Does IP matter with Open Source Software? – *Video Transcript*

Key concepts for intellectual property in software

Click here for video

Introduction

Hi, I'm Paul McAdam, director of Source Code Control and in this section, we're going to be looking at Intellectual Property and how it relates to open source software.

One of the things I am passionate about in life is sandwiches – everybody loves a good sandwich, and so much so, that I put together a little application there called Rate My Sandwich. And what Rate My Sandwich does – and it's available as a webpage, it's available on your phone etc – people can upload pictures of their sandwiches, you can see the ingredients, you've got a little bit of advertising in there and what I am going to be doing is to use the Rate My Sandwich application to explain some of the key concepts in intellectual property and how that relates to open source software.

So, let's look at the key concepts, just to review them – I am sure you are aware of all these things.

Firstly, we have the idea of a patent. A patent is something that covers an invention and that invention has to be novel and not obvious. It provides the person who creates the invention, it provides them with a limited monopoly and that's there to incentivise innovation. It's incredibly important in some industries, like the pharmaceutical industry, where all the research and development activity that goes into creating something is then provided with a return on investment with that limited monopoly where they're the only organization allowed to sell something for a period of time.

The second key concept is that of copyright. Now copyright protects the way in which you have expressed the idea, the authorship, and not actually the idea itself. It protects a number of types of product, such as software, books, music etc.

The third concept in intellectual property is trade secrets. Now trade secrets are there to protect valuable information, typically maybe before they get covered by other forms of intellectual property – but there's a whole other section on trade secrets in the PatSnap Academy.

And then finally we have trademarks. Trademarks are there to protect words, logos, slogans, colours etc. which belong and are associated with the company and that's there to help the consumer understand exactly what it is they're getting – and they're getting it from the correct organization.

Now I mentioned earlier that copyright is the key concept as far as software is concerned and you can see from that little code snippet on the screen in front of you that we've got a little copyright statement in there and that's a mark that's identifying that piece of software as being copyrighted and associated with an organization. Remember that the copyright protects the created work, the way in which the work has been put together.

The key concept as far as software is concerned is copyright – and you'll notice on the code snippet on the screen that there's a copyright statement in the comments just above the code. Remember that copyright protects the way in which the code has been put together, it doesn't protect the concept, that's a patent that's going to cover that for you. And the copyright owner only has control over their own work even if somebody else has a similar idea and has put that together in code, they cannot prevent that from happening.

As far as software is concerned, copyright controls three key right. First of all, the right to reproduce the software. In other words, making copies of the software and making those available to other people.

Secondly, it's a right to create derivative works. In other words, taking the code as it stands, making modifications and then perhaps making that available to other people.

And then finally, the right to distribute – making that software available to other organizations.

Combined, what they do is protect the rights of the person who originally developed the software. So, let's turn our attention to patents in software. If I'm being honest, as far as rate my sandwich is concerned, I'm very unlikely to be awarded a patent. It's not novel and it's not un-obvious. It's a concept that other people could come up with. But if I was to be awarded a patent on Rate My Sandwich, I could potentially stop 'Burger Love' from publishing their software or making their software available, even if they created something completely independent without knowledge of my creation, because I have a monopoly on that idea for a limited period of time.

I always think of copyright as being the pin-holder and attached to that pin-holder is a sheet, and that sheet of paper is the licence. The licence is what gives you the permission or the right as somebody else to use that piece of software. The licence can have a number of terms attached to it, it can be limited in terms of geography, or for a period of time, you can have exclusive rights to the software, or non-exclusive rights to the software. You can also include on there a number of business contractual terms, such as warranty, support, indemnification and so on.

You will be accustomed to licensing from mainstream software vendors – those tend to be proprietary licences, or what we call closed source licences. The terms on those licences are unique to each individual vendor and they'll be expressed in things such as an end user licence agreement or a commercial licence – they'll have normal restrictions on usage, modification, etc and that's different to open source licences, where the licences tend to be standard – perhaps with some custom terms included.

So when I was younger, application development used to be starting with 'hello world' and keeping on going with your own code. But application development is changing. Modern application development is typically about bringing together open source components and stitching them together with your own code. If you

listen to the commentators, as much as 95% to 96% of software solutions published into market are created in this way – and with it, the approach to licensing has to change as well.

So let's look at my application Rate My Sandwich. It's a very simple application, you can see that I've got just five components on there. I've got a search box, I've got a calendar control, I've got a photo manager, there's a database behind it, we've got cross-reference capability and there's a little bit of advertising in the bottom right hand corner. But just with that very simple application, I can immediately see that I have licencing issues and operational security issues.

I've created on the right-hand side there what we call a bill of materials, so I've got a list of the components with the dates that they've been published and the licences that have been used for each of those individual components. Each component can have a separate licence, so here's a fictitious example but immediately I can see that I've got problems. So for example, Photo It has a creative commons licence but that's not for commercial use and this is clearly a commercial application. I can see that I've got a conflict between the GPL licence and the Apache licence – I was intending to publish this under an Apache licence but because I've used the GPL licence, I can't do that. I've also included Cross-Ref, which has been released under WTFPL. Now that's going to cause us problems – and more on WTFPL later.

I can see as well that I've got operational risks. If you just take for example Photo It, Photo It was last updated in 2010. Now, would you want to rely on a component that hasn't had any code commits in the last seven or eight years. How confident would you feel if there was a problem with Photo It, if there was perhaps a security vulnerability and yet no-one has been working on that particular part of the application. So, there could be security risks there, there could also be security risks for example with Datum, which as you can see by the release version, that it's a beta product and you don't want to be running your code on someone else's beta technology.

So you can see that by creating a bill of materials, immediately there's a list of actions that I need to address. Apart from the intentional incorporation of open source components in your delivered code, open source can pop up in lots of different ways. So for example, if you'd outsourced work to other developers, they may have grabbed some open source components to include in there. Or perhaps you used some legacy code from an older piece of work and there may be some open source components in there, of which you weren't aware. There are around 2,400 different open source licences, and there are over 100,000 security vulnerabilities in open source components. All of those things, you need to identify and manage. Don't forget, that even with the bill of materials, you still need to test your software output in the normal traditional ways, although they may help you become better at that.

That concludes everything in this overview. In the next section, we'll be looking at the impact to your business when these things are poorly managed.