Will Open Source Developers be Well Paid?

July 5, 2007
By James Maguire

By his own description, Dirk Riehle is a major fan of open source software. Riehle, leader of the open source research group at SAP Labs in Palo Alto, California, spends countless hours theorizing about the economics of this emerging software trend.

He's the author of The Economic Motivation of Open Source Software, a scholarly article that describes significant shifts in the software business. *Datamation* spoke with him about how these shifts are shaping the pay levels of open source developers.

To understand Riehle's thesis, you have to understand how he uses the term "Committer." A Committer in an open source project is a leading developer, someone who shapes the growth of that project, in contrast to regular developers, who work as contributors. However, Riehle notes, the line between these two positions – the level of influence and responsibility – is far from clear.

"In open source there's not yet a clear distinction between product management and development," he says. "A lot of product management is actually done by the developers, because they are the folks who do it, so they decide their way."

Still, being a Committer matters, Riehle says. "If you are a developer at the Committer status, for an important open source project, you're likely to have a higher salary. That's anecdotal evidence [it comes from a study that hasn't been fully published]. According to that study they empirically verified that. But until I fully see that, it's anecdotal."

Q: You write in your paper that, "for software developers, life has become more difficult and exciting at once." In what sense is it more difficult?

"If you take a labor economics perspective, from the employer perspective it's easier to get developers who are familiar with the [open source] software you're using. Developers might have seen it in a previous job, or when they were students. Because it's open source software, it's freely available.

So I would expect – and again this has to be empirically verified – that because of this increased interchangeability [of open source programmers], that regular salaries might take a dip. I don't know how big it is, if it would be a significant dip, but I would expect it to be visible."

Q: So you're saying that, because knowledge of the underlying source code is less rare, workers are paid less?

"It's about the power between the employer and the employee. The employer is always in a stronger position, of course. But if they can draw on an even larger worker pool than before, who can get up to speed faster, the little advantage that a regular developer has – which is knowing the employer's products really well, while the outside person doesn't – that little advantage gets diluted. And that pushes down salary.

"I assume there would be counterbalancing forces. But from a labor economics perspective, I would expect that high-powered positions in visible open source projects would get an uptick, and the average developer would get a downtick."

Q: Can you talk about the role of the Committer?

"You need to look at how, in the future, or increasingly, software companies develop software. Which involves, almost certainly, incorporating open source components. There will be stuff that will always be proprietary, because, for the typical software vendor, there must be a competitive differentiator. But small parts, or large parts, could or should be open source.

"These open source components will be more important or less important. For those that are more important, [software vendors] will want to have some influence on the direction of that component – because it's part of their product. If it goes a wholly different way than from what they need it for, they might be in trouble.

"You can analyze, for example, the The Eclipse project. You can see how companies that incorporate certain components of the Eclipse platform are trying to get their developers in there as Committers, who are those folks who have a stronger say in what gets done next."

Q: Let's talk about the economic benefits of working on a *community* open source project vs. working on a *commercial* open source project. You write in your paper that it makes little sense for the economically rational software developer to invest time in commercial open source. Why?

"It's always a balancing act. In this paper I'm talking about rational economic models, and in reality there are of course always more factors. [But] If there is a community open source project that is comparable with a commercial open source project, the time investment that pays off – knowing this software, being known in the community – the investment you make will give a higher return if you invest in the community project. Because you take the reputation and influence that you create through your investment – you can take that with you, it's *your* increased value. The community open source project is not owned by a single company.

"Now, a commercial open source project is owned by a single company. So if you invest in that, even if you're an employee of that company, you can't take that reputation with you as easily as you can with the community open source project. It generates less value for you if you work on a commercial open source project. If you do this as an employee, the rights to the software and position that's given to you can be taken away by the owning company.

"Now, if there's no viable competitor to the commercial project, it may not make sense to invest in a community project. But if you have a choice between a viable community project, and a comparable commercial project, I would always choose the community project as an individual programmer."

Q: You write that, "Open source reinforces the trend toward employees becoming 'free agents.'" What do you mean by this?

"With 'free agents' I don't necessarily mean that they would be freelancers. They will be employed.

“If you are a high profile Committer to [for example, community project] Apache, then your reputation is with the community. Your value is based largely on how you can influence and contribute to and have knowledge about this particular product. And that product is not owned by an employer.

“So you can move on to the next employer, you can even be a freelancer. The way you think about what you do is not as strongly coupled to an employer as it was in the old closed source, proprietary source [world]. There, what you were worth to your employer was tied to quite some respect to your knowledge of their product. And you couldn’t take it with you as easily to another place.”

Q: But community open source software is distributed for free. How can working on such a project be lucrative for a programmer?

“You can observe this as a strong trend: People work on company time, full time, for something that is open source, under a license that allows free distribution. So the company that pays the developer is not making any money directly off of that open source component.

“Now this sounds like it’s Communist, or something. The point here is that the company is likely to have figured out a business model that is obviously not selling this software, but is something else, like providing consulting around this product, or having extensions, additional add-ons, or building a complex product around the open source component. And for the company to stay in business, and not be steered to the side, they really need to ensure their influence on their open source components. This is why they have to employ those folks – or some of those folks – who actually work on it.

“So PostgreSQL is a good example. There’s a company, EnterpriseDB. They provide all kinds of additional enterprise readiness features around the open source PostgreSQL community. Of the core team of 7 Commiters, EnterpriseDB employs 3, who work full time on the community product, and thereby benefit the community and not directly EnterpriseDB the company.

“However, because they work at this company, they have the knowledge of how to do proprietary extensions, and to make sure that nothing happens to the PostgreSQL community product that would be strongly against the desire of that company. So the company ensures some influence on the community product.

“It’s never control – it’s important that these are much more subtle influences and discussions that are going on, and in some sense are transparent and open and fair to the community. But because this company has a stake, and is making a valuable contribution, the community recognizes that it should not act against this company in a willful or bad way.

“So it’s a give and take, and I think it’s well understood in non-trivial open source projects.”

Q: If you were to advise a young person to choose a career direction based on income potential, would you advise them to become an open source or a closed source programmer?

“It’s not decidable, it’s not the right criteria for a choice. [Instead] the factors are how significant that piece of software is that you’re working on, and then, for yourself, at what point in time do you come in.

"If you can come in early and you get a Commiter's position easily, always go for the open source. But if you're joining a very successful commercial proprietary company – and why not? it could pay well – the knowledge you have could be worth a fair amount to that company.

"You need to look at the significance of the software – I think that probably trumps most of the things. And then, your chances of getting a powerful position with that software or project of company" is the other major deciding factor.

James Maguire is the managing editor of Datamation.