doesn't know* its own TS before the issue comes up, usually in TS litigation against ex-employees. Of course, the company as a whole has collective knowledge of its own technology, but it may not have thought through *which* portions of its technology are actually TSs (*If Only We Knew What We Know*). Unlike patents which the company must proactively demarcate beforehand from the rest of its technology (see series on patent claims as demarcated boundaries), TSs (like copyrights), do not require such upfront designation until the start of litigation

Further, most attorneys would be hard-pressed to do a TS review of a large volume of source code, similar to the standard "privilege review" of emails to be produced in litigation.

Thus, it is easier for the producing party to use a blanket or umbrella designation of all source code under a PO, have the receiving party extract/select the specific files it wants to make its case, and then review those selected items before printing/production. At this point it likely would be simple to decide whether these items are TSs, and thus whether further PO restrictions (such as “no analysis outside the source code room”) apply. However, this opportunity for subsequent review is not typically used.

Under the Federal Rules of Civil Procedure (FRCP) 26(c)(1), a party seeking a PO has the burden of demonstrating "good cause" by showing a particularized need for the protection sought; good cause is shown “when it is specifically established that disclosure will cause a clearly defined and serious injury. Broad allegations of harm, unsubstantiated by specific examples … will not suffice” (*Glenmede Trust v. Thompson*; see also tobacco litigation *Cipollone v. Liggett*; prozac litigation *In re Eli Lilly*).

The need to specifically show "good cause" appears to be ignored once source code is involved, either because of the common but incorrect assumption that source code itself, as a whole, is a TS (even though no one would ever regard the near-parallel category of “company internal email” as inherently a TS), or because it is (with some good reasons noted above) seen as too time-consuming or expensive to figure out beforehand where the TSs are located within the code as a whole.

As already noted, companies often have little idea what their valuable trade secrets actually are, until the issue comes up when employees walk out the door to start their own company ("you don't know what you got until it's gone"), and many litigants would be unprepared, if pressed, to show particularized good cause (*i.e.*, something more specific than "our source code is our crown jewels") as typically required in PO case law.

One solution may be for the producing party to have its own engineers review the relevant source code for TSs or other confidential information, prior to production to the other side. This would be feasible for smaller amounts of code, especially if the original developers are still available,

but may not be feasible for large bodies of code (it's not as if there were software that could automatically scan files for the presence of TSs), and often the developers most knowledgeable about source code have left the company by the time litigation occurs. As it is, a corollary of a blanket source code PO is often a blanket source code production, which often includes irrelevant as well as relevant code.

Because of the special blanket protective status typically bestowed on source code as a general category, and because the issue comes up again and again in civil litigation, there are model POs covering source code. See, *e.g.*, N.D. Ca. (section 9 on source code); E.D. Tex.(paragraph 10 on source code); D. Del. ("Default Standard for Access to Source Code"); ITC ("Source Code Provision to be inserted in Model Commission APO;" see also commentary).

Some of these model POs enshrine restrictive source code examination practices that are unnecessary in most cases. While there are benefits to a one-size-fits-all standard over case by case customized procedures, the current *de facto* standard source code PO restrictions are a poor choice. The scenario noted at the beginning of this article -- two source code trees requiring comparison with each other, but located on two different machines in two different cities -- may arise in a copyright or TS case when adopting a court's model source code PO that was designed for use in a patent case. In software patent litigation, source code is compared to patent claims, not with other source code, so the use of a patent related model PO for a copyright or TS case -- which likely requires source-to-source comparisons -- is misplaced.

Note that the discussion above regarding the TS or non-TS status of source code should be relevant in any software litigation in which the other side's access to source code is being restricted with a blanket PO, based on claims that the entire body of source code is a TS. This is distinct from software TS cases specifically, in which a *substantive* issue will be whether source code, allegedly taken by defendants, is a TS. For further perspective on "Examining and Protecting Trade Secrets in IT Litigation," see an article by DisputeSoft founder Jeff Parmet.

Legitimate reasons for source code PO restrictions

My experience is that source code POs are becoming more restrictive. Some underlying reasons are legitimate, while many appear to be illegitimate. First, some *legitimate* reasons for PO restrictions on source code. Not surprisingly, all the following reasons boil down to the goal of protecting the producing party's TS or confidential information, including from inadvertent leakage, while still providing the receiving party with access to information to make its case in litigation:

- Keeping the other side's engineers and patent prosecutors from seeing your source code; see AEO and patent prosecution bar above. This legitimate reason -- keeping source code from competitor engineers -- would seem at first inapplicable to "trolls" or non-practicing entities (NPEs) which, by definition, are not putting a patent into practice. But an NPE may be actively prosecuting patents, and the last thing D wants is P using its source code as a pointer to help P amend its patent claims.
- Maintaining TS status (especially vis-a-vis one's own employees, in case of future TS litigation) of source code, by noisily exercising "reasonable security precautions" (RSP) in this current litigation with outsiders. But then, the producing party's own outside experts and consultants perhaps should (despite the inconvenience and cost) be similarly restricted.
- Protecting the likely genuine TS or CBI status of some portions of the source code, coupled with the perceived difficulty noted above of demarcating these portions beforehand (though shouldn't the source code owner know some specifics about its "crown jewels"?), may favor a blanket PO on the source code as a whole.
- Perhaps protecting the source code as a whole as a single TS "compilation." Even if known to be comprised largely of publicly-known information such as open source, a company's source code might be viewed as a secret aggregation of otherwise public information (see United States of America v. David Nosal). While reasonable sounding, the author has never heard this put forward as a reason for a blanket PO. TS compilation status likely requires showing that the selection itself is a valuable secret that has been reasonably protected. (However, see *e.g.*, *Decision Insights, Inc. v. Sentia Group, Inc.*, in

which a source code compilation was held to be a TS, apparently because it contained TSs.)

The above could all be done through standard NDA terms, without the need for special source code rooms, analysis only *in situ*, and so on. These terms could be enshrined in a court-ordered PO, with contempt of court for violations.

Some legitimate underlying reasons appear to favor some of the highly-restrictive clauses:

- Preventing inadvertent leakages (through, *e.g.*, Google searches for function names appearing in the code) that might occur were the source code viewed on an internet-connected machine.
- Preventing source code printouts from floating around and being misplaced.
- It would be too difficult to monitor or enforce the other side's experts and consultants with source code access from their own offices or laptops. Apart from inadvertent leakage on the level of small phrases pasted into Google, such distrust of the professionals working in patent litigation (for whom misuse would be too career-adverse) could only be justifiable if the source code as a whole really is a TS, which as shown above is almost never demonstrated beforehand.

Even with these legitimate reasons, litigants should ask not whether protection in the abstract is needed, but whether a highly restrictive PO fulfills those reasons better than would an NDA, to an extent that justifies the extra burden to both parties of the PO.

Less legitimate reasons for highly restrictive POs

Some less-legitimate reasons to seek a restrictive PO include:

- A desire to create artificial “crown jewels” aura around source code. This may later backfire in damages calculations, *e.g.*, when the source code owner requests a restrictive PO on the basis of the valuable "crown jewels" or “secret sauce” nature of what it may later, to reduce damages, characterize as its stale, no longer used code (“oh, that old thing”). It may also backfire when the party turns out to have *lost* some of the source code; because the wheels of justice grind slowly, litigation in this area will often be about partially missing older code (specific versions and dates are often crucial in source code examination), which the owner will have previously held up as its crown jewels.
- A desire to make source code examination inconvenient, expensive, and time consuming for the opposing party, especially if that party is seen as otherwise immune to normal discovery mutuality as discussed by Professor Robert Bone in *Economics of Civil Procedure*.
- The producing party’s law firm may enforce bizarre restrictions (*e.g.*, no coffee in the source code room) to keep its client happy: "look, we're really sticking it to the other side." Curiously, the law firm sometimes relaxes these restrictions when the client is not present.
- Cynically, the producing party’s law firm may be billing its client by the hour, and/or may charge a markup for expenses. Elaborate PO measures can be very expensive for the producing party as well as for the receiving party.
- The producing party may desire early access to the receiving party’s source code examiner’s work product, with the ability to view this as a side effect of the PO.
- Desire to make the other side's discovery less thorough; a testifying expert may be hesitant to complain about how a PO-restricted source code exam negatively affected their work, or that of their assistants or consulting non-testifying experts. You can just hear the deposition question/answer: "Q. Your source code access was greatly restricted, Dr. Expert, so your report is based on only a partial or limited view of the code, unlike the conclusions of my expert, who had full access to our own source code?" "A. But wait, you're the one who put those very restrictions in place, despite knowing that over half the source code was open source." "Q. Be that as it may, nonetheless your view was restricted and my expert's view was not?" There may be perfectly good answers, such as "A. Actually, no, because like I said much of the code was open source, and I examined

> that, and also your product was shipping with debugging symbols enabled, so the product itself is practically an open book, even without the source code. We needed your produced source code largely to *confirm* what we already knew from the product itself, and so your restrictions made that confirmation more expensive and time consuming, but didn't undermine the basis for my conclusions." Also, note the point made earlier that if the PO's purpose is to protect TSs, the producing party's outside experts perhaps should be made to operate under the same restrictions as the receiving party: "what's good for the goose is good for the gander."

It is possible that some PO restrictions will interfere with what would otherwise be the examiner's normal non-litigation practice (which is, in turn, a *Daubert*-related factor in assessing the reliability of testifying experts). At the very least, the typical source code PO will affect the examiner's normal working style, and will differ from how source code is typically examined in non-litigation contexts (though the purpose is also different). Nonetheless, while operating under sometimes-strange PO restrictions, the examiner must do a reasonable examination of the source code.

One partial solution to the overly restrictive PO is based on reverse engineering, as alluded to above: when the other side has a publicly accessible software product, this can often be examined for code related information in a much less restrictive environment than that for the PO restricted source code, and the PO restricted source code examination used largely to confirm what is already known from the publicly-accessible product. Future articles will cover the use of reverse engineering as a litigation tool, including legal concerns over software reverse engineering (such as, "doesn't the DMCA or a clickwrap license prevent me from doing a detailed pre-filing investigation of a software product?").

A plea for reasonable PO restrictions (and for more particularized discovery requests)

Those negotiating source code POs should not be overly impressed by utterance of the seemingly-magical incantation "source code." It is far better to think of source code as akin to a

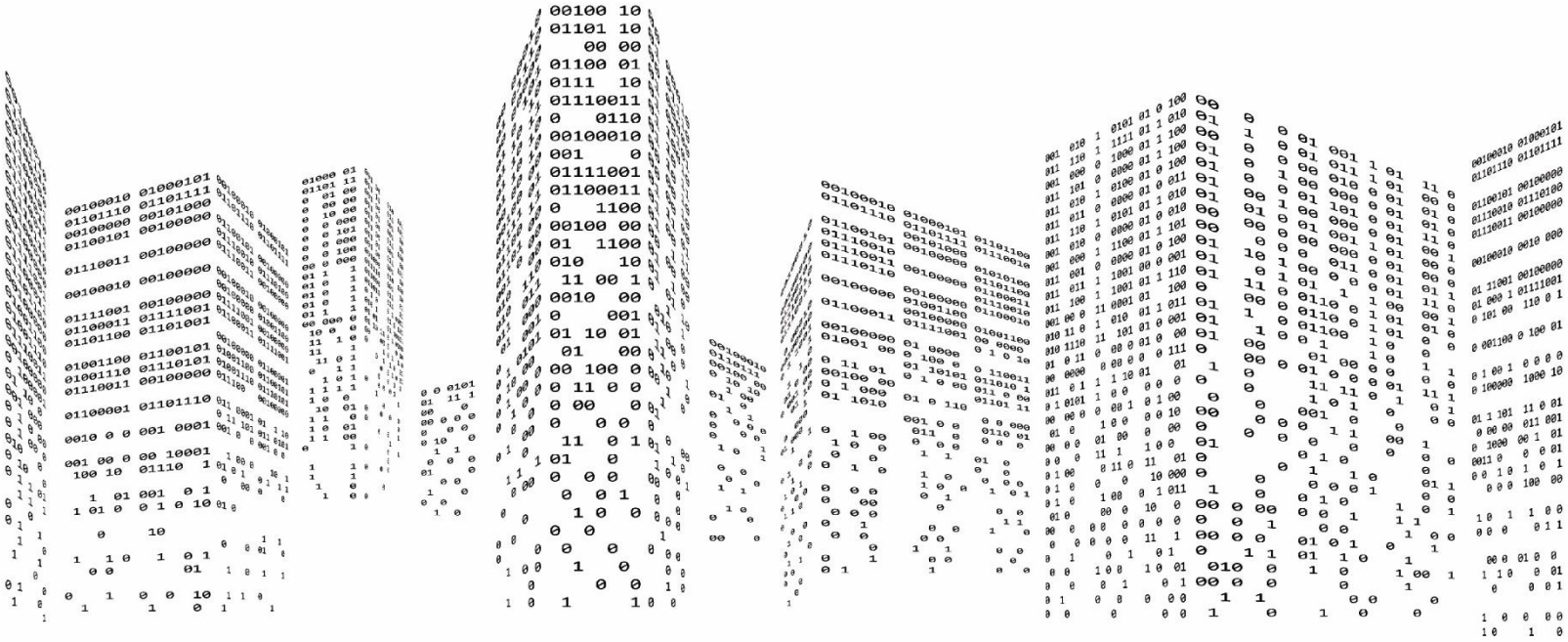
company's confidential internal emails, *i.e.*, as just another type of document (albeit one which definitely has a more direct role in producing the company's products).

Attorneys negotiating source code protective orders should try to have their technical expert or consultant assess the impact of a proposed or model PO will have on the source code exam, and on the ways in which the needs of a given case may depart from the assumptions of a model PO. Attorneys should push back against certain restrictions, such as forbidding a laptop in the same room as the source code; such a ban results in a requirement that all notes be handwritten, which raises costs.

At the same time, receiving-party attorneys should also try to limit source code discovery requests. As noted above, the receiving party can often learn beforehand (from examining a publicly-accessible product) the names of, and then in discovery ask specifically for, particular relevant files and functions, so that the other side is less tempted to retaliate with, or has less excuse for, overly restrictive source code POs. Were more carefully particularized source code discovery requests to become more common, hopefully unnecessarily restrictive source code PO clauses would become correspondingly less common.

=====

Andrew Schulman is a Senior Software Litigation Consultant at DisputeSoft. He focuses on software patent litigation, pre-litigation investigations, and source code review. Mr. Schulman is also the founder and principal of Software Litigation Consulting. As a software engineer, he edited and co-authored several books on the internal operation of Microsoft operating systems, and is an attorney with an LL.M. in Intellectual Property.



If you are in need of a software patent dispute expert, we invite you to consider DisputeSoft.

Contact Information

Jeff Parmet, Managing Partner
301.251.6182
jparmet@disputesoft.com

12505 Park Potomac Ave. | Suite 475 | Potomac, MD | 20854