Editor: Diomidis Spinellis ■ Athens University of Economics and Business ■ dds@aueb.gr

Open Source and Professional Advancement

Diomidis Spinellis

Doing really first-class work, and knowing it, is as good as wine, women (or men) and song put together. — *Richard Hamming*

I recently participated in an online discussion regarding the advantages of the various certification programs. Some voiced skepticism regarding how well you can judge a person's knowledge through answers to narrowly framed multiple-choice questions. I believe that the way a certification program examines a person's skills is artificial to the point of uselessness. In practice, I often find solutions to problems by looking for answers on the Web. Knowing where and how to search for an answer is becoming the most crucial problem-solving skill, yet typical certification exams still test rote learning. Other discussants suggested that certification was a way to enter a job market where employers increasingly asked for experience in a specific technology. My response to that argument was that open source software development efforts offer us professionals a new and valuable way to obtain significant experience in a wide range of areas and to advance professionally.

Software development

The most obvious way for a professional to benefit from open source software is by fixing and improving existing open source code. We all know that 40 to 70 percent of the effort that goes into a software system is spent after the system's first incarnation. Yet coursework and textbook exercises seldom ask us to maintain an existing system. On the other hand, many (perhaps too many) open source projects have lists chock-full of exciting additions and obscure bugs eagerly waiting for us developers to get our hands on them. By joining an existing open source software project, we can immediately practice the art of maintaining other people's code and sharpen our corresponding skills. Also, once we get our hands dirty, we'll see ourselves gradually adopting a more readable and maintainable code style.

Joining an open source project is an easy way to rub shoulders and interact with highly respected professionals. From day one, we can see what their code looks like, how they address new issues, and how they interact with other developers. Even better, as we begin to contribute code to the project, these colleagues might send us feedback that will help us improve. (The first comment I got from my FreeBSD mentor when I submitted code for review was that I had left blank spaces in the ends of the lines. Up to that point, I'd never thought that these could be an issue; from then onward, I configured my editor to color them yellow so that I could easily spot them.) I can't guarantee you friction-free interactions with the other developers, but those heated email exchanges can help us become better team players: we'll gradually learn to focus on an argument's technical aspects and not take attacks on our code personally.

Through participation in open source projects, we can also perfect our nonverbal communication skills. Open source projects, being globally distributed, typically rely on a multitude of collaboration tools ranging from email and instant messaging to issue management databases, wikis, and version control systems. Communicating requirements, design and implementation options, and bug descriptions through these media in a precise and technically objective manner is an important skill in today's global marketplace—a skill that will surely sharpen through our exchanges with our fellow open source developers.

A valuable feature of the open source landscape is the breadth of available applications, implementation technologies, and project sizes. By choosing cleverly, we can maximize both our professional gain and our personal joy. We can select a project to learn a new technology or to improve our skills in an existing one. We can thus transfer our skills from, say, ASP to Ajax or, alternatively, cut our teeth on advanced Java programming through Eclipse's multimillion-line code base. We can also enter a new application domain, such as game programming, networking, or kernel hacking. Finally, the diversity of open source project sizes gives us an excellent opportunity to inject variety into our professional life. If we work in a small start-up, we can join a large project to gain a taste of a structured development process; if our company is process heavy, joining a small project lets us experience once again the joy of coding.

System administration

Increasingly, software systems aren't monolithic blocks but complex, large, heterogeneous ecosystems. In such an environment, the software professional must be a system administrator, selecting, configuring, connecting, and tuning subsystems into a robust and efficient larger part. Again, the typical classroom or corporate development setting is often a sterile affair involving preselected, preinstalled, and preconfigured components that just work.

With open source software development, we can get the larger picture. We get a chance to experiment with the components, tools, and our development environment, choosing and configuring a setup that works and lets us be productive. At different times, we wear the hats of a system administrator tinkering with operating system releases, a database administrator configuring a relational database, a security officer implementing our security policy and installing patches, and a network manager making the pieces of second-hand hardware junk that inevitably piles up in any self-respecting hacker's basement talk to each other. Any of these skills is valuable in today's marketplace, and the cross-disciplinary mixture that we'll acquire from our involvement in open source projects is even more so.

Development process

Consider development practices such as issue tracking, version control, unit testing, style guidelines, the daily build, code reviews, release engineering, and traceability. If you work in a small development group or a startup, chances are that you (or, worse, your boss) consider some of these practices obscure and irrelevant. Yet they are anything but. Although a talented programmer can often get away with developing software by piling code layer upon code layer, this process isn't sustainable in the long run. When the software and the team that builds it grow large, failure to adopt a process that includes the practices I named borders on hubris. Joining a large open source development project will get you first-hand experience with many cutting-edge development practices. So, apart from polishing your coding skills, you'll also become a better manager by observing how things you might have heard only in a boring software engineering lecture actually work in large, real-world projects.

Later on, hopefully, you'll also contribute. Despite the size and complexity of some large open source development efforts, most projects are still typically too light on process, so it's relatively easy for somebody with time and ideas to make a contribution in this area. Initially, this can be simply a skunk works subproject you launch on your own: a framework for unit or regression testing, a bug-finding tool, or a more efficient release mechanism. As your idea is proven on the field and you gain respect from other developers, chances are that the project's community will officially adopt your pet venture.

Cashing in

Proponents of *psychological egoism* maintain that we're always motivated by self-interest, even when we behave altruistically: deep down we seek the better feeling we derive from our acts. This argument has been criticized as circular and nonfalsifiable. Fortunately, when working on open source projects, we won't have to entangle ourselves in this logic: there's nothing wrong with advancing professionally while helping worthwhile projects. Nobody (yet) has promised eternal life through code churning.

We already saw how participation in open source projects can make us better programmers, system administrators, and managers. As contributors to open source projects, we can also often gain a significant edge in interviews: "I see you're using Firefox as your browser. You know, I've implemented the hyphenation functionality in the text-rendering module." Developers with commit privileges in certain high-profile open source projects often find themselves in a seller's market. Demand for their skills typically outstrips the available supply, and therefore they can command better employment terms. Nevertheless, in the end, the best reward we gain from our participation in open source projects is the joy of contributing to work that could improve millions of people's lives.

Diomidis Spinellis is an associate professor in the Department of Management Science and Technology at the Athens University of Economics and Business and the author of *Code Quality: The Open Source Perspective* (Addison-Wesley, 2006). Contact him at dds@aueb.gr.