Patent Litigation: An Introduction to Patent Claims, “Limitations,” Infringement, and Invalidity -- Part Four

Andrew Schulman

Senior Software Litigation Consultant, DisputeSoft

SoftwareLitigationConsulting.com

In Part 1 of this six-part series, we discussed how the actionable part of a patent is its “claims,” and in Part 2 how claims are made of “limitations,” which are generally those elements of an invention necessary to distinguish it from earlier technology (“prior art”). The previous installment, Part 3, looked at the following:

- Using a patent claim as a parts list or checklist to investigate infringement by searching for products that include the so-called “limitations” (elements) of the claim.
- Defending against a patent owner’s assertion of infringement by showing a product’s absence of one or more limitations of an asserted claim, and/or by showing the patent claim is invalid (*e.g.*, by finding each and every limitation in a piece of prior art).
- Why nomenclature is important: the names the patent owner uses for claim limitations, and the often-different names used for matching elements found in accused products or in the prior art.
- How a patent claim often changes during examination by the patent office (PTO), and why such changes -- often made to avoid the prior art -- are important for understanding the scope of the claim.
- Why the drawing on the front page of a patent isn’t necessarily a good representation of a patent claim.
- Why the title of a patent, or what otherwise seems like what the patent is “all about,” may not be the focus of patent litigation.

In this Part 4, we’ll look at:

- Investigating possible infringement of a software patent claim (see Part 5 for investigating possible invalidity);
- Brainstorming search terms when using a patent claim to find infringement;
- Testing a possibly-infringing product for infringement, with a limitation-by-limitation comparison of a claim to an accused product (we'll see in Part 6 that the Local Patent Rules in key federal districts require this limitation-by-limitation comparison);
- How to structure the comparison of a single claim limitation with something possibly matching it in an accused product.
- "Means for" and functional claiming; and
- The Doctrine of Equivalence, and the function/way/result test for equivalence.

Some terminology that we'll be using in this part of the series includes:

- Construed claim: a patent claim that incorporates its meaning (following the process of claim construction), in contrast to the raw language of the claim (in the absence of claim construction).
- Accused product: defendant's product (or service or internal usage; more formally called "accused instrumentality") that the patent owner has accused of patent infringement.
- Candidates: Products or services that a patent owner is considering as possibly infringing, and requiring further investigation.
- Product attribute: what in an accused product is being juxtaposed with a claim limitation.

Investigating infringement of a software patent claim

In Part 3, we worked through each of the limitations that comprise claim 1 of a software patent (7,472,398, titled "Method and system for hosting an application with a facade server"). But we did so entirely on the patent's own terms, without considering any specific possibly-infringing products. In this Part 4, we're going to actually *do* something with the patent: go looking for infringers, *i.e.*, outsiders who are practicing the patent, who might therefore be targeted with a request to license the patent (with a possible threat of litigation).

By having first worked through the patent claim in Part 3, without a specific target in mind, we followed the order the courts dictate: first do "claim construction" (determine the meaning of the claim's terminology), and then read the construed claim (not the raw claim language) onto a product that has been accused of patent infringement. "Construed" refers to the interpretation of the claim's wording, following claim construction. On the other hand, many patent owners will have started with whom they wish to sue, and with that target's products or services which seem to relate to one or more patents in the owner's portfolio, and only then back into an interpretation of the patent claims that helps create a match with the target's most valuable products.

Either way, the patent owner must at some point conduct a patent-to-product comparison. Note: *not* a comparison of the plaintiff's product with the defendant's product, but of the plaintiff's *patent* to the defendant's product).

The reader will know from earlier parts of this series that this comparison of the patent to a product must be based on the patent *claims*, not on the patent's title or drawings, nor on something vague like the inventor's aggrieved "They're doing X, and my patent is the only way one could possibly do X, so they stole my idea!" gut-level feeling.

Further, this claim-to-product comparison must focus separately on each and every *limitation* (element or step) that comprises the claim. At this stage, one puts on blinders: for the first limitation, what matches it in the accused product?; now the second limitation; and so on.

We walked through a little of this procedure in Part 3 when looking at a short simple claim in a toner-related patent (8,951,704), and saw that, out in the real world, there were well-known named commercial embodiments for two of its three limitations. These commercial embodiments were made by companies such as BASF and had names such as "IGACURE 819" and "KarenzMT." To infringe, a product would need to incorporate all three limitations, and even these names for two of the limitations might just be a few of many possible pseudonyms, but knowing some alternative *names* for some of the limitations is an important first step in the search for infringement.

Why this focus on *names*? Engineers (for whom patents are ostensibly written) care about what things *are*, not what they're called. As Juliet almost said to Romeo, a pentaerythritol tetrakis(3-mercaptobutylate) by any other name would smell as sweet (though the toner patent actually says it is "preferable from the viewpoints of having less bad odor and the like").

But patent claims use human language to set out the boundaries of an invention. This means that those working with patents, to look for infringement and invalidity, have to immerse themselves in how language maps onto technology. Much of patent infringement analysis is about names or, to put it more formally, *nomenclature*.

Here, for each claim limitation, we (currently playing the role of patent owner, plaintiff, or P) are going to want to provide some name for what corresponds to the limitation in an accused product or service belonging to the defendant or potential defendant (D). Or, if D's product doesn't have a formal name for this component, or if we're too early in the litigation to have learned what names D uses in its internal documentation, we'll want to at least designate a location (*where* in the product or service a given limitation is found; see Part 6 on claim charts required by Local Patent Rules).

One can't just say, "D meets the 'framis' limitation of P's patent claim, because D's product includes a framis." Actually, plaintiffs often do say just that sort of thing, but it's called "aping" or "mimicking" the claim language, and courts frown on it. We also don't want to say, "Believe us, it's in there, even though we don't know what D calls it, or where it is located inside D's product, because there's no way the product could operate without it." Actually, P might say that initially for a limitation or two, before it learns more about the product it's accusing, but such "on information and belief" assertions should be kept to a bare minimum, and used only after a diligent search for more detailed, publicly-accessible information about how D's product works, and where each limitation likely resides inside D's product.

The reader may wonder about these "names" that we're saying are used for internal components of products. Often the product itself, the thing sold to consumers, may not contain any names as such. However:

- Vendors often produce manuals, specifications, and parts lists, and are often required to make certain disclosures (see, *e.g.*, schematics submitted to the FCC).
- Reverse engineering is often a prerequisite to filing a patent-infringement suit. With a textual product like software, while some names found in source code are boiled away in compilation to produce an executable software product, or obfuscated in web-based software, information regarding the internal operation and composition of software products and services is often publicly accessible to those with the right tools to view it,

and is often also reflected in materials such as config files, error logs, API descriptions, and the like.

- Last but not least, in a patent-infringement lawsuit, D is generally required to disclose relevant internal documents (including source code) to P, and vice versa, as part of the so-called "discovery" phase of litigation (Part 6 of this series will discuss pre-filing investigations, diligent use of public information, and discovery).

Typically, P's patent claim limitation is worded in one way, and a corresponding feature or attribute in D's product or service, or a corresponding disclosure in the prior art, may be described with an entirely different wording, yet still (at least arguably) be the same thing. Two different names for the same thing still fall within what's called "literal" patent infringement. That P calls it A and D calls the same thing B does not mean that A and B are "equivalents" -- the doctrine of equivalence (DoE) is a fallback position, when A and B are different, though insubstantially so. Below, we'll discuss DoE, and the "function/way/result" (FWR) test for equivalence.

Even getting the very-partial distance we covered with the toner patent claim in Parts 2 and 3 is going to be more difficult for the software facade-server patent that we started to look at in Part 3, which we're going to more thoroughly cover here, because whereas some fields -- such as chemistry, biotech, pharmaceuticals, and electronic engineering -- have well-established ways of naming components, software has much looser nomenclature. The IEEE and ACM have various classification schemes, but even when one has full source code for a software product, it typically won't reflect these classifications. To pick an arbitrary example, how many different ways are there in software of referring to a hash chain? It might be called hashChain or hash_chain, but it might instead be called a Merkle tree or blockchain. A patent claim might use one term, and an accused product a different one. Indeed, the code for the accused product might simply contain an unnamed loop such as: `while (! some_goal) x = md5(x).`

As a different example, a patent claim might include as one of its limitations the computation of a square root (no, this isn't a "patent on square root": we said this was just one limitation). Trying to find something that matches this in an accused product or system might be easy, because the square-root component might come labelled as sqrt, squareRoot, or the like. But what if the name was newtonRaphson or eulermethod, or if names were obfuscated, or if there were simply a few lines of inline code, as part of a larger function? If you asked fifty programmers to write a square-root function, you would likely end up with fifty different pieces of code, all of which might satisfy a "square root" limitation in a patent claim, but which might be devilish to locate, without knowing specifically what to look for, in a product with millions of lines of code.

Conversely, just because a piece of code is labelled "DoXYandZ" does not mean it is doing the same X, Y, and Z specified in a patent claim. Comparing the claim limitations with a product or with a prior-art reference is not a matter of mindless keyword searching or "pattern matching."

In part because of these naming and no-name problems, it often makes sense to look for the most specific, or specifically-named, claim term first, along with some synonyms. In our facade-server claim, it wouldn't make much sense to start by looking for CPU or memory or a web browser; while we'll need those, they are too generic to start off a reasonable search. Instead, the most specific thing we have is the term "facade server."

Brainstorming search terms

One search term is of course "facade server" itself, but that's not a standard term of art, so we'll also need some synonyms, or terms that are likely associated with any infringing technology. If someone were infringing, what terminology would they use for the different components or steps that would comprise infringement, and where would such terms appear? This is more-than-superficially like a forensics question: if D did X, what would evidence for X look like, where would it be, and how would we find it? Fancifully, if V was hit over the head with a facade server, and we think D may have done it, what evidence would exist that D has a facade server?

To refresh the reader's recollection from Part 3, the patent we're using as an example (7,472,398) claims a program, an application, and a "facade server," where "the program creates an interface between the facade server and a web-browser for exchanging data associated with the application," and "the facade server hosts the application without utilizing network protocols and without opening network ports."

We know (the reader may need to briefly revisit Part 3 at this point) the patent is about displaying local app contents in a web browser, and using a fake web server (the "facade server") as part of this. The specification itself, or dependent claims, likely provide clues to potentially-infringing technology that are more specific than what appears in the claim. Here, the patent specification discusses the now-ancient common gateway interface (CGI), and we know the patent claim explicitly rejects network access, so one phrase to look for might be “local CGI.” The customer is likely using the facade server to put a local web-browser front-end on a “legacy” app.

Other names might come from searches the patent examiner conducted, reflected in the file wrapper. The patent examiner was looking for prior art as grounds to not grant the patent, whereas right now we’re interested in infringement of an already-granted patent. But to adopt a phrase noted earlier in this series, “that which almost invalidated, before it was granted, might be a good thing to look for as potentially infringing afterward.”

For example, the file wrapper indicates the examiner searched for (loopback OR web browser) AND (shared memory OR named pipes). The examiner found an “interactor” in 6,717,593 that can download XML and JavaScript via inter-process communications (IPC) rather than via HTTP. While P tried to distinguish its facade server from this interactor, on the basis of the interactor not hosting an app, and therefore not being a server, at any rate this suggests that IPC connected to web browsers as another place to look for infringement.

All of this suggests, at least to a software engineer, some possible terminology that someone might be using, if they were infringing:

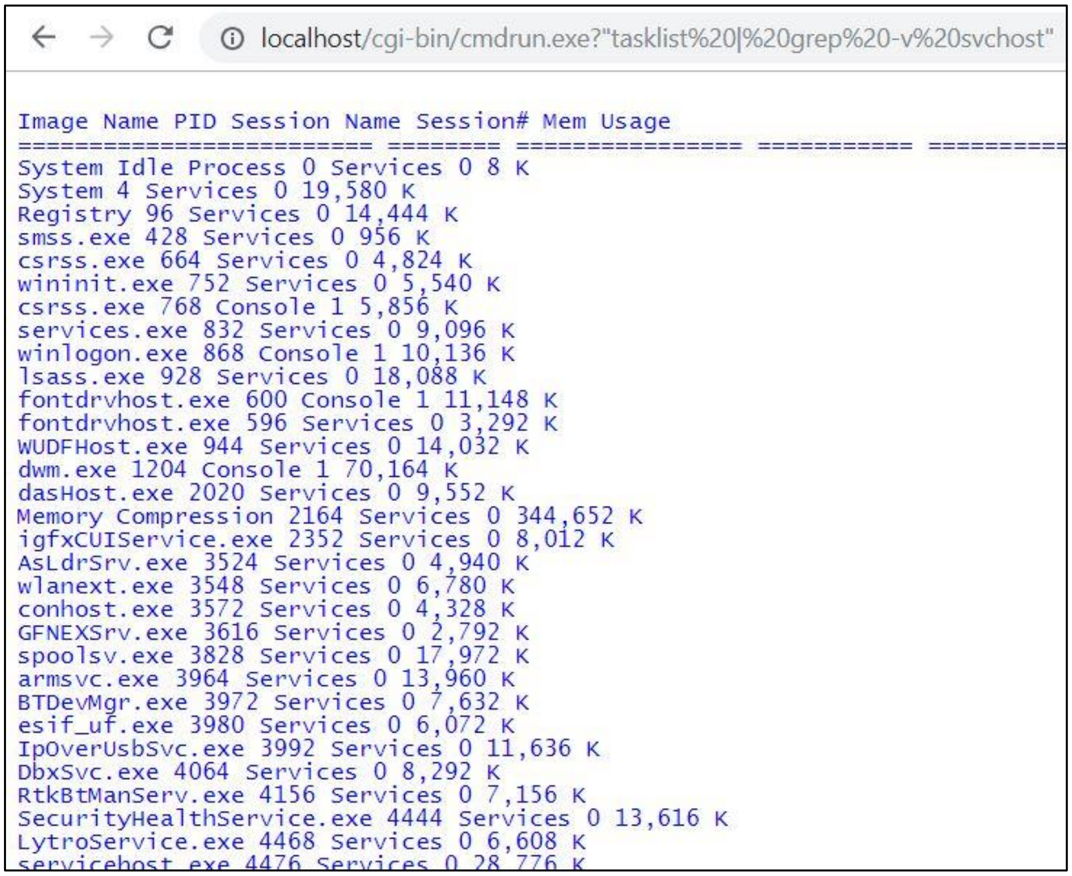
- (localhost OR loopback) AND web
- local CGI
- “cgi-bin” AND (localhost OR loopback)
- LPC (local procedure call, as a local form of RPC, remote procedure call)
- legacy; legacy AND gui; legacy AND browser; CLI (command-line interface) AND browser
- (front-end; facade; wrapper; shell; shim) AND web AND legacy
- web front-end; browser front-end
- (loopback OR web browser) AND (shared memory OR named pipes)

Which does *not* mean any product described with one or more of these terms is infringing. We're just generating *candidates* at this point.

So, armed with a set of search terms, we look for infringement of our patent claim.

A possibly infringing product

To save time, I will pretend to have found, using the search terms above, an infringing product made by D Inc.; in fact, I cobbled together a prototype using an existing web server. The hypothetical product is named Legacy2Web. and its pretend marketing literature states "Use a web browser for your locally-stored legacy apps! Put a shiny new front-end on old programs, even ones for which you no longer have source code!" It can run local character-mode command-line apps, and have their output appear in a web browser, as shown in Figure 1 below. The blue Lucida Console font indicates that the output is rendered in HTML:



```
localhost/cgi-bin/cmdrun.exe?"tasklist%20|%20grep%20-v%20svchost"
```

```
Image Name PID Session Name Session# Mem Usage
========================= ======== ================ =========== ==========
System Idle Process 0 Services 0 8 K
System 4 Services 0 19,580 K
Registry 96 Services 0 14,444 K
smss.exe 428 Services 0 956 K
csrss.exe 664 Services 0 4,824 K
wininit.exe 752 Services 0 5,540 K
csrss.exe 768 Console 1 5,856 K
services.exe 832 Services 0 9,096 K
winlogon.exe 868 Console 1 10,136 K
lsass.exe 928 Services 0 18,088 K
fontdrvhost.exe 600 Console 1 11,148 K
fontdrvhost.exe 596 Services 0 3,292 K
WUDFHost.exe 944 Services 0 14,032 K
dwm.exe 1204 Console 1 70,164 K
dasHost.exe 2020 Services 0 9,552 K
Memory Compression 2164 Services 0 344,652 K
igfxCUIService.exe 2352 Services 0 8,012 K
AsLdrSrv.exe 3524 Services 0 4,940 K
wlanext.exe 3548 Services 0 6,780 K
conhost.exe 3572 Services 0 4,328 K
GFNEXSrv.exe 3616 Services 0 2,792 K
spoolsv.exe 3828 Services 0 17,972 K
armsvc.exe 3964 Services 0 13,960 K
BTDevMgr.exe 3972 Services 0 7,632 K
esif_uf.exe 3980 Services 0 6,072 K
IpOverUsbSvc.exe 3992 Services 0 11,636 K
DbxSvc.exe 4064 Services 0 8,292 K
RtkBtManServ.exe 4156 Services 0 7,156 K
SecurityHealthService.exe 4444 Services 0 13,616 K
LytroService.exe 4468 Services 0 6,608 K
servicehost.exe 4476 Services 0 28,776 K
```

Fig. 1: Sample screenshot from a hypothetical product, Legacy2Web, showing HTML output from the local command "tasklist | grep -v svchost".

As a product, this seems somewhat uninteresting without properly rendering tabs, and without being able to further interact with the output appearing in the window. The user would want to click on a line of output, and somehow feed that back into the underlying program to get further details; perhaps the product should support something like “Expect” scripts. But at any rate, Fig. 1 looks “in the ballpark” of the facade-server claim; we won’t stop at that, but it’s a candidate worth further investigation.

In this example, the “cmdrun.exe” sample takes anything the user puts in a browser address bar after “localhost/cgi-bin/cmdrun.exe,” runs it as a shell command, and displays the command’s output in the browser. Let’s not get into the security implications of this product, except to say it had better *not* have any network connections or ports open, because otherwise any malicious web user could do what they want with this server, simply by entering arbitrary command lines. So either this product is one massive security hole, or it has a reasonable chance of meeting two crucial claim limitations (“without utilizing network protocols and without opening network ports”).

Limitation-by-limitation comparison of a patent claim with an accused product

We can’t be satisfied with how this product merely seems reminiscent of the facade-server patent. We need to compare the facade-server patent claim limitations (for which we’ve done some initial claim construction in Part 3) with what appears to correspond with each limitation in the Legacy2Web product. It might feel like we already did this in Part 3, but that was claim construction in the absence of a target product; now we’re comparing the (construed) claim to an accused product. Here, duplicated from Part 3, is the claim:

- [1pre] 1. A computer system comprising:
    - [1a] a central processing unit (CPU);
    - [1b] a memory unit coupled to the CPU;
    - [1c] an application stored in the memory unit and executable by the CPU;
    - [1d] a facade server stored in the memory unit and executable by the CPU; and
    - [1e] a program stored in the memory unit and executable by the CPU,

- [1f] wherein the program creates an interface between the facade server and a web-browser for exchanging data associated with the application,
- [1g] wherein the facade server hosts the application without utilizing network protocols and without opening network ports.

Part 3 noted the [1a] **CPU** and [1b] **memory** limitations might present issues, but for now note that D Inc.'s website says the product runs on Microsoft Windows (which in turn uses an Intel-compatible CPU), and requires 50 MB or more free memory. The claim requires memory coupled to the CPU, and this Windows-based product comprises EXE and DLL files, which contain instructions such as `MOV DWORD PTR EAX,[01234h]`, which in turn shows memory such as [01234h] coupled to CPU registers such as EAX. It sounds dumb to spell out such basic points, and we need not spend much time on them, but we can't ignore them when using a patent claim as a device to test for infringement.

[1c] **Application**: D Inc.'s Legacy2Web product includes a CMDRUN.EXE sample application, which takes its command-line argument, executes it, and renders the output as HTML, with a "Content-Type:text/html" MIME tag. Keeping an eye on indirect infringement, we note the marketing literature (and the very product name) indicates the product is to be used with the customer's "legacy" software, which is a type of application.

Our choice of which part of the product corresponds to the [1c] application in the claim is *constrained*: whatever in the product we select as a match for the [1c] application must, per [1f], also have data "associated with" it that is exchanged between the facade server and a web browser and, per [1g], the application further must be "hosted" by the facade server. As a general statement, the choice of which parts of a product are juxtaposed with each claim limitation is constrained by how the limitation fits into the claim as a whole.

[1g] **Application hosted by facade server**: That the app is "hosted" is shown (somewhat superficially) by the "host" in the "localhost/cgi-bin/" prefix needed to run the CMDRUN.EXE sample app. That the hosting is done by the facade server is less apparent; see [1d] below.

[1d] **Facade server**: If the accused product contains a facade server, it is constrained to be whatever hosts the app, such that the app's output gets rendered by the web browser. The Legacy2Web product comes with a file called SERVER.EXE. Cursory inspection of the file with

a “strings” utility (note that we’re actually looking at the product itself, not relying solely on its marketing literature or the vendor’s website) shows that it contains text strings such as “HTTPServer,” the “text/html” MIME type, “cgi-bin,” “CGI/1.1,” and “GATEWAY_INTERFACE”. The server accepts requests, and sends responses, using HTTP; but it appears restricted to access via localhost. It appears to be implementing the CGI interface, and we will at least for now designate SERVER.EXE as the facade server. Since the CMDRUN.EXE sample runs out of the cgi-bin directory, and since SERVER.EXE appears to provide cgi-bin, we can at least provisionally conclude that this matches the claim requirement that the facade server host the application.

[1e] **Program creates data-exchange interface** between facade server and web browser: There doesn’t seem to be a separate component in Legacy2Web that does this. However, such an interface clearly exists (local app commands display their output in the web browser), and for now we’ll assume it’s either right inside SERVER.EXE, and/or that the interface is created during initial setup/configuration by another file included with the product, INSTALL.EXE. We’ll infer that turning “localhost” into messages sent to SERVER.EXE is something present in the browser (“doh,” some readers will say -- which does not mean it should be left unsaid; patent infringement analysis can include explicitly stating what everyone already knows).

INSTALL.EXE also contains the string “app://” and so likely registers an “app://” local protocol handler. We might use this to assert infringement of dependent claim 2:

- “2. The system of claim 1 wherein the program interacts with the facade server through a local protocol registered on the system.”

Note that in claim 2, the “program” actually “interacts” with the facade server, in addition (since claim 2 incorporates claim 1) to merely creating an interface between the facade server and the web browser.

[1g] **Without network protocols or network ports**: The claim only requires that this apply to how the facade server hosts the app. That’s a good thing if we want to find infringement, because if we open up the Network tab in the browser’s Developer Tools, it becomes clear the browser is turning the “localhost/cgi-bin/...” string into an HTTP request sent to IP address 127.0.0.1 via port 80. We can show that it’s entirely local; running a remote firewall test indicates that port 80

is not visibly open to the outside world. So, in that sense, we haven't opened a network port, and are not sending HTTP requests over the network. Still, D could say that, even when used exclusively for localhost access, HTTP is still a network protocol, and that 80 is a network port. That's a reasonable argument, and typical of arguments in patent litigation.

Now, this claim does not rule out all network protocol or port usage; it's only the facade server which must host the app without those, and we've seen the facade server hosts the app using CGI. But we know from claim construction in Part 3 that the facade server as a whole cannot use net protocols. This is not spelled out in the claim (which only explicitly eschews net protocols for hosting the app), but the patent owner said this during PTO examination to distinguish prior art found by the examiner, and the patent owner now is stuck with it. Therefore, even if local-only HTTP to localhost isn't a [1g] "network protocol," it's still important to P if SERVER.EXE as a whole has no non-local connections. Hopefully, this is true for security reasons, as allowing *any* net access (including links invoking "cmdrun.exe" from non-local web pages) is dangerous if anything passed to CMDRUN.EXE runs as shown in Fig. 1.

[1f] **Web browser**: The web browser is likely more of a *precondition* than a limitation (see Part 3). But even if an infringer need not make or sell a web browser, one must still be present for infringement to occur. Fig. 1 shows a web browser displaying HTML output transmitted by SERVER.EXE and generated by CMDRUN.EXE. But what if the product came with some other way of viewing the output of legacy applications? The "Equivalence" section below discusses how P might handle the presence of an XML viewer, rather than of a web browser.

Structuring the comparison of a single patent claim limitation with a product attribute

Above, when trying to write explanations of why the defendant's Y is literally the same as the claim limitation's X, it was hard not to sound forced or contrived: "D's web browser is the same as the claimed web browser, because, well, because they're both web browsers." Treatises on patent litigation provide little guidance on literal infringement, in contrast to extensive coverage of the doctrine of equivalents (DoE), in part because it appears there's nothing to say: two things are either the same, or they're not. In the words of a friend of mine who is also an expert witness on patent cases, "My job in these cases is to say that something that is red … is 'red'."

But a moment's consideration shows that things are not so simple, and it should be possible to at least *explain* how or why there is or is not a match between a claim limitation and a product attribute. Take the expert's "red is red," for example. One could at the very least refer to a standard, such as the PANTONE color system: "D's gizmo matches the 'red' limitation of P's claim, because the gizmo's color matches PQ-18-1664-TCX, which is a standard for 'fiery red' or 'fire engine red.' " One could compare RGB or CMYK color values. There is a lot to say. As a promotional video at Pantone's website says, "Color is complicated."

And the role of a technical expert, opining on whether D's product attribute Y matches P's patent claim limitation, is generally *more* complicated than announcing that a red thing is, indeed, red. Assessing literal infringement is more like saying whether a given painting is by Van Gogh, and not simply a very good forgery. This is an objective test — something really either is, or is not, by Van Gogh -- and yet it has been easy for experts to get it wrong, and for experts to have a difficult time explaining the basis for their opinions.

What does this have to do with patent claim charts? As with the art expert struggling to determine the identity of a painting, or a forensic expert determining whether two fingerprints are from the same person, the necessary patent-infringement comparisons are often not simple. In any field, saying that X is-a Y, or that X is-not a Y, is the sum of many smaller comparisons, which can (and often should) be spelled out. As noted in an excellent evidence-law paper by Liebman et al., "all evidence of identity derives its power from the aggregation of individually uninteresting matches or non-matches."

In other words, the assertion that D's product attribute Y matches P's claim limitation X can be *structured*. The previous section discussed structuring the comparison of the claim as a whole, with an accused product, by dealing separately with each limitation making up the claim. This section is arguing that some of the same disaggregation applies to each individual limitation. P wants to say that D's "gizmo" matches the "widget" limitation of P's claim? Or even that D's "red widget" matches the "red widget" limitation of P's claim? Fine, there should be some way to explain that conclusion, given that the conclusion must be made up of some smaller comparisons, and not based merely on a "shucks, they look the same to me" general impression.

One way to structure such a comparison is to first break down the limitation into subparts. For example, “the program creates an interface between the facade server and a web-browser for exchanging data associated with the application” has at least four distinct subparts: (1) facade server, (2) web browser, (3) interface between facade server and web browser, and (4) interface for exchanging data associated with the application. We could further separate out (5) application and (6) data associated with the application. Each such facet or attribute of the limitation can be compared with what supposedly corresponds to it in the accused product.

That’s fine for a long compound limitation, but what about something like “facade server”? One could first deal with server, and then deal separately with the “facade” adjective, but while adjectives often should be handled as separate sub-limitations, here that seems contrived and unhelpful.

It makes more sense to list all the attributes that we know the “facade server” must have, based on our reading of the patent and of the file wrapper in Part 3:

- Acts as a web server ...
- … but (per the file wrapper) “does not use any network protocols”;
- Has an interface to web browser, for exchanging data associated with application;
- Hosts the application ...
- … without using network protocols, and without opening network ports.

We can then show how SERVER.EXE in the accused product is, or is not the same as the facade-server claim limitation, by marching through each of these attributes:

- Acts as a web server -- As noted earlier, SERVER.EXE uses HTTP to accept requests and send responses, and supports the “text/html” MIME type.
- … but without network protocols -- HTTP is itself a network protocol, but the “without network protocols” attribute is still arguably met if P can show that SERVER.EXE only accepts HTTP on local address 127.0.0.1.
- Interface to web browser, for exchanging data associated with app -- Fig. 1 shows a web browser displaying output from the app; the browser received this output from SERVER.EXE.

- Hosts the application -- SERVER.EXE uses CGI to run the app, and to capture the app's output.
- Without using network protocols -- see local HTTP above.
- Without opening network ports -- SERVER.EXE listens on port 80, which is the network port for HTTP, but the "without opening network ports" attribute is still arguably met if P can show that SERVER.EXE only listens for port 80 on local address 127.0.0.1.

Another way to structure comparisons of single claim limitations with product features is addressed below at "Equivalence."

Two more points before we leave Part 4 and the topic of using a patent claim to investigate infringement, and shift to Part 5, where we'll use the same claim to investigate prior art that might render the patent claim invalid. We should discuss so-called "means-plus-function" claim limitations, and the Doctrine of Equivalence we keep mentioning.

"Means for…": Functional claiming

One important type of limitation, not employed in the claim we've been using, is a functional limitation. The claims we've looked at so far in this series have included limitations corresponding to elements of machines (possibly virtual machines) or to steps in a process. Such limitations are in this sense *structural*. In contrast, a *functional* limitation is one that indicates what something is *for* (what its purpose or function is), without indicating how that function is achieved (its "means" or implementation).

Reference is often made to "means-plus-function *claims*," but they really are present at the *limitation* level; a single claim might contain a series of means-plus-function limitations, possibly in combination with standard structural limitations.

Typically, a means-plus-function limitation begins with the phrase "means for…," and goes on to indicate some desired function or purpose the "means" (*i.e.*, implementation) would produce. For example, claim 15 of the facade-server patent includes a "means for viewing application data," "means for generating application data from a web-based application," and so on. In other words, the claim language indicates rather vaguely that *some* means, not specified inside the claim itself, will serve the purpose of viewing application data.

Such claim language seems slightly counter-intuitive: the limitation says “means for doing A,” yet it’s precisely such means that are *not* set forth in the claim. Instead, “means for doing A” indicates that all the claim explicitly represents is that there is something that can do A; *i.e.*, there is some structural thing that can perform the function A.

But *not any* thing that can perform the function A. Importantly, such a limitation does *not* read on any product that, by whatever means, yields the same result (in claim 15 cited above, viewing app data, or generating app data from a web-based app). Instead, 35 USC 112(f) states that a limitation “may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim *shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof*” (emphasis added).

In other words, to properly construe and then use such a claim limitation, we must go into the patent specification (spec) and find structure, material, or acts (SMAs), or “equivalents thereof,” that serve or can carry out the stated function. Those SMAs are then essentially “imported” into the raw claim language to yield what the limitation actually covers. Any such SMA in the product constitutes a match, if it serves the claimed function.

While “importing” from the spec into the claims is a cardinal sin in patent law, so is trying to work with the raw unconstrued claim language. Means-plus-function limitations are an important example of how vital claim construction is to proper use of claims in patent litigation. In our example, “means for viewing application data” cannot be used to identify infringement in anything that provides some way of viewing app data. It must be one of the specific ways that are spelled out in the spec.

In this way, the raw unconstrued functional claim language is effectively *replaced* with the disclosed means. This is also an example of how claim construction often expands the claim language: given a limitation A, and the court’s construction of A as meaning, *e.g.,* “B or C, but not D,” the search for A in the accused product has effectively been replaced with a search for B or C and not D.

So what means does the facade-server patent disclose for viewing app data? The ‘398 patent spec doesn’t refer to “viewing” app data, but it does state, “The browser **106** may be capable of rendering application data generated by application **112** onto the display **114** … The browser **106** may be any type of web-browser, such as Internet Explorer®, Netscape®, and Mozilla®.” D might lamely argue that this is a means (SMA) only for rendering app data onto a display, but not for “viewing” it. Assuming that argument failed, D would have a difficult time arguing that, *e.g.*, Chrome, Firefox, and Safari are not *also* means for viewing application data. However, if the accused product instead used a custom local-only XML viewer, for example, D could argue that this isn’t one of the disclosed means (and that it isn’t equivalent to one either), and therefore that claim 15 isn’t infringed.

Equivalence and function/way/result

As a final point before we turn to an invalidity-related search in Part 5, we’ve mentioned the doctrine of equivalence (DoE) several times in this series. What if we (playing the role of patent owner P) could find a match in defendant D’s Legacy2Web product for all but one or two limitations from the facade-server patent claim, and those one or two could only be matched-up with something in the product that was similar but not identical to what the claim required?

For example, what if the product doesn’t use a web browser, but instead comes with a custom local-only XML browser that it uses for displaying app output? This would not literally meet the claim requirement for an interface, created by the program, between the facade server and a “web browser.”

Rather than throw up its hands and admit defeat, at least for this claim, or pretend that a local XML browser is literally the same as a web browser, P can instead use DoE as a fall-back position. While using DoE is a fallback position, less desirable to P than showing literal infringement, DoE is nonetheless a crucial part of patent law. In this sense, it is somewhat like obviousness, which as we’ll see in Part 5 is also a fallback position (defendants would prefer to show the invalidity of P’s patent by pointing to its complete anticipation by a single piece of prior art), yet is viewed as crucial to patent law (it is, mostly unlike equivalence, also enshrined in the patent statute: 35 USC 103).

Using DoE, P will argue that D’s custom local-only XML browser is, for purposes of this patent, *equivalent* to the web browser in P’s claim. P cannot merely assert this, in what is called a "conclusory" manner. Instead, P must use one of the well-established tests for equivalence to make this point.

One test is that of "known interchangeability," *i.e.*, that an ordinary non-inventive practitioner (the "person having ordinary skill in the art," or PHOSITA) reading the patent would, at the relevant time, have known that an XML browser could be substituted for the web browser (again, at least for purposes of this patent). P could for example point out that most web browsers also render XML. Because the web browser here must be local-only, P could also argue that most XML browsers can be limited to local-only access (at the very least by disconnecting the internet or by exercising tight local or intranet control in a firewall).

Another DoE test is "function/way/result" (FWR). This asks whether a feature in D's product, while not identical to P's claim limitation, nonetheless serves largely the same **function** (purpose or role), in largely the same **way** (implementation), to produce largely the same **result**. Each of function, way, and result must be analyzed separately; P often tries to slur them together in a conclusory manner. Note that the function, way, and result are not of a web browser generically, but of the one specifically claimed in this patent.

In P's claim 1, the **function** of the web browser is to reside on the other side of the data-exchange interface from the facade server (note the function in this claim is not necessarily to display app output). The way and result are less evident from the claim itself, but we'll make a small leap and argue the **way** is reading data from the interface, and the **result** is that the app, hosted by the facade server, has its output displayed. In the version of D's Legacy2Web product which uses a local-only XML browser, the **function** would also be to sit on the other side of interface from the facade server; the **way** is also to read data from the interface, and the **result** is to display this data (which now presumably is in XML rather than HTML format, with the display an XML tree rather than a local web page).

Note these DoE tests are performed *on particular limitations, not* holistically on the entire claim.

While the analysis above feels tedious, actually P probably ought to explain its literally-infringed limitations with a similar level of detail. After all, if two things are literally the same for purposes of the patent claim, all their relevant attributes (including function, way, result, inputs, outputs, and so on) must also be the same, and P should be able to point out some of these relevant matching attributes, rather than conclusory leaving it at “they’re the same.” This was discussed earlier in the “Structuring the comparison” section, though now having seen the function/way/result test for equivalence, it should be clearer what it means to structure a comparison.

Figure 2 illustrates the difference between literal infringement on the one hand, and infringement under the doctrine of equivalents on the other hand. In the "literal infringement" portion of the figure, A B C D in the claim exactly matches W X Y Z in the accused product, but there is not quite such an exact match in the "equivalence" portion ("foo" has a different shape and an additional outward connection, not found in C). Under the function/way/result test, equivalence could be shown if "foo" plays substantially the same role as C (note from the arrows that it has the same connections to W and X, that C has to A and B; though "foo" also has an additional connection to Z, which C does not have to D), is implemented in substantially the same way, and yields substantially the same result as C.

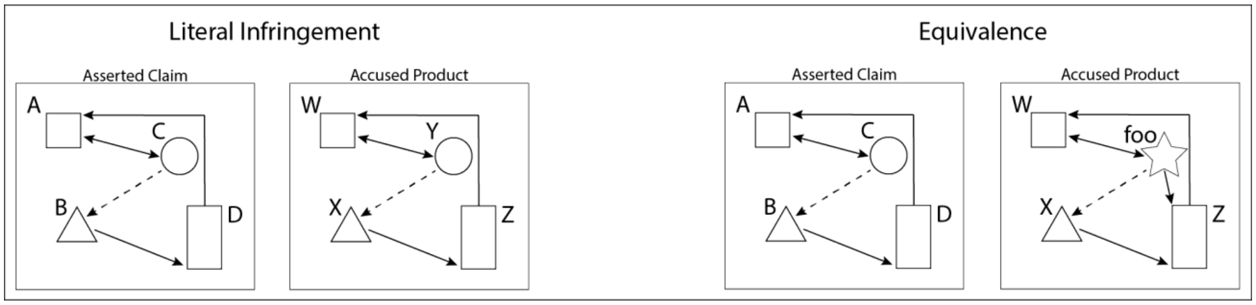


Fig. 2: Literal infringement of a patent claim requires that each and every limitation of the claim (here, elements A B C D) must be present in an accused product, albeit likely under different names (here, W X Y Z). Under the doctrine of equivalents, one or more claim limitations may not be literally present in the accused product, if it can be shown that a substitute product element (here, "foo") is equivalent to the missing claim limitation (here, C).

There are restrictions on what P can point to as an equivalent. If, in order to acquire the patent in the first place (during patent prosecution), P told the patent examiner that P’s limitation A differs from some B the examiner found in the prior art, P cannot turn around in litigation, find a B in D’s product, and point to that B as an equivalent to P’s A. This is called prosecution history estoppel (PHE) or “file-wrapper estoppel.” Estoppel means a litigant is precluded from arguing a point.

That P shouldn’t be allowed to talk out of one side of its mouth in patent prosecution, and then out of the other side in patent litigation, is a general point (not limited to DoE) that we noted earlier, with the patent-law “nose of wax” simile: P cannot treat its patent claim like a flexible toy nose, twisting it one way (to originally get the patent granted, or later to escape invalidity) and another way (to capture infringement). Somewhat similarly, D can’t point to something in the prior art as anticipating P’s claim, and then turn around and argue that the same thing, in D’s own product, doesn’t help show infringement. The invalidity/infringement “near-mirror image” (discussed earlier in this series) helps inhibit litigants from taking unreasonable positions.

As another possible limit on equivalence, what if P points to something in D’s product as equivalent to a limitation in P’s claim, and that thing was already known, or foreseeable, at the time P applied for its patent: if it’s so darned equivalent, wouldn’t P’s claims have already covered it (even if not having explicitly noted it)?

Conversely, what if P points to something in D’s product that simply didn’t exist at the time P applied for its patent (so-called after-arising technology): should pioneer inventor P be able to “reach through” the claims to capture later progress in the field? That would be consistent with the “prospect theory” noted earlier in this series (a patent is like a claim to future extractions from a piece of property), yet on the other hand much progress comes not only from original inventions, but from the accretion of small improvements, for which the patent system also must provide an incentive. How much do we want to reward the inventor of the “facade server,” over others who later make the bare principle work on a large scale for millions of users? We’ll take this up in a later article, and for now simply note that patent law, and intellectual property generally, is filled with such tensions.

Conclusion

In this Part 4, some of the major points included:

- Claim-to-product comparisons are limitation-by-limitation
- Patent infringement analysis generally requires disaggregated comparisons using the claims of a patent, the limitations of a claim, and the attributes of a limitation.

- Patent litigation requires thinking about how language maps onto technology; software in particular has looser nomenclature than other fields such as chemistry or electronics.
- Patent owners (P) suing for infringement must conduct a diligent search for detailed publicly-available information about how the defendant (D)'s accused product works, and about the location in D's product where each of P's claim limitations likely reside; this diligent search often includes reverse engineering to learn product internals.
- Reading the construed claim (not the raw claim language) onto an accused product: while "importing" from the patent specification into the claims is a cardinal sin in patent law, so is trying to work with raw unconstrued claim language; claim construction may in effect expand the claim language.
- Comparing claim limitations with an accused product or prior-art reference is not mindless keyword searching or "pattern matching."
- In the search for patent infringement, a key question to ask is: If someone were infringing, what terminology would they be using for the different components or steps that comprise infringement, *i.e.*, that correspond to the limitations of the asserted patent claim?
- The choice of which parts of a product are juxtaposed with each claim limitation is constrained by how the limitation fits into the claim as a whole.
- A typical argument in patent litigation might be whether port 80 (HTTP), used solely for local 127.0.0.1, is nonetheless a network port.
- Saying that X is-a Y or X is-not a Y, is the sum of many smaller comparisons, which can be spelled out: "all evidence of identity derives its power from the aggregation of individually uninteresting matches or non-matches."
- The assertion that D's product attribute Y matches P's claim limitation X can be structured, based on attributes of the claim limitation.
- Using "means for" and other forms of functional claiming requires taking corresponding structural details from the non-claims portion of the patent specification, and finding at least one such structure in an accused product.
- The Doctrine of Equivalence (DoE) is a fallback from literal infringement, but is crucial to patent law (similarly, we'll see in Part 5 that obviousness is a fallback from

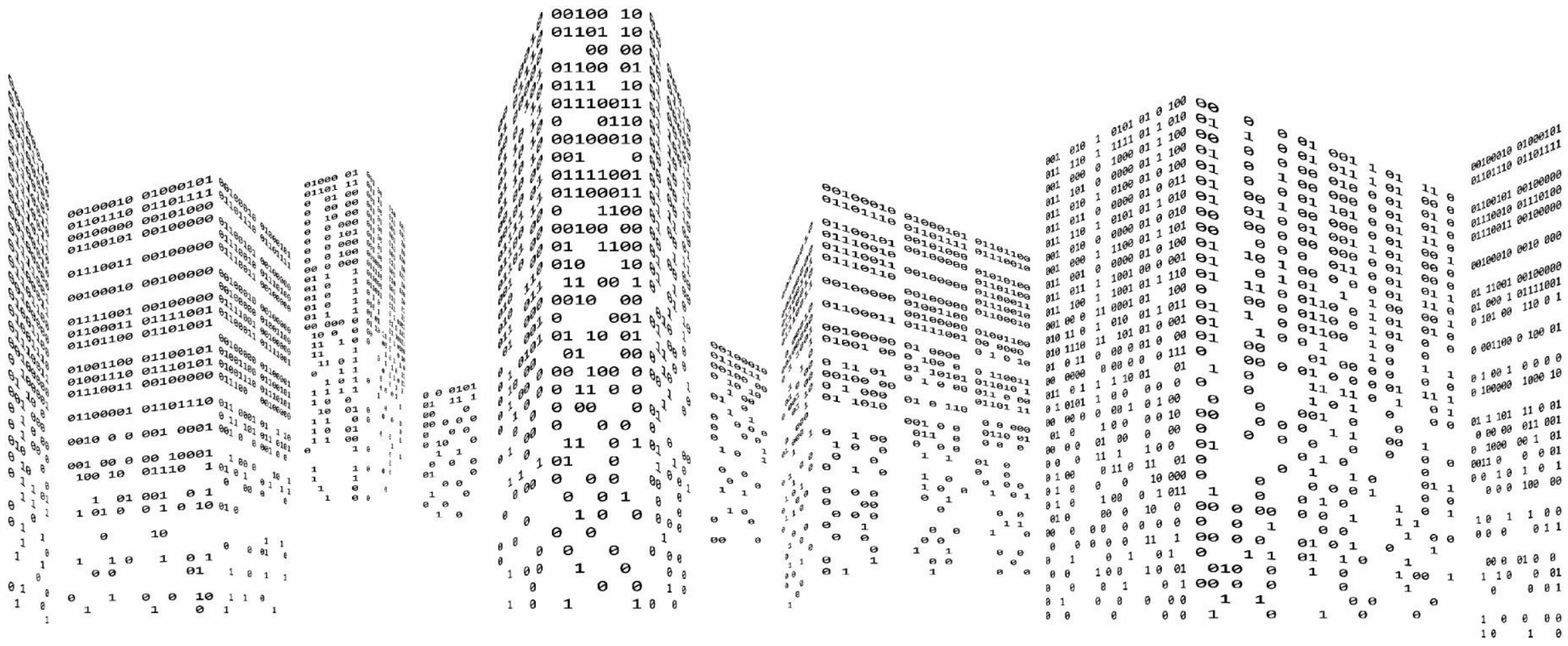
anticipation/novelty, yet plays a larger role in patent law than one might think from this fallback role).

- There are well-established tests for equivalence, such as function/way/result (FWR), which are applied to each limitation (not to a patent claim as a whole); there are restrictions on what P can assert is an equivalent.
- The patent-law "nose of wax": P can't twist its claim one way to get the patent granted (or to later evade invalidity), and then twist it a different way to capture infringement; the invalidity/infringement near-mirror image helps inhibit litigants from taking unreasonable positions.
- "After-arising" technology: how much should a pioneer be able to reach-through to capture later improvements?; improvements, by making an original basic invention actually work on a large scale, may turn out to be as important as the basic invention.

In Part 5, we'll discuss:

- The concept of patent invalidity; patents are shaky property because of the risk of patent invalidity;
- Invalidating a patent by showing anticipation (lack of novelty);
- Investigating prior art for a software patent claim: what counts as "prior art," and how a patent claim is compared with prior art;
- Prior art already found by the PTO, as reflected in the file wrapper;
- Searching for a negative limitation in prior art;
- A specific prior-art example relevant to the facade-server software patent;
- Limitation-by-limitation analysis of a prior-art reference;
- Obviousness as another way to invalidate a patent: "motivation to combine" multiple prior-art references.

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