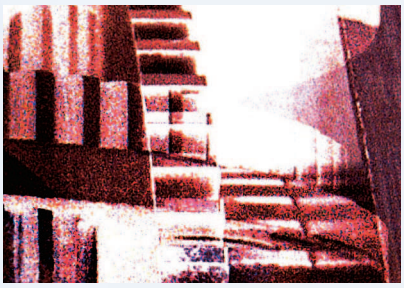
The Economic Motivation of Open Source Software: Stakeholder Perspectives

Open source software has changed the rules of the game, significantly impacting the economic behavior of stakeholders in the software ecosystem. In this new environment, developers strive to be committers, vendors feel pressure to produce open source products, and system integrators anticipate boosting profits.



Dirk Riehle
SAP Research

The advent of open source software has produced more than lower software costs for users. It has also caused major changes in the economic interaction among players in the software ecosystem. For many, open source embodies a specific approach to software development—even a lifestyle. But it's also sound business strategy. Ron Goldman and Richard Gabriel suggest that companies should use open source software to grow their user communities and build an ecosystem around their products and services.[1]

Open source software is typically free and comes with the source code needed to adapt it to users' needs. Most open source licenses let users redistribute the software, including possible changes, and charge for redistribution as long as source code changes are publicly available (www.opensource.org).

There are two types of open source software. *Community open source* is software that a community develops. Rather than a single corporate entity owning the software, a sometimes broad community of volunteers determines which contributions are accepted into the source code base and where the software is headed. Individual developers, the *committers*, and not a specific company, make decisions about the software, as in the case of the Apache Web server (httpd.apache.org).

Commercial open source is software that a for-profit entity owns and develops. The company maintains the copyright and determines what is accepted into the software code base and what to implement next, as in the case of MySQL and its MySQL database (www.mysql.com).

Prior work on community open source economics focused mostly on labor economics, that is, the frequently surprising amount of volunteer work that goes into open source software. Eric Raymond notes that developers contribute to open source projects for the personal gratification that comes from increasing their reputation among peers.[2] Ernan Haruvy and his colleagues reached similar conclusions in their empirical study.[3]

Joshua Lerner and Jean Tirole, meanwhile, argue that developers contribute to document their technical capabilities and improve job prospects with future employers.[4] And Karim R. Lakhani and Robert G. Wolf report that enjoying their work is a key intrinsic motivation for developers to contribute to open

0018-9162/07/$25.00 © 2007 IEEE

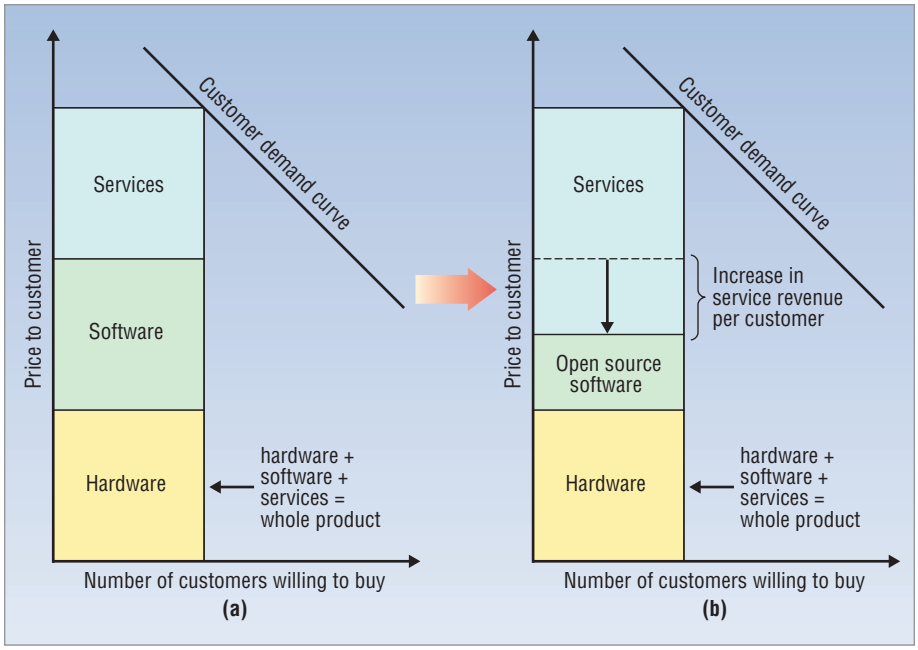


Figure 1. IT solutions demand curve. (a) System integrators sell a stack of hardware, software, and services. (b) Integrators can charge customers similar prices even if they use open source software.

source projects, although survey respondents also revealed that financial incentives are important.[5]

While this explains some of the volunteer work, it doesn't explain why companies today employ people who contribute to open source projects on company time. Il-Horn Hann and colleagues found that the salaries of Apache Software Foundation project contributors correlated positively with the contributor's rank in the Apache organization.[6] They therefore concluded that employers use a developer's rank within the foundation as a measure of productive capabilities.

SYSTEM INTEGRATOR PERSPECTIVE

Large system integrators, or solution providers, stand to gain the most from open source software because they increase profits through direct cost savings and the ability to reach more customers through improved pricing flexibility. Every dollar a system integrator saves on license costs paid to a software firm is a dollar gained that the customer might spend on services.

IT solutions demand curve

Customers typically want information technology providers to deliver "solutions." A solution solves a customer's IT problem, freeing the customer to focus on business rather than IT. A comprehensive solution comprises hardware, software, and services. Indeed, the IT industry earns its living by removing or reducing customers' IT worries.

System integrators deliver solutions by selling a stack of hardware, software, and services as one product. That allows the customer to talk to one company, rather than many. Figure 1a illustrates this stack together with the customer demand curve.

The demand curve shows how many customers are willing to buy the system integrator's solution at a given price. On the *y*-axis is the customer's cost to purchase a solution, and on the *x*-axis is the number of customers who are willing to pay for that solution at the given price. The form of the demand curve varies depending on what is being sold. However, in general, the demand curve is downward sloping: The lower the price, the more customers are willing to buy.

A system integrator's profits depend on which of the stack's components it owns and which it must buy. Usually, a system integrator's stronghold is services, which puts together the hardware and software pieces to meet the customer's need. However, if the system integrator owns only the services component, it will have to pay other companies for the software and the hardware and thereby share revenue, leaving less profit for itself.

It's therefore in a system integrator's interest to acquire hardware and software as cheaply as possible. Open source software, if an option, is typically much cheaper than closed source software, hence its use increases profits for the system integrator.

Figure 1b illustrates how with stable supply and demand, more money is made in the services part of the value stack if software cost goes down.

Software cost savings aren't easily passed on to customers for two reasons: First, customers tend to care about the whole product rather than individual components; second, large system-integration projects are complex and new competition doesn't spring up easily. Thus, system integrators can maintain their prices.

While this is one good reason for system integrators to support open source software, there's another equally compelling reason for them to support and contribute to open source software.

Business growth

The simple value stack that Figure 1 illustrates suggests that system integrators charge customers only one price. In reality, the system integrator can charge varying prices for a total solution to a prospective

customer's problem. Only one thing is certain: A system integrator will want to at least cover its costs.

The price charged per customer that Figure 1 shows can be split into the system integrator's service cost, plus the markup or margin needed to make a profit. If the system integrator owns just the services part of the stack, the cost for providing that service defines the lower price limit for the work. In a reasonably competitive market, the system integrator will accept all deals above this limit if it has the resources.

This limit, together with the demand curve, determines the maximum number of customers the system integrator can sell to and take on, as Figure 2a illustrates.

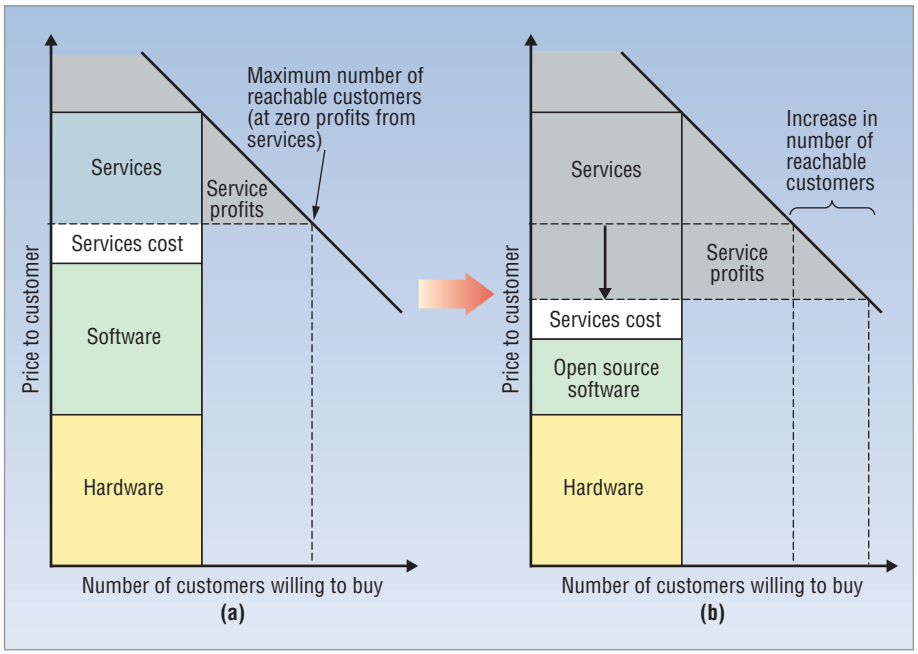


Figure 2. Sales margins and number of customers. (a) The lower price limit determines the customers the system integrator takes on. (b) Switching from closed source software to open source software can result in more customers and higher profits.

Switching from more expensive closed source software to less expensive open source software increases the profits of a sale through the money saved on the software. It also reduces the lower price limit for possible deals and puts a new set of more price-sensitive customers within reach. Not only does open source software improve profits on the original individual sales, it also increases the total number of potential customers.

Figure 2b shows how a switch from closed source to open source software results in more potential customers. And more potential—and presumably satisfied—customers mean higher sales and profits. The total profit is represented as the area of the gray triangle under the demand curve, showing the increase in profits when moving down the curve. Since in reality a system integrator might own many of a total solution's components, including software and hardware, more customers mean more profits through these components as well.

Pressures in the IT value stack

If it were up to the system integrators, all software would be free (unless they had a major stake in a particular component). Then, all software license revenue would become services revenue. To this end, I believe that system integrators prefer community open source over commercial open source. Only community open source software prevents vendor lock-in.

Community open source ensures that prices for software support are subject to market forces rather than one owning corporation. Community open source is a strategic weapon for system integrators to squeeze out proprietary as well as commercial open source software vendors.

SOFTWARE VENDOR PERSPECTIVE

Independent software vendors provide only a few software products, sometimes just specializing in one. Understanding the independent software firm strategy requires comparing open source software and closed source software cost and pricing.

Software cost and pricing

Why is open source software typically much cheaper than closed source software? In a closed source business, most of the investment in new software comes from shipping the first copy. The initial investment is recouped with increasing sales. The additional cost of producing and selling another copy is small, consisting of producing another CD or allowing for another download plus providing the (frequently minimal) free support that comes with a software license.

As the number of copies sold increases, the average cost per copy declines and the profits rise. Figure 3a illustrates the long-run average cost curve for a single software product.

The more mature a market for a specific software component, the higher the investment in the existing products, the higher the barrier to entering this market, the more established the existing players, and the more stable the price for the software component. A common scenario is "the 800-pound gorilla" firm that dominates a market and is surrounded by smaller players catering to market niches.[7]

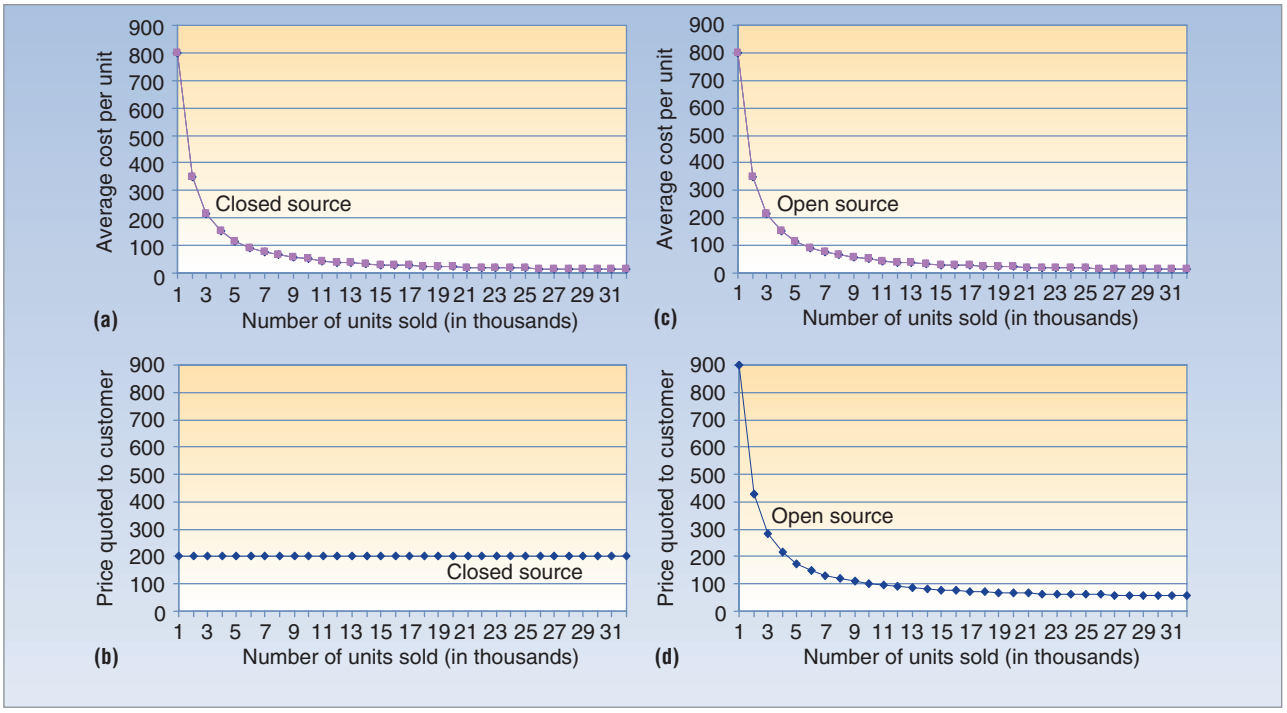


Figure 3. Differences in cost and pricing. (a) and (c) show a similar average cost per unit for closed source and open source software, while (b) and (d) indicate the relationship between price and number of units sold for open and closed software.

In such a market, the leading software vendor sets a price that maximizes its profits. Since the market is fairly transparent, the vendor can set just one pricing schedule, offering the product to different customers at the same price. (This is in contrast to the highly individual deals of large system integrators.) The result is frequently a flat price, as Figure 3b shows. Remarkably, the price of the proprietary closed source software doesn't directly depend on the actual cost incurred to develop, maintain, and provide the software.

The profit-maximizing price is largely independent of cost; the cost provides only a lower limit. Competition that drives prices closer to cost can't spring up easily due to the large initial investment a software product requires.

In a community open source situation, no such market-entry barriers exist. Given the right license, anyone can set up a company and start selling software. What the company will sell, of course, isn't the software itself, but its provision, maintenance, and support.

Because anyone can enter an attractive open source market, competition is fierce, and pricing will be based on markup over cost. If the markup is too high, new companies will enter the market; if it's too low, companies will leave the market. Moreover, the more mature the product, the lower the overall price.

Figure 3c shows the *total cost* of developing open source software. The total cost and the resulting *average cost per copy sold* is mostly the same as for the closed source solution. The main difference, of course, is that the different contributing companies now share this cost.

Figure 3d illustrates the pricing of open source software from a single firm's perspective. Because of the competition, the price charged for providing the software plus support is based on markup over cost. (The graph merging various dimensions into one 2D graph simplifies the situation, although the basic argument holds.)

Different firms will have different costs depending on their share of contributions to the open source project. However, with increasing project share, the company can charge higher prices because customers are likely to receive better service. The basic relationship remains unaffected: Price is markup over cost and varies depending on cost.

Customers love this situation because prices are substantially lower than in the closed source situation. System integrators love the situation even more because they can squeeze out proprietary closed source software.

Generating software profits

A system integrator can increase its profits if it reduces the software cost. By reducing software cost, it can move down the demand curve and sell to more customers.

Closed source software is the main obstacle to doing this: It cuts into profits on an individual sale and reduces overall pricing flexibility. Hence, system integrators have a high interest in turning closed source software markets into software markets with at least one viable community open source product.

Before the advent of open source software, entering an established and well-defended market was a risky proposition. With increasingly well-understood open source processes, setting up an open source project competing with an established closed source market leader's product is much less risky and carries a significantly higher chance of success than before. But it's not just a specific system integrator that will want to do that. It's pretty much everyone who isn't the closed source market leader.

To understand this, put yourself in the shoes of the CEO of an also-running traditional closed source company. At one point, it's becoming clear that you won't be the leading firm in your market and that your company's profits will come from market niches at best. Neither you nor your investors are happy with the projected return on investment.

Open source software has a high chance of taking new markets early on.

Your best option now is to open source your product. You might be reducing the market's overall return on investment, but at least you'll have a second chance at satisfying your own investors by making your company a successful open source business. You'll be in good company. With a proper license for your open source product, you might well receive help from the system integrators, customers, and software vendors higher in the IT stack.

Now, assume you're the CEO of the market-leading company in some space. Thinking ahead, you have to assume that either a competitor will open source its product or that a system integrator will instigate an open source product—or both. The proactive answer to this scenario is to open source your product, even though you're the market leader and would win big in the old closed source world. But it's better to win in an open source world than not to win at all.

These two thought experiments show that commercial open source software has a high chance of taking new markets early on. Only strong intellectual property protection or other competitive advantages might lead a closed source company to win and keep a new market. Leaders in established markets might be able to defend their positions for a long time. They tend to dig in with complex products, established processes, customer data lock-in, and many other positional advantages. Still, open source might well prove to be disruptive enough to conquer even these markets.[8]

Commercial open source

With such gloomy prospects for closed source businesses, independent software vendors have sought business models for harnessing open source software's benefits while gathering some of the profits of a closed source business. The answer is corporate open source.

Strictly speaking, commercial open source is a misnomer because community open source can be commercial as well. The key differentiator is whether a community or a single entity like a corporation holds the power to make decisions about the project.

Commercial open source software is typically available for free to nonprofit users. Sometimes commercial use is free as well. Usually, companies make money by providing support services. Sometimes they make additional money by selling proprietary software enhancements.

Like community open source, commercial open source is available in source code form. Unlike community open source, however, one company controls commercial open source. This way, commercial open source software can gather some of the benefits of community open source: faster adoption, free and speedy user feedback, and possibly volunteers' code contributions. This approach is mostly a marketing strategy, however, because the company that owns the software still must do the development. Hence, the company must employ and pay the software's developers.

During the early days of an open source project, this is an advantage, as the company can provide clear direction and muster more resources than community open source projects typically can. As the project matures, this can turn into a disadvantage, as a competing community open source project might have more resources at hand in the form of volunteers.

The upside for the owning company is that little open source competition can spring up for its product. However, system integrators have a strong interest in providing alternatives to proprietary software, and this applies to commercial open source as well. Hence, this business model is likely to experience the same pressures as proprietary software.

Open source service firms

If it isn't possible to be a profitable closed source business, what does it mean to be a successful open source business? The market's answer is the *open source service company*, which comes in at least two kinds.[9] One provides first-level support and implementation services; the other provides second-level support, training, and development services.

Clients of the first kind of firm are typically IT users who employ the firm's services to put the open source product into place in their IT operations. Clients of the second kind of firm typically need to get trained on the product or need to have a technical problem fixed that they can't handle themselves.

The strength of a service business usually lies in its ability to

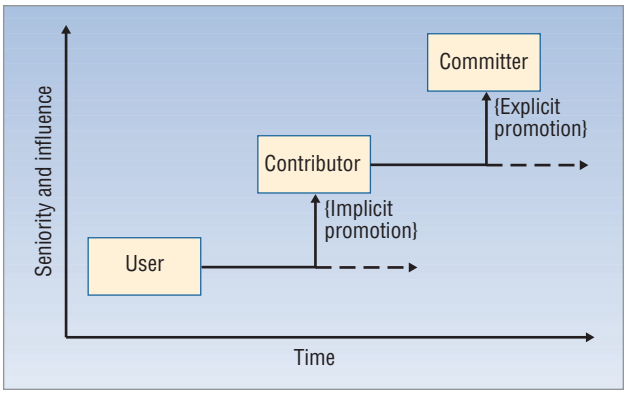


Figure 4. Positions and promotions. In open source projects, users are implicitly promoted to contributor status, while contributors are explicitly promoted to committer status.

- recruit and retain the right people,
- reliably set up and execute specific service processes, and
- bring to bear expert domain knowledge and unique intellectual property.

In the open source situation, this is usually labor economics. Technical skills around the open source product are a key part of determining an employee's value to a firm. Anyone who's smart enough can develop these skills because the open source software is available to people outside the firm.

Hiring and firing becomes easier because there's a larger labor pool to draw from, and switching costs between employees are lower compared with the closed source situation. Given the natural imbalance between employers and employees, this aspect of open source is likely to increase competition for jobs and drive down salaries. Lower salaries aren't as much of an advantage to the software vendor as might be expected because in the more transparent and competitive open source situation, such cost savings are likely to be (at least partially) passed on to customers.

The need for committers

An employee's position in the open source project is another key part of his or her value to a firm. The organizational setup varies between open source projects, but in some form, people always play user, contributor, and committer roles. Users use the software, contributors contribute in some form, and committers decide what contributions to accept into the project.

Figure 4 illustrates how a developer might progress through the ranks of a community open source project: A committer typically promotes a user to a contributor role implicitly by accepting the user's contribution into the software. A contributor is typically promoted to a committer position explicitly, through a prior vote of the existing group of committers and a subsequent public announcement of the contributor's ascension to committer status.

Committers determine where the open source project is headed, strategically and on a day-to-day basis. They can typically resolve technical problems faster than noncommitters, and have high visibility to the user community. Most projects are set up so there's only a small inner circle of committers, a larger set of contributors, and an even larger user community.

For an employer, the value of employing a committer is manifold. Through the committer, an employer:

- gets problems with the open source software fixed faster and better,
- can better align company strategy with the open source project and vice versa,
- appears as a more attractive employer than competitors who don't employ a committer, and
- has higher visibility with the user community and can reach out more effectively.

A major goal of any open source service company is to convert freeloading users into paying customers. A committer's visibility with the user community is an important marketing advantage that an employer can use to support this goal.

Thus, committers have a strong negotiation position with their employers. Employing a committer is important for a first-level support and implementation services company, and it's critical for a second-level support service company.

THE EMPLOYEE PERSPECTIVE

Open source software and service businesses make life more complicated for employees. Employees build up less firm-specific knowledge simply because there's less of it. People from the outside can replace them more easily. At the same time, an employee's day-to-day work improves non-firm-specific knowledge of an open source project that can be taken to another employer. So a developer who is fired can find a job faster than before.

Benefits of being a committer

An employee who is a committer is likely to earn higher compensation. Hann and colleagues have empirically verified this for committers to Apache Software Foundation projects.[6]

At any time, the committer-employee can credibly threaten to leave the company, taking significant power and reputation away from the current employer. Employers often pay premium salaries just to employ prominent committers.

But how do you become a committer? Community open source projects tend to be meritocracies, judging developers by their social and technical contributions.

In contrast, a company owning some commercial open source gives committer status to its employees (and takes it away) as it sees fit.

Consequently, it makes little sense for the economically rational software developer to invest time in commercial open source. The value these developers create is tied to the product and the owning company. Unless the product is in wide use or the developer wants to work only for this one company, it makes more sense to invest time in a community open source project.

How to become a committer

Developers who start projects immediately become committers. However, they now face the task of creating a successful project out of nothing. This is a highly entrepreneurial activity: Developers must promote their project while doing the actual programming work, understanding that the outcome is uncertain.

The window of opportunity is small for those aspiring to achieve committer status in an important open source project.

It's more common to join an existing open source project. Assuming a fair and transparent promotion process, the two main criteria that will get a developer promoted from contributor to committer are

- the developer's social and technical abilities, and
- demonstrated commitment to the open source project.

This is what most project Web sites state and what developer surveys have revealed.[2,3,5] However, these surveys rely on what developers say they do, and actual behavior could vary from what people believe motivates them.

Because being a committer can have clear financial benefits, keeping the group of committers small is in the economic interest of a committer to a successful project. Not doing so would dilute the committer's value to current and future employers. At least this would be an economically rational person's train of thought.

Counteracting the existing committers' economic interests is the need and desire to build a working community. Also, the participants in a new project are likely to appreciate every helping hand while a mature project might not need any additional committers.

Thus, the following forces possibly influence a developer's promotion:

- the economic self-interest of the group of existing committers,
- the committers' philosophical convictions on running the project, and
- the project's need for more committers.

In many ways, investing in an open source project is like joining a startup. The earlier a developer joins, the higher the risk of the project not working out but also the more likely the ascension to committer status. The later a developer joins, the lower the risk, but also the lower the chances of becoming a committer any time soon.

The window of opportunity is small for those aspiring to achieve committer status in an important open source project. With the ongoing commercialization of open source, many current projects expect a committer to work full-time on the open source project. Otherwise, committer status wouldn't be granted. This, for example, is what the Eclipse project Web site states about its core projects (www.eclipse.org). However, a company is likely to let an employee work full-time on an open source project only if that person is already a committer; otherwise, how many of the benefits of its contributions the company would reap is uncertain.

A developer who chooses the right project can gain and maintain a position that will increase salary-negotiation power and job prospects. The developer will enjoy those benefits as long as the project is of significance to potential employers.

Open source reinforces the trend toward employees becoming "free agents." Committers who rationally follow their economic interests are likely to be more loyal to the open source project than to their current employer because that's where their market value lies. This results in a more fluid job market where developers can be expected to move around more freely and more frequently than in the past.

Open source software has enabled large system integrators to increase their profits through cost savings and reach more customers due to flexible pricing. This has upset existing ecosystems and shuffled structural relationships, resulting in the emergence of firms providing consulting services to open source projects. This new breed of service firm in turn lives or dies by its ability to recruit and retain appropriate talent.

For such talent, in particular for software developers, life has become more difficult and exciting at once. Developers face new career prospects and paths, since their formal position in an open source project, in addition to their experience and capabilities, determines their value to an employer. Economically rational developers strive to become committers to high-profile open source projects to further their careers, which in turn generates more recognition, independence, and job security.

References

1. R. Goldman and R. Gabriel, *Innovation Happens Elsewhere*, Morgan Kaufmann, 2005.
2. E. Raymond, *The Cathedral and the Bazaar*, O'Reilly, 2001.
3. E. Haruvy, F. Wu, and S. Chakravarty, "Incentives for Developers' Contributions and Product Performance Metric in Open Source Development: An Empirical Exploration," Univ. of Texas working paper; www.iimahd.ernet.in/publications/data/2005-03-04sujoy.pdf..
4. J. Lerner and J. Tirole, "Some Simple Economics of Open Source," *J. Industrial Economics*, vol. 50, no. 2, 2000, pp. 197-234.
5. K.R. Lakhani and R.G. Wolf, "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects," *Perspectives on Free and Open Source Software*, J. Feller et al., eds., MIT Press, 2005, pp. 3-22; http://freesoftware.mit.edu/papers/lakhaniwolf.pdf.
6. I-H. Hann et al., "Economic Returns to Open Source Participation: A Panel Data Analysis," unpublished working paper, Univ. of Southern California; http://opim.wharton.upenn.edu/wise2004/sun412.pdf.
7. G. Moore, *Inside the Tornado: Marketing Strategies from Silicon Valley's Cutting Edge*, Harper Perennial, 1999.
8. C. Christensen, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*, Harper Business, 2000.
9. S. Krishnamurthy, "An Analysis of Open Source Business Models," *Perspectives on Free and Open Source Software*, J. Feller et al., eds., MIT Press, 2005, pp. 279-296; http://faculty.washington.edu/sandeep/d/bazaar.pdf.

Dirk Riehle *leads the open source research group at SAP Research. His research interests are open source software, collective intelligence and wikis, and object-oriented software architectures. Riehle received a PhD in computer science from ETH Zurich, the Swiss Federal Institute of Technology. He is a member of the IEEE and the ACM. Contact him at dirk.riehle@sap.com or dirk@riehle.org; see also www.riehle.org.*