Q&A

Carlo Daffara

Q. What are the long-term effects of open source?

A. It is no longer fashionable to talk about open source. We debate clouds, virtualized infrastructures, services, and BYOD (bring your own devices) as if those happened in a vacuum. To the contrary, I would like to present some data points that indicate how economics and physical constraints are turning open source into the hidden background on which a substantial number of IT trends are being based. In this way, we can see the long-term effects of open source.

Writing good software is an art more than a science, with several trends-du-jour that appear and disappear at breakneck speed. However, it still seem to be impossible to beat Fred Brook's law: “there is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement within a decade in productivity, in reliability, in simplicity … [and] we cannot expect ever to see two-fold gains every two years” (Brooks, 1986; http://wikipedia.org/wiki/No_Silver_Bullet). This proclamation dates from 1986, and we still see a yearly compound aggregate growth rate that is between 3 and 7% (European Commission Technology Working Group, 2004; http://tinyurl.com/728xu9w). In contrast, the code complexity in systems and devices that include software is growing at a much higher rate – a 58% compound annual growth rate – and the number of devices is growing as well, at around 10% yearly. This means that there is no way for software developers to be able to create all that source code in time, independently from the use of whatever tool or methodology.

This is, basically, one of the major reasons for the adoption of free/libre open source software F/LOSS, even by companies or groups that are fundamentally averse to the idea – they had no alternative. And the amount of code reuse is growing at a substantial rate as well. Recently, Black Duck (a provider of code auditing services) found that, in an analysis of large scale code bases with an average of 700MB of source code, 22% was open source and that 80% of new development is avoided through reuse of F/LOSS code. Thus, open source compensates for the increased complexity of software projects while at the same time containing costs; it also reduces the time to market and maintenance effort.

What is the indirect effect of such a massive introduction of F/LOSS code within multiple software projects? There are several, interconnected results:

1. The reused code improves faster than the rest of the code base, indirectly increasing the economic value of the F/LOSS projects. This effect is the basis for several research results that demonstrates that large, successful open source projects tend to have a very high code quality in terms of defect per line of code, on par or better than proprietary code. (See Mohagheghi et al., 2004; http://tinyurl.com/6o95opr for details and further observations on the effect of reuse on code quality and maintainability.) This in turn increases the probability that the code will be reused in the future, and reduces the cost of integration – a positive feedback loop for adoption.

2. Even with a small number of adopters contributing back patches and effort, the increased participation and the positive feedback creates an opportunity for superlinear growth in the affected project – an effect that is not hampered by increased complexity and communication costs, further increasing the value of the reused code.

3. The implicit support of open standards by open source code facilitates the adoption of open standards in the assembled code as well – thus “osmotically” promoting openness. In fact, this is one of the reasons for the extraordinary support of recent HTML5 engines and libraries and at the same time the maturation of the web as a delivery medium for applications at the expense of non-open additions and protocols.

It is possible to continue further in this little experiment: the “good enough” status of HTML5 as an application delivery platform allows users to reduce the reliance on locally installed apps, up to the point where all apps are delivered this way (eventually with a gateway bridge between legacy apps and the web, like VMware's project AppBlast; http://tinyurl.com/bnuvfyy). At this point, if all you need is a browser, execution platforms become interchangeable – you can use an ARM processor, a MIPS one, whatever. It means that cost-effective alternatives become feasible, such as the RaspberryPi platform (http://raspberrypi.org), which at $25 can be even embedded directly in a monitor at little added cost.

A further long-term effect will be an increase in structured collaboration across industries and companies participating in F/LOSS development – something that is now restricted mainly to a few horizontal platforms such as Eclipse or embedded Linux. As the economic advantage of F/LOSS becomes more visible, a larger number of participants will start to explore collaboration in vertical frameworks, such as industry specific toolkits or individual packages that may be relevant only to a few. Examples such as Albatross (http://albatross.aero), an air traffic control workstation, that now seem peculiar will become quite the norm, as more and more developers will go from pure integration of open source pieces to reduce development cost to a more structured collaborative participation. This shift will occur especially for companies that are not primarily IT producers, but users; this will further increase the shift from packaged software to reusable components, again reinforcing the movement towards F/LOSS.

Another effect will be changes in revenue per dollar spent: because software can grow faster with more or less the same spending level, the company will grow faster. A dollar invested in F/LOSS collaboration will bring back a real value that is substantially higher, thanks to the sharing of costs across collaborators.

F/LOSS is a game changer in more ways than simple reuse. Reuse at large scales changes the economics of IT in more profound ways, allowing better software, more software, and more affordability for everyone.

About the Author

Carlo Daffara is head of research at Conecta, an open source consulting company. He is the Italian member of the European Working Group on Libre Software, chairs several other working groups, including the Open Source Middleware Group of the IEEE Technical Committee on Scalable Computing and the Internet Society Working Group on Public Software, and contributed to the article presented by ISOC to UNESCO on global trends for universal access to information resources. His current research activity is centered on the sustainability of business models for open source software.

Citation: Daffara, C. 2012. Q&A: What Are the Long-Term Effects of Open Source? *Technology Innovation Management Review*. January 2012: 29-30.