Northwestern Journal of Technology and Intellectual Property

Volume 8
Issue 3 *Summer*

Article 6

2010

Redefining "Free": A Look at Open Source Software Management

Jon Christiansen

Alfred E. Hanna

Joseph A. Herndon

John L. Hines, Jr

N O R T H W E S T E R N
JOURNAL OF TECHNOLOGY
AND
INTELLECTUAL PROPERTY

Redefining "Free": A Look at Open Source Software Management

Jon Christiansen, Alfred E. Hanna, Joseph A. Herndon, & John L. Hines, Jr

Summer 2010 VOL. 8, NO. 3

Volume 8, Number 3 (Summer 2010)
Northwestern Journal of Technology and Intellectual Property

Redefining "Free": A Look at Open Source Software Management

JON CHRISTIANSEN,[*] ALFRED E. HANNA,[**] JOSEPH A. HERNDON,[***] & JOHN L. HINES, JR. [****]

¶1 MR. RILEY: Welcome, and good afternoon. My name is Karl Riley. I am the Managing Editor of the Journal of Technology & Intellectual Property. We will now have our final panel for today, called "Redefining 'Free': A Look at Open Source Software Management." We have four distinguished members of our panel, but I'll introduce it. Some think of open source code as a contribution to community property. Others consider it a useful tool in product placement. Either way, open source code provides unique difficulties in licensing, due diligence and patenting. We look at the pitfalls and issues that arise when open source code is included in a deal.

¶2 Today we have Jon Christiansen. Jon Christiansen is an attorney at TechLaw Ventures, a technology law firm in the Salt Lake City area of Utah. His legal practice focuses on technology law with an emphasis on software and other information technology matters. His experience includes many multi-million-dollar licensing, development, and acquisition transactions involving software and other technologies.

¶3 Mr. Christiansen is also one of the founders of EscrowTech International, Incorporated, a company that provides software, SaaS and technology escrows, data and IP archives, and open source due diligence services. He remains active in the management of EscrowTech and its business development in the U.S. and internationally.

¶4 We also have Mr. Alfred Hanna. Alfred Hanna is a registered patent attorney in Reed Smith, LLP's Intellectual Property Group, focusing on commercial patent litigation, patent prosecution, and global IP portfolio development. He has a BA in both chemistry and international relations from Knox College, an MBA and an MS in computer information systems management from Northern Illinois University and a JD from The John Marshall Law School. That's a lot of degrees.

¶5 MR. RILEY: Mr. Hanna has represented clients in a wide range of patent, copyright, trademark, trade secret, and licensing matters, involving computer software, hardware, telecommunication, wireless, mechanical, chemical, energy, pharmaceutical, nutritional, pharmaceutical, biochemical, and apparel technologies.

¶6 We also have Joseph Herndon. Mr. Herndon graduated from the University of Illinois with a degree in electrical engineering and from DePaul University College of Law summa cum laude. Mr. Herndon has experience in all areas of patent and trademark

---

[†] The Journal would like to thank the sponsors that made this Symposium possible. For their generous support, we extend our gratitude to our Platinum Sponsor–McDonnell Boehnen Hulbert & Berghoff LLP, as well as our Gold Sponsor–Knobbe Martens Olson & Bear LLP, and our Silver Sponsors–Goodwin Procter LLP and McAndrews Held & Malloy LLP. Thank You.

[*] Jon Christiansen is a Partner at TechLaw Ventures and a founder of EscrowTech International, Inc.

[**] Alfred E. Hanna is an Associate at Reed Smith.

[***] Joseph A. Herndon is a Partner at McDonnell Boehnen Hulbert & Berghoff.

[****] John L. Hines, Jr. is a Partner at Reed Smith and an Adjunct Professor at Northwestern University School of Law. He would like to thank Cecilia Bauer for her assistance with this presentation.

law practice, including patent prosecution, client counseling, due diligence, opinion work, and all phases of trial work, from pleadings to verdict. Mr. Herndon's work covers a diverse range of technologies including telecommunications, mobile Internet platforms, computer networking, aerospace/automation and control, semiconductors, consumer electronics, geophysical prospecting, Internet gaming, digital map and vehicle navigation, music discovery, automotive lighting and diagnostics, orthopedic and medical implants, batteries, software, business methods, and financial planning.

¶7 Finally, we have Professor John Hines. John Hines has taught Internet law at Northwestern University School of Law since 1998. He graduated from Northwestern University School of Law in 1981. He also teaches intellectual property at Northwestern and speaks and writes regularly on legal issues related to technology and the online environment. Professor Hines is a partner in Reed Smith's Chicago office, where he concentrates his practice in the areas of intellectual property, Internet law, and software licensing (including open source). Professor Hines is a member of the Social and Digital Media Task Force. He co-authored the Commercial Litigation chapter of the Social Media White Paper entitled "A Legal Guide to the Commercial Risks and Rewards of the Social Media Phenomenon." And now, I'll introduce Professor John Hines.

¶8 MR. HINES: Hi there. Can you hear me okay? Well, I am surrounded by all of this brain power here, all these engineers. You notice he did not say that I'm the only one with a philosophy degree.

¶9 MR. HINES: But that is significant because I'm privileged to work with this guy (indicating to Mr. Hanna). Open source is one of the areas in which you do need to have, more often than not, someone who can actually look into the code and understand and see it in three dimensions, so we will hear more about that as we go.

¶10 We're going to go from general to specific here. I'm going to give just a little introduction, and then I'll move to Joe, who will tell you about an important case, Jacobsen v. Katzer. Then we will move to Alfred, who will talk to you a little bit about some of the patent issues involved in open source, because there are many, and at the same time maybe give you a feeling of how we approach a development in a case in which we have had to deal with open source issues. Then we will end with Jon, who has just an absolute wealth of experience, not only in his law practice, but with some very interesting technology solutions that assist and supplement lawyers' abilities to deal with the very complex issues involving open source. Those issues really arise in the context of large M&A agreements involving technology-rich companies in which you have to figure out what the target has, or in a big licensing deal dealing with reps and warranties, and Jon will unfold those issues as well.

¶11 So open source licensing. Generally, what are we talking about when we talk about open source licensing? Well, what are we talking about when we talk about the licensing generally? When you have your Ford automobile and you've got a problem with your car, I can go down to Don's Auto Shop on the corner of my house, and Don can open up the hood, look into the car, figure out and fiddle around with the engine. He can see it and fix it. There may be some warranty issues. I may lose some warranty from Ford or something. But the fact is, you can do that act of opening the hood up and looking into the automobile—look into the engine. That's kind of a good way to start thinking about software, the value of software. Microsoft, the value that Microsoft has in its software is that I can't look into the hood. I can't open it up.

¶12 In addition to the legal rights, the legal requirements, the legal control under the copyright law, Microsoft prevents me from looking into their code because it's, in effect, locked. The source code, which is the human readable program code, the code that the programmers use and type in when they create the code, that's not available to me. Microsoft keeps that locked up. That's sort of the analog to the engine that I can actually look into if I want to in the case of my car. I can't do that with software.

¶13 So we're talking about source code. Source code is the blueprint of the software. It's the part that the computer actually reads, the code that is compiled from the source code, the 0s and 1s that the computer reads, the bits and bytes. People can't read that. When I get a disk from Microsoft, that's essentially what I get. I can't get the human readable source code or Microsoft wouldn't have a business. People can't read the 0s and 1s, or if they did, you wouldn't want to hang around with them very long.

¶14 MR. HINES: When we talk about source code, we're talking about the blueprints of the code, and those are secret. They are a tremendous source of value. When you do a software deal, you give somebody a license. Microsoft gives you a license, and it's for some period of time, whatever software, Adobe, whoever your provider is.

¶15 But the value of the license is not just in the hundred dollars, thousand dollars, hundreds of thousands of dollars you pay for that license. It's in the maintenance and support that you have to pay every single year. It's usually about 20 percent of the cost of the license.

¶16 So if the initial license, to get a license, say it's $100,000 or $400,000 for an enterprise license. The provider has you on the hook for another 20 percent every year because you know it's just absolutely—it's just like your car. You know you're going to have problems, and the only people that are capable of solving those problems, because the only people who have access to the source code are the providers from whom you bought. They're just delighted that it's locked up, nobody can see it, and they continue to cut their coupons and make money on this closed source. So it's a tremendous, tremendous source of value. When we talk about regular software normal proprietary software, that's what we're talking about. We're talking about where the code is locked up, the source code is locked up, okay?

¶17 The "open source" is just what you think it is. It's where developers decide to open the source code up and make it visible and actually publish it. It's published on the Internet, and you can look at open source. Open source is used everywhere, operating systems, Linux applications, Open Office, Adobe Acrobat, Mozilla Firefox, MySQL, Apache, et cetera. It's used everywhere, and you can easily get and obtain the source code. In fact, the licenses, what we call open source licenses, require the source code to be available. And you sort of say, well, why would anybody do that? Why would anybody choose to open the source code?

¶18 There's not necessarily an easy one-size-fits-all answer. There are people who feel that in a software community, a very sort of politicized software community, that believe you get much better software if you leave the source open, because more developers will have a chance to look at it and contribute. Okay? And you say, "Well, why would anybody do that?" Well, it's sort of a California thing, right? It's a very political, sort of "radical people" who are very committed to this idea that software is political. It's nothing if it's not political.

¶19 There are businesses that believe that using open source software will free them from what we call vendor lockdown; the other side, the people that buy the software say, "I don't want to be stuck paying 20 percent a year for maintenance and support. I want to have a system that's open, that I can have outside developers look at or that my own developers can look at, and I don't want to be stuck with having to pay Microsoft 20 percent every single year." So there are a variety of motivations for opening the source, but the fundamental picture here is just that. It's opening that human readable code.

¶20 It is proprietary. The fact that the code is open doesn't mean that it's not proprietary. Copyright still applies. In common parlance, "open source," as we said in the slide, is proprietary. Just because it's open source doesn't mean that it somehow loses its proprietary quality. The rights are retained. You will hear later a little bit about Jacobsen v. Katzer, which is one of the few reported decisions which upheld the rights to copyright that exist in open source software.

¶21 You can sort of think about open source as a requirement. Where the copyright owner exercises his or her right to the source code and enforces it not by the traditional methods of requiring you to be confidential, by requiring you not to reverse-engineer, by requiring you not to try to open up the code and look at it. Instead the copyright owner enforces his right by the opposite approach—by having, if you will, a covenant in the license that requires you to keep it open if you're going to use it, that requires you to give attribution to the original creator, that requires you to keep open in some instances, your modifications, requires you to give attribution to yourself to show what you changed and how you changed it.

¶22 So when you talk about freedom of free software or open source—the word "freedom" is thrown out around free software, Free Software Foundation, freedom—it's the freedom of that software. And I haven't seen this yet, but this is the way I think about it—the freedom of that software to continue to be open and to continue to "talk" to the rest of the world. So it's free in that sense. It's free in that we keep it open with the covenants and the provisions, in more or lesser degree. There are many, many different versions of open source software, and you'll hear about some of those, but it's open and free, really, in that sense.

¶23 Monetarily, it's not always free of cost. People will create businesses around open source software. For example, companies that are really capable in servicing or maintaining Linux will make money off of that, even though you might be able to obtain Linux for free.

¶24 And then there are various definitions of "open source" or "open source software" and "free software." The Free Software Foundation, which maintains some of the key open source software licenses, is a warehouse of information about open source software. Some of the other freedoms include the freedom to run, the freedom to study, the freedom to distribute, and these become requirements of the license. You can't close it off.

¶25 Probably the biggest distinction, and one of the most important distinctions, and one of the most confusing and dangerous problems associated with open source software deals with derivative works, works that are based on another work that is subject to an open source license.

¶26 So let's say I'm licensing a particular piece of software, and I'm going to build something on top of it that I'm coming up with, but the software itself that I'm licensing

to help build this other product is, in fact, subject to an open source license. And depending upon which type of license—and there are different names. They have funny names like GPL, DST, Mozilla, you can see some of the names, the Apache license, they have different names, and we will hear a little bit about those.

¶27 Depending upon which license you're using, you may be required to disclose the human readable code of the entire compilation, the entire thing that you have built. If you are somebody that is intending to make money off a proprietary model, that may not be good for you. You just bought yourself into or licensed a component that is forcing you to open up the code of the entire work which, in the language of one of the licenses called the GPL, might be construed as a work based on that first component.

¶28 So it is back to what I was saying about having Alfred around. It becomes really important to have people that can look inside of the code and help you look at the way the code is structured, to look at the topology of the design of a particular solution, to put the technology next to the particular license.

¶29 You need to know whether you are talking about one of the more restrictive licenses, such as the GPL, which has that weird language that says if something is based on a component that is licensed under the GPL, you've got to open up the whole thing. Well, you've got to know what you're really doing, and that really requires a very close cooperation between—and nowhere more in our profession than in this area—a close cooperation between the people who have a real technical understanding and the lawyers. So we will hear a little bit more about the taint issue.

¶30 But the high level issues, such as open source licensing, affect developers. They have to know what the legal consequences are of using open source, for the reasons that I was just saying. It affects buyers in big transactions. What am I buying? Am I buying something that's going to have to open up, because it's subject to the GPL license? And now I'm going to buy this thing. I'm going to find out that I bought something that I didn't know what I was buying. Now I realize I'm going to have to unfold something. I'm going to have to open up the technology once it goes to market and disclose the source code. It's really important for licensees of large licensed projects, and Jon Christiansen will talk about the audit issues. So there are a number of issues. There's the derivative, bullet point two, the taint, the forced disclosure. We just talked about that.

¶31 There's license compatibility issues, which is another large issue. If you're dealing with multiple open source licenses, frequently the terms of these licenses are incompatible with one another. So if you're a developer, you have to pick and choose the various components that you're dealing with, depending upon what the open source license is, because one open source license may not be compatible with another. Some of them just don't work well together.

¶32 Another thing about open source licensing is that this is a world in which there's a lot of technical arrogance. This is a great area to realize how valuable we could have been as lawyers, because some of these licenses are just dreadfully drafted, and they really, really require good lawyering, and we don't have any good lawyering. Some of them are drafted by people who just aren't trained in drafting. Again the issues include license compatibility, taint and the absence of any kind of warranty. When you get open source licensing, the software isn't built in a controlled environment, so you sort of don't know what you're getting. There may have been a lot of people contributing, a lot of different software developers, but maybe they picked and chose something from a

proprietary model, and you're stuck with maybe an infringing component inside some open source solution, and then you've got a hole—such as absence of warranties and absence of a controlled environment. With patents there are a bunch of issues, which Alfred will discuss. And now, I'm going to turn it over to Joe.

¶33 MR. HERNDON: You guys got a little free look of the future there. I just wanted to touch upon some of the specifics of licensing issues that you might come across with open source software. It's maybe a common misunderstanding that if you obtain open source software, you now have a right to use that in any way that you want and all you need to do is comply with the open source license that attaches to that; and if you don't comply with it, well, unlike contract law, they might be able to come after you and see you on that alone.

¶34 But this case, Jacobsen v. Katzer came out and said, well, that's not entirely true. There may be instances in which you can still be guilty of copyright infringement. You might think to yourself, well, this is open source software. I have a license to use it. How can I be infringing their copyright? It doesn't make a lot of sense to me. The devil is always in the details. You have to look at these licenses, as John said. They are drafted in unique ways, and you have to look to them for the exact language to see what it is you may or may not have done wrong.

¶35 So there's only been, to my knowledge, one Federal Circuit case that has touched upon this issue, and I'm going to spend maybe a couple slides just talking about it. That issue is the ability of a copyright holder to dedicate his work to free use and still enforce his copyright. Again, the terms "open source" and "free use," "free" meaning free of cost, but not free to do whatever you wish with my copyright—with my software that is copyrighted.

¶36 So what occurred in this case? Jacobsen has a copyright to software that he made available free of cost pursuant to an open source license. As John mentioned, there are many, many open source licenses. There are licenses that have been approved by the open source agencies, hundreds of them. There are even other licenses that have not necessarily been approved by open source agencies, but the software will still promote itself as open source software so you have to be careful with what the license itself says.

¶37 Katzer came across the software, and they put together a proprietary software package that they sold to be used, and it was built upon Jacobsen's underlying source code. So this is what John was referring to as a derivative work. They started off with Jacobsen's work, made changes to it, made additions, but they did not follow the terms of the license. Specifically, some of the terms of Jacobsen's open source license, I think it mentioned that the software needed to include a notice of copyright to Jacobsen. They also needed to include specific details indicating what had been added to Jacobsen's license—or Jacobsen's software if it were to form a derivative work. There's really specific terms that they did not follow, just little nuances, really.

¶38 Jacobsen did not sue for a breach of license. He sued for copyright infringement, and there's a procedural strategy for doing so. One of them is because they moved for a preliminary injunction. They wanted to shut down Katzer. Katzer was the competing software product out in the market that Jacobsen was losing his market share to. Jacobsen, like I said, was providing his software out to the public free of cost; so he was not suing for any money damages. He simply wanted to shut down his competitor. So you look at the two different ways that Jacobsen could have attacked Katzer. Obviously,

he had a breach of contract issue, a breach of license. Katzer admitted copying. They admitted violating the license. He could have just attacked him in that route. He did not want to do that. He wanted to stick with his cause of action for copyright infringement. Why? What are the strategies for doing so?

¶39 When you're moving for a preliminary injunction, there's the common four-factor test: likely to have success on the merits, irreparable harm, balance of hardships, and the public's interest. When you have a claim for a breach of contract, proving irreparable harm is difficult. If you have a claim for breach of copyright, if you can prove likelihood of success on the merits, irreparable harm is presumed.

¶40 So, essentially, Katzer had already admitted that he copied his code. The likelihood of success on the merits is a done deal. Irreparable harm is proven. His injunction is likely to be issued if he can maintain his cause of action under a copyright infringement case. So how do you determine if this will be a breach of the license or a breach of the copyright? Again, we turn to this open source license. If you are a user and you use open source software, you're thinking to yourself, how could I possibly be guilty of copyright infringement? The guy put this out there free of use. He gave me a license to use it. Well, if you don't follow the terms of that license, you can be guilty of copyright infringement. It comes down to conditions versus covenants, and it's such a gray area that I can't provide you with exact black-and-white details about what license would be considered—or what violations of a license would be considered a copyright infringement versus what violations of a license might be considered a breach of license. The court looked to the terms of the license. They said if those terms define the scope, that's called a condition. The copyright law applies. If the terms of the license are just terms of use, then that is a covenant to a license. The contract law applies. Here, we already said that the court, the Federal Circuit, came down and said these are conditions to the license. They define the scope, how broad it is. Therefore, your copyright action can be upheld. The copyright is infringed, and the case was remanded to determine whether or not the injunction should be issued.

¶41 Now, what would be some examples of terms that would define a scope versus terms that define terms of use? Here, the court turned to contract law, to some examples for that. If they use—it really gets down to nitty-gritty details there. If you use words in your license such as "you need to"—I think it was "provided that" you follow this, you do such-and-such—or, rather, you will be provided this license—you have this license "provided that" you do such-and-such. And in this language, it was" provided that you include a copyright notice to myself, Jacobsen."

¶42 It did not provide examples of what might be considered a term of use. The Federal Circuit, they made up their mind about how they wanted to decide this case, they just talked about the facts that were in their favor, and they didn't give you an alternative use. So we're left a little bit in the dark about what might be considered a term of use.

¶43 However, for contract law, you look to the Circuits, the Federal Circuit law that's on file. You look to the Circuit where it's sitting. So in this instance, they looked to the Ninth Circuit law. You might look up cases on breach of contract, what defines a covenant, and go through those issues.

¶44 So the takeaway from this case is that even though you have open source code, you think you have a free license to use, you still could be found guilty of copyright infringement if you violate that license. And these licenses, like John mentioned, they

are all across the board. There is not one—some of the open source agencies try to provide these common licenses, and there are some that are very simple, and they're great. And then, there are other one hundred page ones that you're wondering whether this is going to take too long to even understand. Maybe I should hire an attorney.

¶45 MR. HERNDON: There are many other issues here, open source license issues, and legal risks that you might come across. The typical license does not include any IP representations, warranties, or indemnities whatsoever. You are at the full risk here. But these licenses, they contain a broad disclaimer of all warranties, shift all the risk of any type of IP legal issues to the person using it. Again, you might look at this as thing that you get what you pay for. These are free of cost, so you're at your own risk.

¶46 As John mentioned, too, there's no representations or warranties that this will work for the purpose that you want or even that it will work. It might crash your system and fail. You know what? That's what you get. You didn't pay for anything. If you were to purchase a license, you would then have your handy-dandy hotline to call and complain to. In this instance, you do not.

¶47 Another issue that you see come up is, who is the owner of the copyright? Well, in the first instance, the very first person that puts out the open source software is the copyright owner. The author is the owner. That's copyright law. But in instances where you had these derivative works and you have many different people contributing to the progress or success of this software, you might have multiple owners, and each different person owns another small portion of this software that's now available. Who has the right to enforce that copyright? There is at least one foundation, the Free Software Foundation. If you want to represent that your software belongs to the Free Software Foundation, then you are requested to assign over any copyright that you might have to the Free Software Foundation that then states they will be the body that enforces your copyright. Once you assign your copyright, then all your rights are gone. But in instances where there are not clear assignments here, it becomes such a big deal about who owns the copyright, who can enforce it, and what you do in those instances. I think that's all I have.

¶48 MR. HANNA: Thank you, Joe. My name is Alfred Hanna, and I've had the pleasure of working with John on some very interesting IP projects. The aspect that I'd like to share a little bit about today is on navigating and managing open source and patents in a broader scope of IP strategy. I tend not to be myopic about how I manage IP, and I think that particularly in the area of software, we should not be so myopic. I find, in my experience in dealing with software developers and our software clients, that there are generally two camps: those for proprietary software and those against proprietary software. And what I find is that those who are against proprietary software are vehemently against it on the surface, but they do consult patent attorneys. They do call me, and they do at times say, "Look, I really hate patents, but I've got to manage this. I've got to deal with it. How do you recommend I do it? How do I get defensive about my open source usage?" And I think that's a very valid question, and it's a very realistic one.

¶49 So the first thing I've got to do when I speak with this kind of a client is to say, "Well, let's back up. Let's look at what patents are, and let's consider what they are not." The anatomy of a patent is pretty straightforward. The most common mistake that people make in the software business is to judge a patent by its cover or, often, to judge a patent

by the first twenty or thirty pages of the specification, called the “written description” and that is what a patent does and does not cover. A patent is really ultimately defined by the claims of the invention, and the claims of the invention are much narrower than the written description.

¶50 So if we look at it from that perspective, that can help talk a client off the ledge and say, "Hey, man, this isn't so bad." Remember what the purposes of patents are. They are a bargained for exchange to incentivize invention. They give you the right to exclude others for the claimed invention for a limited duration of time. Now that the basics are covered, we can get into the intermix and how we can consider patents and open source in the scheme of IP strategy. Invariably, there are unavoidable decision points that every company has to face. Today, I can say with almost 90 percent certainty that open source is ubiquitous and it exists in almost every hardware and software device in some element in some way or another. Major corporations have failed to track their usage of open source. They have failed to comply with licenses. Developers and software engineers have failed to report to stakeholders in corporation about their use of open source. We, as technology lawyers, want to advise our clients against the pitfalls associated with turning a blind eye. Proprietary license holders are like that Santa Claus M&M’s commercial. They're in a state of denial until one day they see the M&M's, and Santa says, “They are real."

¶51 MR. HANNA: Open source is real, and it's a potential problem, so you've got to deal with this. So the invariable decision point is, are we going to deal with open source? Are we going to incorporate it into our proprietary product or not? If you do, there are some considerations, and there are some pitfalls. You've heard, probably, those of you who have attended all day, so much about open source software, at a very high level. You've got two principal concerns. Are you concerned about a derivative work, and are you concerned about distribution? This is particularly the case for patent protection, when you're dealing with a piece of open source software that is written by many, many people over the course of time. Over a long period of time, several contributors quite possibly have contributed, and there is an issue that John referred to called taint. If you incorporate tainted open source software into your proprietary software and you go out and get a patent, you may find, to your surprise, that you have given, inadvertently, a license grant to the patent that you spent $90,000 to obtain after three office actions and a reexamination, too. You didn't know it, but the claims cover it. So you need to understand that there are issues that are involved with incorporating open source software. It affects the ability to use your patent rights bundle and may foreclose your right to exclude others. They may also limit the ability to use proprietary or dual licensing strategies.

¶52 John mentioned a case where what we did was look under the hood for our client. First, we talked to the software engineers and said, "All right. Share with me, what have you done? Give me an idea of what it is you're trying to accomplish and what software you're using." "Well, we're just using a little bit of open source, and we're going to create this really cool infrastructure that allows—and it's only going to be for internal use, and it's going to allow our internal customers to access vital information that pertains to the business on the outside." "Okay, very well. How does that operate?" "We had to go through maybe twenty or thirty iterations and discussions about how to get to how this thing really works." But it wasn't until then did we understand how to advise

the client. It wasn't until then did we understand that there surfaced some fourteen different open source licenses that were interplaying. And in a case like that, when you've got a broad range of licenses, some permissive, some restrictive, about the use of the software, about license grants, about what constitutes taint and what doesn't, there's a risk that your proprietary software component is no longer going to be proprietary but, in fact, it would be a derivative work subject to a grant.

¶53 And here, what we did for the client, after looking under the hood and studying the licenses and the interplay between the licenses and considering any patent rights that they may want to pursue, we drew a three-tiered architecture so that they could understand what's detailed in their application. In the middle, you have got a coded proprietary application that the client had prepared. We had to counsel the client to make sure not to compile statically any of the open source components into the proprietary work because if they did, there was a risk of taint. Taint would result. It would become a derivative work, and it would be subject to that license. Because they were using other products that were open source but were more permissive in their nature, we could counsel them to access the open source software dynamically as libraries. The difference between a static and a dynamic link is, ultimately, how the program is compiled. Basically, a program that is compiled statically is ultimately bundled all at once and executed all at once, whereas if you have compiled a software program dynamically you are having the program access referenced libraries only as needed, only when the program is actually executed, which means that, at times, a library may never be referenced.

¶54 So in that case, using more permissive licenses like the LGPL or Apache, we were able to counsel the client on how to structure the application so as to preserve as much as possible the proprietary work and also allow them to derive the benefits of using open source software. It's there. They don't have to take the time to code it. It's copy and paste. There are templates. They can be very valuable and more reliable, in many cases, than other products that are out there on the market. This is an example of how important it is to look under the hood and consider the interplay between licenses and also the interplay between proprietary works. A company is also going to be faced with the decision of whether it should release authored software under open source, and there are companies who do that and, indeed, they do it strategically. That is where the interplay of patent prosecution—that is procurement—and open source software licenses come in. Basically, we find that companies like Red Hat and Novell and Sun and Google do this. They will seek patent protection for their property and then make it available as open source. If a competitor comes along, like in the Jacobsen case, you not only have a copyright cause of action, but you have a patent cause of action, because you have potential infringement of your claims. That's how you can use it.

¶55 You can use it, basically, as a deterrent, not to mention the asset value that it builds in the corporation. You have a dual licensing option, right? Basically, you can say, "You're free to take the software. You're free to modify the software. But if you do, you have to covenant that you're going to take 'these measures'," whatever measures we set out. If you want a maintenance contract with the software, you're going to have to pay certain fees, and those certain fees are going to help fund the patent that we had to pay for.

¶56 Here's a simple example of a recent Red Hat patent application. I've been following it since '08. It just got a notice of allowance on Tuesday. They have a bunch

of others. Just to kind of wind down, I want to share a little bit of an illustrative IP strategy. Basically what we want to recommend when you consider the interplay between patents and software is make sure that you understand who the author is if you're going to seek to protect your software. You've got people in engineering who are working to put together a product in order to meet deadlines, and there are benchmarks, and there has to be auditing of that code. You've got to know what goes into the code and the source of the code, who certified the code, its signatures. You need to know about the code. You get auditing tracking software, and you get your attorneys involved as quickly as possible. You can patent that software if you are clear on that aspect—if it is clear that you are the owner. You can basically say by policy that you have instructed and you have verified that your employees are not using any open source materials, and they will put together the software for you, as you seek the patent, and then yourself to distribute it under open source licenses.

¶57 If you are concerned about publication initially, and you are going to do this in stages, you can file a nonpublication request in the Patent Office. At times, you may want to go the other route and file for early publication. You can do that as well. Then you release the software to open source, and you begin to build this defensive portfolio. You also want to remember that IP is broader than just patents. For goodness' sake, register your software with the Copyright Office. Initially, if you've filed the patent application and a nonpublication request or if you're not expected to publish for twenty-four months now—at the rate the Patent Office is going—you file it a trade secret, and you can redact, and you only need to submit, I think, the first twenty -five and the last twenty-five pages of the author's work, all right? Cheap. And that has worked very successfully for us and for our clients when it came to enjoining another firm, and then you can enforce your patent rights.

¶58 You have to consider the different open source licenses for the different strategies that are out there. There's no one-size-fits-all license. There are 40 to 50 dominant open source licenses out there. They're very different, often poorly written, and somewhat treacherous. You also want to be mindful that the patent law is evolving and looking very carefully at the question of patentability with respect to software patents. In particular, the In re Bilski case indicated—the Federal Circuit said yes, surely, software patents are patent-eligible. The subject matter is patent-eligible. And then the Supreme Court has been on record like in eBay, saying with Judge Kennedy that there sure are a whole lot of patents out there that to be quite invalid to us. So it will be interesting to see how that comes out. There's also some evolution in the area of obviousness and the written description. And with that, you have a very high-level overview about how open source and IP can interplay to the benefit of the client. Thank you.

¶59 MR. CHRISTIANSEN: I'm Jon Christiansen. I appreciate the invitation to be here. My comments should be taken with the understanding that I actually wear two hats as far as these comments are concerned. I think that the two hats reflect my experience and my bias with respect to the topic of open source due diligence. On the one hand, I'm still a practicing lawyer with an emphasis on computer and IT law, but I'm also a bit of an entrepreneur and have some businesses. One of them is EscrowTech, which can be accessed at www.escrowtech.com. EscrowTech originated as a source code and technology escrow company but in recent times has expanded into open source due diligence, so I've also seen this from that perspective as well. I think, obviously, the

situations for open source due diligence would involve anything, any situation where you need to know if a commercial software product has been contaminated—that's the word I use—contaminated with OS code. OS code is a term I use and others use for open source code. Some people call it free software or open source software. FOSS is also a term used. Actually, I think in my own mind, free software is ally a subset of open source software, but that's another topic.

¶60 But in terms of these situations in my own experience as a lawyer, I've encountered situations, such as acquisitions of a company, where the acquirer believes that the company has proprietary commercial software products, but they include open source code. As you've learned today, or you probably already knew, if those commercial software products are tainted or contaminated with certain kinds of open source code, specifically, those that might be governed by the GPL or other copyleft licenses, that could have a serious adverse effect on intellectual property rights with respect to what were believed to be proprietary commercial software products but, instead, might be "Freeware" subject to open source obligations.

¶61 Not only have I been involved in this in terms of acquisitions, but I've seen the need for open source due diligence in connection with investments. For example, an investor investing in a software company doesn't want to be surprised to find out that these commercial software products really aren't proprietary and commercial anymore, even in the context of a loan. This problem comes up in the acquisition of a company that has commercial software products. It can come up in the context of, say, a software development deal. So one simple example. I have a friend who was general counsel of a software company here in the U.S. The company outsourced the development of one of its commercial products to a software developer in India, but it could have anywhere in the world, even in the U.S. The developer, despite what was said in the contract, developed and delivered what was supposed to be a proprietary commercial software product, but the developer included some GPL code in the product. Within a few months or more after the product was introduced to the market here in the U.S., the actual copyright owners of that GPL code sued. It was a very unpleasant surprise for these folks, and the matter was settled, but it cost money to settle a claim that should never have arisen. This is just an example of a warranty (indicating). This warranty isn't necessarily good or bad, but as I said, I just quickly pulled it from a contract that involved the acquisition of a software company, as a "real world" example.

¶62 So if you represent a company being acquired or shareholders of the company being acquired, you may be expected to d behind or your client may be expected to stand behind a warranty of this nature that specifically addresses copy left or certain types of open source software. And that can put you, as the lawyer, in a very difficult situation. I have been in that situation many times. Maybe some of you have been in it many times. What do you do when you're confronted with something like that? How do you really know if you have a problem or not? In other words, can your client really make the warranty?

¶63 Well, I first started encountering this issue back in the mid '90s and then increasingly over the years, more and more. Nearly all acquisitions of software companies include agreements that have warranties of this nature. What do you do as far as due diligence is concerned? Well, what you want to do is identify the commercial software products that have OS code in them. Better yet, you'd like to identify the

specific files in each product that have OS code in them. So, as a given software product may have hundreds or thousands of files, and you'd like to know which of those files might include some OS code governed by an open source license. After identifying the potentially contaminated files, then the next step in due diligence is to identify what those open source licenses apply to the OS code in those files.

¶64 The next step after that is to look and see if there are special exceptions in those licenses. Now, under GPL Version 3, they're referred to as "additional permissions," but what I've found over the years is that you might find a program that's governed, say, by Version 2 of the GPL, but you might also discover that whoever originally released that particular code under the GPL added a little paragraph or two, a so-called "special exception."

¶65 And as John and Alfred already said, if you think that the open source licenses are poorly written, boy, some of these special exception things are just a nightmare to try to decipher and read. Although usually the intention of these special exceptions is to allow additional rights or to loosen up some of the restrictions under the GPL, you have to assess those so-called special exceptions on a case by case basis. And then, of course, the bottom line of the due diligence is to assess the impact of those various licenses on your client.

¶66 If your client is only interested in internal use, this is almost a non-issue. On the other end of the spectrum, though, if your client is interested in having software products that can be commercialized to others, the presence of OS code in the products can have a very serious adverse effect on intellectual property. As Alfred has pointed out, it can impact patent rights, and then, even more seriously, it can turn what you believe to be a commercial software product into Free Software.

¶67 The next step of due diligence sometimes gets to a point of remediation or disclosure and so on. Disclosure is an example of how the results of due diligence can be used. The way I've used it is, in typical transactions, if I'm representing the party making the warranty that there's no open source code in the commercial software products, I make those warranties subject to a disclosure of exceptions. You might have a disclosure schedule that discloses the results of the due diligence as an exception to the warranty.

¶68 I had a situation once where we were allowed three months after the agreements were signed and the transaction closed, to make a disclosure of exceptions to the warranty. I don't think, normally, you can expect to have that kind of leeway, but we did in that case. We did a very extensive due diligence, and I ended up sending to the attorneys on the other side a stack of papers about this thick (indicating) with all sorts of disclosures of open source code and other problems. I think they were a bit dismayed at what they received. But that was the deal, so I didn't hear any complaints.

¶69 Again what does due diligence consist of? Sometimes, it's nothing more than sitting down and interviewing the programmers. I don't know if any of you are computer programmers. I've done a little bit of computer programming myself. Two of my brothers are software engineers. Hopefully, no one takes this the wrong way, but communicating with computer programmers can sometimes be a little bit difficult, especially when you're talking about legal issues and intellectual property. Also, they can be very, very defensive in this context. In other words, they're programmers. They work for a company. They may have even been told, "Don't put any open source stuff in these products without permission." And then, they have to sit down with me or another

lawyer, and they get grilled. And so they can get very defensive about what they did. Just as a little side tip, in my experience, it's extremely important to get management on board with the concept of not making these programmers feel defensive. We just want to know what the facts are so we can prepare our disclosure statement which will be an exception to the warranty.

¶70 In addition to interviewing the developers and designers, talking to management, in-house legal counsel and others can sometimes yield interesting information. Document review can be part of this as well, although usually not as fruitful. Code review can also be important, and there is some technology out there to do it. There are technologies now that enable scanning of code to generate reports that will help you identify and find code from certain open source projects in the corresponding licenses. Some of the best known are Palamida, Black Duck and Protecode, but there are others like OpenLogic and so on. But in my case, my own any, EscrowTech, uses Palamida and Protecode technologies and some of our own proprietary expertise when we do the scanning as a service for attorneys and others.

¶71 But the scanning is the easy part. Briefly, when some of these technologies scan the code, they look for snippets of code or fingerprints that match up to a very extensive library of open source projects. The idea is, if these snippets or fingerprints trigger a match, which will be reported in the report for a given file, you will know that there's a potential match. There's usually a code rank which is somewhat a percentage measure of relevance and then an indication of what license or licenses might apply.

¶72 That's a good starting point, but then, sometimes, there are false positives, and it still takes some review of the reports and some analysis by IT and legal professionals to dig into the matter. And then sometimes after that, it's a matter of remediation. Remediation includes fixing the problem after the fact.

¶73 Just to give you a brief idea of the effectiveness of this, I will describe one due diligence examination that I conducted. The manual part came first. This involved interviewing the programmers. This was a massive software project. There were over a hundred programmers. Many were still in the organization at the time of due diligence, while others that were long gone.

¶74 In interviewing the programmers, we complied an extensive list of known uses of open source code in the software. That resulted in a long list. Then we did a scan of the software. Through the interviews, we only found about 50 percent of the problems. After the scan, we discovered many other open source problems. We also discovered that there was one programmer who had put code into the software with a copyright notice in the name of his own family trust. He didn't reveal it when I interviewed him or interviewed the group that he was in. It was strange that he would put code in a company's software product and still claim that it was proprietary to his own family trust. But, anyway, you never know what you're going to find. I can tell that our time is up, but I appreciate your attention to my comments. Thank you.

¶75 MR. RILEY: Thank you very much for your time. Thank you, once again, for attending the symposium today. We would also like to thank our sponsors, McDonnell Boehnen, Knobbe Martens, McAndrews Held and Goodwin Procter. Right now, we will be having our reception. That will be in Lowden Hall. If you just go directly out the door and to the right, we have people that will be there to direct you to the place, to our reception. Thank you once again for attending, and have a great day.

¶76 MR. BOETTICHER: Excuse me. I'd like to remind all of you to apply for CLE credits. I know you had such a great time that you don't think you really earned anything for this, but we have forms outside, and our staff at the table will answer any questions you have on that, and it helps support this symposium. Thank you very much.