Open Source Software in Industry

Christof Ebert, *Vector*

Many of today's most innovative products and solutions are developed on the basis of free and open source software (FOSS). Most of us can no longer imagine the world of software engineering without open source operating systems, databases, application servers, Web servers, frameworks, and tools. Brands such as Linux, MySQL, Apache, and Eclipse have shaped product and service development. They facilitate competition and open markets as well as innovation to meet new challenges. De facto FOSS standards such as Eclipse and CORBA simplify the integration of products, whether they're all from one company or from multiple suppliers. *IEEE Software* has assembled this theme section to provide a brief yet practical overview of where FOSS is heading.

Open source refers to software that you may freely use, modify, or distribute provided you observe certain restrictions with respect to copyright and protection of its open source status. A major difference between FOSS and freeware or public-domain software is that FOSS generally has a copyright. FOSS isn't free software and often requires substantial investment before you can deploy it in the marketplace. Many current engineering processes have evolved from the way open source is developed. This holds for iterative development and agile techniques as well as globally distributed software engineering.

Today's trends

Innovation is driven by open source software.[1] Today, we rarely see traditional software development, where one company designs and builds an application or product from scratch and then integrates and evolves it within a defined market. That's no longer a healthy business case. Engineering life cycles, business models, distribution channels, and services have dramatically changed. Software products are so complex that it would be impossible to develop, maintain, and evolve only unique software. Companies are moving up to the applications and service level, increasingly relying on reused, externally available components. Typical schemes today imply that software companies work in a market segment, building their specific products or services on top of existing frameworks, libraries, Web services, and other components. Many companies focus on a specific value creation model built upon reusing components with as little reworking as possible. Competition is fierce; it demands that we reuse software on a level that even reuse protagonists some 10 years ago would have considered hard to believe.

Today, using FOSS at some point along the product life cycle or as a technology basis is no longer optional—whether for commercial or for engineering and quality reasons. Of course, not all software is built with FOSS components, and certainly the software industry's value networks would cease to exist if we simply gave our products away for free. But just think for a minute about where you're using FOSS during your daily software engineering work. The communications networks you depend on are built with FOSS components. Some of your libraries and engineering tools use it. Most intriguing, however, is the evolution of supplier networks based on the spirit of FOSS. A general trend in the software industry is toward supplier-user networks that cooperate and collaborate in many ways. For instance, paid online services are

0740-7459/08/$25.00 © 2008 IEEE

offered on the basis of FOSS. Closed communities have been created with suppliers and their customers, using open source processes and mechanisms to provide faster access to hardware drivers, software updates, or specific features. Suppliers are teaming up to share their software base and address single-user segments with tailored packages of software and services.

Advantages

Independent software vendors distribute popular FOSS solutions and components or integrate them and thus help accelerate integration efforts. This reduces the cost and risks of integrating several software components from various suppliers. Because FOSS makes its code available, you can substitute or revise your list of suppliers and support conditions. Other software vendors or software houses can start supporting specific FOSS at any time, mitigating the risk we face when the sole supplier stops supporting its proprietary software.

Depending on the product, its usage, and market constraints, FOSS has several advantages. With a substantial user community, many FOSS components and tools have evolved into de facto standards. This makes the FOSS a lasting solution that can resist the commercial-supplier uncertainties that can often abruptly end a product's life. Developing with mainstream FOSS reduces cycle time for component updates and corrections. More, and often free, labor is available to localize and correct defects. Especially for embedded systems, FOSS provides fast, new drivers and hardware-related features—even for rather exotic environments. Developing an application on a proven, de facto standard and binary compatibility protects the application against changing supplier conditions.

Many studies evaluated COTS and FOSS quality. They found that because of the principle of "many eyeballs," widely used FOSS has better quality than COTS. Some fear that security is at greater risk because the source code is available, but many more people review the source code of broadly used FOSS than that of proprietary software. For that reason, security breaches are typically fixed quickly and with broad notification to the user community. Students use open source in school, which substantially shortens their learning curve when they go to work for software companies. Engineers often have the same FOSS tools at home, which affects the work climate and productivity positively.

The articles

We've selected two articles for this mini-theme on open source software. The first article, "Should You Adopt Open Source Software?" by Kris Ven, Jan Verelst, and Herwig Mannaert from the University of Antwerp, looks into five of the commonly quoted factors used to support a FOSS decision-making process. This includes cost advantage, source code availability, software maturity, avoiding vendor lock-in, and the availability of external support. The article evaluates these criteria on the basis of literature surveys and interviews with decision makers from 10 companies. The results underline that all these factors have their pros and cons. Using FOSS in a practical way requires substantial evaluation of the individual impacts and influences from the selected software, its supplier and service networks, and the organization that wants to use it.

The second article, "Running a Bazaar inside a Cathedral," by Jacco Wesselius from Philips Medical Systems, looks at how companies evolve community source mechanisms to the benefit of faster and innovative platform development. For more than five years, Philips has applied open source mechanisms for sharing and creating communities with suppliers and customers. This article provides insight on how the company approached this and what lessons could be learned so that other companies might choose a similar route. He provides criteria for selecting the right business model for sharing and collaborating, such as taxation, fixed fees, or codevelopment.

Developers typically associate OSS with a bazaar-style approach, replacing the more traditional cathedrals.[2] In fact, companies can transition to OSS in many ways—bazaars don't replace cathedrals. Neither is there one given road to evolve from a cathedral-driven approach to the free and easy-going bazaar. Coexistence between these two approaches seems a valid model for many companies that don't want to share everything and for those that prefer to develop business models specifically tailored to their needs, products, customers, and life-cycle processes. Philips found that running a bazaar inside a cathedral is possible. The conflicting natures of the cathedral-like closed environment and the bazaar-like internal software market can be effectively merged into a codevelopment model. Such projects give participating stakeholders—both suppliers and users—a high level of control of timing, quality, and specification of the platform.

FOSS software can fulfill software engineering's basic needs; it's a natural, inevitable evolution. With a more mature FOSS business model and value chain, it's increasingly important that software engineers become proficient regarding the advantages and constraints associated with specific FOSS components.

For a guided introduction to FOSS with references to key articles, take a look at the IEEE Computer Society's TechSets (bundles of articles) on FOSS: "An Introduction to Free and Open Source Software," "Working with Free and Open Source Software," and "Free and Open Source Software: An In-Depth Look" (click on the TechSets link at www.computer.org/buildyourcareer). These online packages provide a practical introduction to FOSS concepts together with useful guidance on applying FOSS to your projects.

References

1. C. Ebert, "Open Source Drives Innovation," *IEEE Software*, vol. 24, no. 3, 2007, pp. 105–109.
2. E.S. Raymond, *The Cathedral and the Bazaar*, 2006, www.catb.org/~esr/writings/cathedral-bazaar.

Christof Ebert is managing director at Vector. In his prior role at Alcatel, he led the introduction of real-time Linux and other FOSS components. Contact him at christof.ebert@vector-consulting.de; www.vector-consulting-services.com.