Colorado Software Summit 2006 – Simon Phipps, webmink@sun.com, http://www.webmink.net/

Whose Freedom? Software Freedom and what we all mean by it

Good morning! I can't tell you proud I am to be invited back for my 8th year to speak to you all. I'm Simon Phipps, and my day job is oversight of the open source and Free software activities at Sun Microsystems. I do that job because of the huge amount of open source software that Sun works on. It may come as a surprise to hear, but Sun is responsible for vast amounts of Free software that people use every day. In fact, Sun created more than a quarter of the software in Debian, and is the largest creator of Free software by a factor of four. That number is about to soar even higher because we now have a policy to open source as close to all of our software as we can manage. The Java platform is next on my list and is consuming a great deal of time considering licensing, governance, hosting and more.

Whose Freedom?

I sat last night looking at all my slides from the past five years, and I realised that we have come a long way together since then. More importantly, I realised that many of you have seen all my slides before! So this year, I'm taking a different approach. No slides. Not even photos. Let's see if it works!

Wayne is very patient with me and graciously leaves me to set the subject for my talk here until the last minute. He contacted me a few weeks ago and asked for a title and I was stumped. I looked around my office and found a book that I'm reading, and "borrowed" the title. The book was "Whose Freedom?", by George Lakoff, a cognitive scientist whose work I greatly admire. His is the analysis of how language reflects thought and how it can also shape thought. His was the bestseller "Don't Think of an Elephant" which was a hurried booklet explaining the concept of "framing" in the context of the last US election, and "Whose Freedom" is a brilliant, readable and much more thoughtful revisiting of the same themes.

At the heart of the new book, Lakoff considers the ability of the educated mind to understand and accept complex systems – that is, networks of effects where the consequence at the end of the network is not obviously associated with the effect at the start of the network. Among his examples of complex systems are global warming - "how can my SUV possibly be melting the arctic". He points out that while some of us are willing to trust that a complex system like this actually exists, follow the rational explanation and choose to accept limits on our personal actions based on an acceptance they contribute to the complex system which defines freedom for all of us, others consider such limits an unwarranted restriction on their freedom. Lakoff extends this theory to help explain different political uses of the word "freedom" and how apparently rational people can use the word in ways that

other apparently rational people consider a tautology.

I know some of you read my blog and know my politics but don't panic, I'm not going there – although I would very much encourage progressive thinkers among you to read Lakoff's book. But I was struck by the way the word “freedom” is used in connection with software as well. At its roots, the open source software movement is about Freedom – that's where the phrase “Free Software” comes from, no matter how many times executives and CIOs assert it's about cutting costs.

Open Source Community Members

Now, to be clear, open source software is not primarily about cutting costs. Yes, there are ways it is producing a step-function in cost reduction for many people, but the savings are part of a one-time switch as the available margins on software are changed. I'm coming to the conclusion that there are three kinds of people engaged with open source. Those three categories are what I term co-developers – people co-operating over the development of a base software system; deployer-developers – people who assume the existence of the base system and either write applications that depend on it, or configure or script upon it with the goal of deployment; and end-users, people who take the work of the co-developers and the deployer-developers and use it in production to achieve some end. Obviously there are no solid lines between these three groups and they can overlap substantially, but the three groupings are useful because they help us understand dimensions of software freedom. Each of those three groups of people are likely to define “freedom” differently, and ignoring that fact lays at the root of many of the arguments I observe in the free and open source software communities.

First let's consider co-developers. Open source co-developers gather around a source-code “commons”, where a bunch of source code is maintained. They use the contents of that commons to produce derived software works that are of value to them for some reason – social, financial, political are the usual reasons, although there are some hobbyists too.

I'll take some time here to recap points I've made in previous years about the factors that matter to co-developers.

Licenses

Software licenses define a set of freedoms for the use of the copyright materials contained in the commons. Before the late 1990s, pretty much all software licenses were bespoke, written to define freedoms around a particular body of code. This caused a problem for software developers because, to understand the exact nature of the freedoms a license created, it was necessary either to engage

the services of a lawyer, or to trust the author of the license. To the individual co-developer, this was a problem because she could afford neither.

The Open Source Initiative addressed this problem by standardised licenses. By getting a community of license experts to review each license, the OSI Board used a benchmark – the Open Source Definition – the create a list of licenses believed by open source developers to create the freedoms necessary to make new software works in a way that did not damage the virtuous cycle of open source software development. This creates a delicious paradox – licensing without lawyers. OSI was very successful and today it would be unthinkable for a company to propose an unapproved license for an open source community. Open source licenses remain a fundamental issue for the freedom of programmers to create wealth. However, because of the success of OSI, they are now less of a problem, although proliferation has paradoxically introduced a new problem – needing a lawyer to help you pick an open source license. You may be interested in a white paper we've created that recounts the simpler process Sun is using – you can get a copy from sun.com/opensource.
The licensing landscape continues to evolve. For example, the Free Software Foundation is hard at work generating version 3 of the GNU General Public License, a process which I believe holds great hope for the wider open source communities but which presents some challenges for the Linux kernel and to software tightly-coupled to it. I'm watching this process closely and I'd be happy to discuss this later if anyone is interested.

Governance

The license on the code commons gives them the freedom to develop derived works, and their experience doing so results in them improving the contents of the commons. They need access to the commons to commit changes back, so the co-developers need to agree rules for who can commit, when, how, and so on – rules I describe as “governance” and which increasingly others are too. I regard open source community governance as the most important aspect of a community of co-developers. Governance defines the freedom of co-developers to commit code back and thus collaborate with each other. Thus, for co-developers, “freedom” means two immediate things – the freedom to create derivative works, and the freedom to contribute back improvements. And it means a more abstract thing for most of them too, the freedom to create something of value from the commons. No open source community of any value would allow just anyone to insert their code into the commons around which the community has gathered. They all have rules about who can do that – who can get commit access, who decides who gets that access, the scope of the access, and so on. More than that, every community has rules about how other decisions get made – who gets to vote, who gets to decide on infrastructure, how the antisocial are disciplined, and so on. It turns out

that today, this is a far more important area of focus than licensing, and to this degree I agree with Tim. Licensing is largely a done deal, but governance is still a problem.

After all, having commit access to the commons is at the heart of many of the business models that members of open source communities use to make a living. You can only promise a big corporate customer that you'll fix bugs or add features if you have some way to ensure the bugs stay fixed at the next release and the features don't need re-implementing. Sure, you could always take the code and start your own community if you can't get commit access, but that is a big deal and you're not likely to succeed unless you have the resources of a big corporation or you have a genuine grievance about the existing community with which others agree enough to swap projects and join you. In practice, you need the governance of the existing community to work right.

The thing is, we have no benchmark for governance. It's easy enough to see when governance is diseased – when all the committers work for the same company, for example, or when commit rights are arbitrarily revoked, perhaps on a job change. There are communities where honest debate has turned into obstructive argumentativeness. And there are more signs of disease that I'm learning to spot. But there's no benchmark for governance, and we need one. The governance of a community like the Apache Software Foundation gives us an exemplar of good practice, but we need an Open Governance Definition against which we can compare the formal or informal rules of a community with which we're considering aligning ourselves so we can tell if it has governance likely to promote freedom.

Deployer-Developers

Now, having considered in some detail the world of co-developers, let's consider my second category, deployer-developers. They use the result of the open source community, often gathering around it as an outer layer. For them, access to the deployable result of the co-developer community is key. We were discussing Maven in the Q & A last night. Most of us have little interest in being co-developers of Maven, but many of us choose to deploy it and to develop resources that make that deployment useful to us. In the process we may well report the occasional bug, maybe even offer a fix or two, but we remain firmly deployer-developers. For deployer-developers, the freedom to use the deployable software is the key freedom.

End-Users

Finally, end-users. For them, the key is usually reducing risks and costs. They have no interest in being co-developers – indeed, they consider the maintenance of thousands of lines of code to be a

risk, not an opportunity – and they also consider deployer-developers to be a source of cost and risk, albeit a necessary one. For them, software freedom is not about source code or even mainly about access to the deployable software. Rather, it's about controlling risks and costs, not least through maintaining flexibility and independence from vendor control.

One way to look at the change that's happening to end users in this regard is to consider the switch that's happening into Software 3.0. When I started in the computer industry, software was typically included with the computer when you purchased it. That was Software 1.0 – the world of bundled software, where developers were paid as part of the hardware purchase – monetisation at the point of system purchase.

Around 1980, there was a shift and the software was unbundled from systems, creating the world of Software 2.0. Software makers charged for the right to use the software – monetisation at the point of software selection. This shift in models also changed the cycles of software development, forcing artificial “upgrades” and release cycles of 12 -18 months aimed at forcing further right-to-use revenue.

What's happening now is that end-users are gradually getting what they actually want, liberated by the freedoms of open source. Their deployer-developers get hold of whatever software they need without charge from co-developer communities and assemble solutions for the end-users that employ them. Those end-users then seek out the resources they need to put and keep the solutions in production. While everything in open source is free of charge to someone, nothing is free of charge to everyone, and end-users are willing to pay both deployer-developers and co-developers whatever it takes for the value elements that aren't available in-house. Those could be update streams assembled from individual bug fixes, support services, education, consulting, systems administration, maybe for large companies legal indemnity. This is Software 3.0, where monetisation happens at the point of deployment, at the point of value, and all open source business models are the delivery of bundles of deployment-time value.

Freedom to Leave

In other words, what end-users value is the freedom from external control that open source brings them, allowing them to pay for value they need. Only it's not just open source that does that. Open source software may bring freedom to deployer-developers but actually it can become a control point for them over end-users. What most end-users need to bring them software freedom is not just open source software but also truly open standards. More subtly, this is also important for co-developers and deployer-developers. Having source code open for use to create wealth is an important springboard to software freedom. Having governance that promotes a transparent meritocracy is

another. But the data formats, protocols and interfaces that the software exposes or uses can trump all that.

A journalist asked me an interesting question recently. "Why is it," he asked, "that we are seeing so many new desktop and Web 2.0 tools?" He's right, of course. There's loads of energy around, with projects like **Google Calendar**, **KOffice**, **OpenOffice.org**, **Chandler** and plenty more, plus new innovations like **Backpack**, **Writely**, **BlogBridge** and others. I've tried many of these and some of them have stuck for me. What's the connection between them?
The thing is, all of these tools have worked out that *lock-in is the new lock-out* – my third koan. The fastest way to send early adopters packing is to make your cool new toy a **roach motel**. To start with, early adopters like me are not willing to put live data into applications that don't offer import and export. My calendar is in **RFC2445 iCalendar format**, so if you want me to try your new calendar thing you'd better accept that as the import format. If I can't add iCalendar and **vCalendar** appointments I'll not be using it for long because that's what other send me, and there had better be an iCalendar sharing facility for scheduling.

What's more, I have to have iCalendar export so I can migrate away from your new toy to things like Apple iCal, **Sun Java System Calendar Server** or any of the umpteen programs that support those standards. The same goes for everything else - I just moved my blog subscriptions from **Bloglines** to **BlogBridge** to give it a try using the OPML export in Bloglines (it's a pretty cool Java Webstart application), and I work with a group of people routinely exchanging documents between a selection of applications that support OpenDocument Format, which has recently been ratified as ISO 26300. The availability of open, freely-usable standards for formats, protocols and interfaces creates a bigger market and promotes innovation because we are all free to give things a try. If "interoperability" meant "import only" or “play only in my space”, I'd never feel safe trying new things so market growth and innovation would be inhibited. People who implement open standards like this are smart, because although they allow customers to leave for greener pastures they also allow them to return - I am still using Bloglines despite the appeal of BlogBridge - and the confidence I feel over "owning" my data makes me a much more interesting customer.

That feeling is caused by more than interoperability - it takes full *substitutability* for me to have the confidence to stay as well as the freedom to leave. That's why **Stewart Butterworth is spot-on with Flickr's policy** and paradoxically will keep my business by allowing me to leave at any time. That's my fifth koan – staying because of the freedom to leave.

More than that, though, full support for truly open standards means that new ways to use the data can

occur. For example, the feeds in Bloglines mean I can use BlogBridge to read the for:webmink tag feed using BlogBridge and **have others send me interesting links** to read without the overhead of e-mail. That's part innovation-by-design and part innovation-enabled, leaving the customer to work out new ways to mash-up the data and create innovative uses for their own stuff. When using the data demands only a particular vendor's software, or a licensing relationship, or some other boundary traversal, the innovation finds it harder to escape.

So what does it take to have a standard that leads to substitutability and the freedom to leave? At a minimum, freedom-promoting standards will have three characteristics.
First of all, it takes confidence over intellectual property rights. I dream of a world where "standard" implies that all parties to the creation of the specification have been compelled to issue **non-assert covenants** so every developer can be sure there's no strings attached.
Second, it takes multiple implementations, proving the format is actually usable in multiple places. This was the genius of IETF and it's one of **the lessons of CORBA**. There have to be at least two substitutable implementations, and at least one of those has to be open source.
Third, the approach must not favour any particular implementation or platform. That's the problem Microsoft's Office 12 XML format **turns out to have**, and no amount of rubber-stamping by Ecma International will fix it.

To be sure they don't have more subtle barriers to freedom, I look for two more characteristics:

Number four, the standard should have been created transparently. Just as an open source community looks with concern on a large, monolithic code contribution, so we should be wary of standards created without the opportunity for everyone to participate or, failing that, with a full explanation of every decision that was made in its construction. Without that there's a chance that it's designed to mesh with some facility or product that will be used to remove our freedom later.

Number five, the standard should be in the care of a transparent and inclusive stewarding organisation. Without that the successor format, protocol or interface will just appear from nowhere unexpectedly and we risk losing the wealth we created if it becomes the required way to work.

This is about far more than interoperability. Interoperability was a fine goal in the 90s, but in today's world it takes much more than just the minimum level of allowing others to use your secret sauce. Pragmatic interoperability is better than nothing and certainly beats the cold-war mindset of the 80s and 90s where incompatibility and isolationism were the rule. But I want more than import-only. I want more than lowest-common-denominator exchange, where I have to rework my data to make it survive

the teleport. Those are the hallmarks of the monopolist's definition of interoperability - letting me play in your market at little risk to your monopoly.

The new world is being made by iCalendar, Atom/RSS, OpenDocument, OPML and the other new standards of Web 2.0 and the new desktop. Truly open formats are creating the new market, and those who attempt to subvert the trend with pseudo-openness will fail. But we need a benchmark against which to measure them. We need an Open Data Definition that can help us tell whether the formats, protocols and interfaces we encounter promote freedom.

Control -> Influence

I've used the word “freedom” in a large number of ways here. Ultimately, there's no one-size-fits all definition. But I've noticed that the positive associations of “freedom” in many cases signal a shift from overt control to the exercise of influence.

We need to be on the look-out for control strategies embedded in open source. There are plenty around – the way the MySQL approach prevents creation of a co-developer community for example. These are not always absolute, as Red Hat is discovering this morning with Oracle using red hat's own lock-in weapon against them. They are also often subtle – for example, the influence mechanisms surrounding Eclipse are an art form – but have no doubt that plenty of players want to move beyond the bounds of acceptable influence to guarantee that key competitors are restrained or key customers locked in.

The shift to an influence economy seems to be an important pattern for the positive changes we see around open source, but it's also an opportunity for those addicted to control to seek to “game” the system. After all, as a wise philosopher said, with every system comes the game that plays it. The price of freedom is eternal vigilance, and as we move further into the world of Software 3.0, of open formats and of open source software, I implore you to be engaged, vigilant, informed and vocal about the processes shaping your professional world.