Software Intellectual Property Rights: Is a Patent or a Copyright Best for Your Program?

by Mike Martensen | Mar 26, 2018 | Copyrights, Patents

As our world changes, the world of intellectual property changes with it. Day by day, courts across the country are signaling that the tools and methods needed to protect such property are changing too. Those not paying attention run the very real risk of getting left in the dust.

Exponential growth in computer programming and cyber technology have accelerated our reliance on the internet and electronics, on devices and apps. In this hyper-digital era where everything happens online, companies can live or die by their software.

But while innovative software has never been more valuable, protecting software IP has never been harder.

No matter what any legal advisor or attorney says, safeguarding intellectual property rights in software is incredibly challenging. It takes forethought and planning and the right team of people. But more importantly, software IP protection requires implementing the right strategy at the right time and with the right tools. Too often, companies and individuals fail to do this, losing the rights to their cutting-edge software, and their competitive edge along with it.

When most people think about intellectual property, patents come to mind. They see patents as the holy grail of IP, so it's no surprise that a company in possession of valuable code would want to know what is involved in patenting software. We are frequently contacted by software engineers who ask, "How can software be protected?"

Patent law covers original inventions - the creation of new things, new concepts, new methods and functions. But as people familiar with computer programming know, those involved in developing software aren't always inventing something new but rather building on what's already there. They take the pre-existing language of computers, and through creativity and ingenuity - changing code here and modifying a code's structure there - they might end up with something different and very valuable.

While this is no doubt a form of innovation, a type of intellectual property, it doesn't mean one can protect it with a patent.

Sometimes new software might have the flavor of original functionality and can therefore fall into the realm of patent law. There are recent rulings from Federal Circuit courts that make eligible for patents certain types of software and certain kinds of technologies and innovations.

But very frequently, patents either don't apply to software IP protection or are next to impossible to actually secure.

This is where copyrights come in as a viable means of protecting software IP.

Though different from patents in important ways, copyrights may offer a wiser alternative for software developers. While patents cover ideas, copyrights cover the expression of ideas, and are often used to protect artistic and intellectual work like that found in a song or an article like this. Think of a cookbook: a patent on a recipe would require a home chef to seek permission to use the same list of ingredients or follow the same steps of putting them together provided that list and those steps are novel and not obvious. In the world of brownies that is a high bar. A copyright, on the other hand, prevents that chef from copying, republishing, or claiming authorship of that recipe especially when that recipe includes some creative additions.

So, for companies cooking up code, seeking software IP protection via a copyright might be the better recipe to follow.

Given the confusion within patent law regarding software, and the increasing difficulty of securing a patent at all, how can one know with any certainty that copyright law even applies to software, let alone whether one can secure a copyright of real value?

Well, thanks to Google and Oracle and their multi-million-dollar lawsuit four years ago, we have some of those answers. In 2014, Oracle sued Google claiming Google's Android operating system had infringed on Oracle's software application programing interface (API) packages, for which Oracle owned copyrights. The district judge ruled that those packages were not protectable by copyrights, but when Oracle appealed, the Federal Circuit Court found that not only was Oracle's API code protected by copyright, but the underlying elements used to develop that code are protected as well. And since the Supreme Court refused to hear Google's appeal, for now, that ruling stands.

The language from that ruling is simple and straightforward: "It is well established that copyright protection can extend to both literal and non-literal elements of a computer program."

While that's great news, don't rush off to register your software's copyright just yet - the process isn't quite so simple and straightforward as such language makes it sound. In fact, there are mistakes one can make when securing a software copyright that could end up rendering the whole thing useless.

A software copyright can cover two parts of a computer program: its literal elements and its non-literal elements. A failure to properly segregate and document those two distinct components can be the kind of error from which a company's copyright attempt can't recover.

I'll prove that, but first, what's the difference between literal and non-literal elements?

The literal components generally include the two types of code: the source code, which is the spelled-out software program commands, and the object code, that lengthy list of zeros and ones that a machine can read. Because these types of code are straightforward languages, it's as easy to identify infringement, or where this code has been copied, as it is to recognize word-for-word plagiarism.

But what about everything else, like all that was used to create that code? A program's non-literal components incorporate the more nuanced and subtle elements of software that aren't as obvious and objective as its code. They include things like the code's structure, the type of code used, the language it's written in, as well as its organization, like whether it's composed of block or modular coding. There's whether its storage structure relies an internal or linked library, and how the cloud is accessed. And, perhaps the most important non-literal element for many companies is their graphical user interface (GUI), which by itself can be a major distinguishing feature of a website or app.

So a software program's literal elements exist in black and white, while its non-literal elements encompass all of the underlying grey area.

And remember from the Google vs. Oracle case that copyright protection extends to both the literal and non-literal elements.

Judges across the country hear cases about literal software copyright infringement all the time, but the issue of non-literal software infringement hadn't really been battle tested in court yet. That is, until a few months ago.

Last year, in a West Virginia district court, a company named CSS, Inc. sued a company named CT for non-literal copyright infringement.

In a nutshell, CSS had selected and arranged software components in such a way that made their transaction flow and the implementation of certain tasks more efficient. CSS alleged that CT had copied and applied to their own software these distinctive non-literal features of structure and organization, and had thus infringed on CSS's copyright.

For one of the first times, the validity of a non-literal software copyright infringement claim would be decided.

The judge had "to determine whether CT's software programs [were] substantially similar to the protectable elements of CSS's software programs."

After reviewing the facts, the judge seemed to believe that they were; however, he wrote that CSS had "not produced any evidence of substantial similarity," and therefore, two months ago, he ruled against CSS.

It is entirely likely, if not probable, that CSS would have won their case had they possessed the evidence and proof that their copyright spelled out and thus explicitly covered the unique way they had selected and arranged their software components. They may have won had they been able to provide a defendable assessment of how important to their overall software program their unique selection and arrangement of components actually was.

But they didn't have all of this, and so they lost.

Because their copyright, or at least the evidence they produced of their copyright, failed to properly identify, document, segregate, and distinguish the creative features and structures of their programs and how they worked all together to produce something unique, their creativity is now being used by a competitor.

On the surface, this ruling might seem like a blow to software engineers and companies seeking to protect their programming innovation. But it actually offers two pieces of very good news: not only do allegations of non-literal software copyright infringement have a fighting chance in court, but companies bringing such allegations have a good chance of winning if they have adequate evidence that their copyright explicitly protects such non-literal features.

But for most people, getting to that point is not easy; the path to bombproof protection for software innovation often resembles a minefield.

Martensen IP knows how to navigate that path. Our team pays close attention to the important legal cases and developments that affect the world of intellectual property. We know how to determine whether patents, copyrights, or other tools are the right ways to shield and secure intellectual property rights in software and we know how to develop winning strategies and implement them before it's too late. We know what needs to be done and when, and we've been doing that for over 20 years.

The world of intellectual property and software IP protection will continue to change, but Martensen IP will be there to ensure your company knows how to change with it.

Post Script: To cement the fact that copyright protection for software is alive and well, recall in an ongoing dispute between Google and Oracle, Google tried to defend themselves by claiming their copying of Oracle's software sequence, structure, and organization was protected by the "fair use" doctrine. But it looks like they lost there too. Just this Tuesday, a federal court ruled that Google's use of Oracle's programming technology was not "fair" and that Google had indeed infringed on Oracle's copyrights. For Google, that non-literal software copyright infringement might come with a price tag of nearly ten billion dollars.

Author's note: this article was written with contributions by Chelsea Samelson.

Contact:
Mike Martensen, Founder
mike@martensenip.com
719-358-2182
Download vCard