Chapter IV

Standardization and Other Coordination Mechanisms in Open Source Software

Tineke M. Egyedi,
Delft University of Technology, The Netherlands

Ruben van Wendel de Joode,
Delft University of Technology, The Netherlands

ABSTRACT

Open-source software (OSS) offers programmers the opportunity to elaborate and adapt source code. It is an opportunity to diverge. We would therefore expect incompatible strains of software to develop, and, consequently, a demand for standardization to arise. However, this is only partly the case. In this chapter we explore which other coordinative mechanisms are at work next to committee standardization. We identify four other categories of coordinative mechanisms and illustrate their relevance in OSS development. They complement committee standardization, can be used in standardization, and are sometimes an alternative to standardization.

INTRODUCTION

“What if, when the software doesn’t work or isn’t powerful enough, you can have it improved or even fix it yourself? What if the software was still maintained
even if the company that produced it went out of business?” (Perens, 1999, p. 171)

Perens (1999) refers to the possibility to adapt and improve open-source software (OSS) to specific user needs. Open-source code means that anyone with programming skills can understand how the software works. Moreover, the user licenses that govern OSS explicitly allow modification and redistribution of the modified source code. Most OSS are freely available, and they are easily accessible on and downloadable from the Web sites of open-source communities. These communities primarily consist of hobbyists. Many of them are sizeable. For example, the Apache community roughly comprises 630 contributors of which about 90 belong to the core developer group (http://www.apache.org/~jim/committers.html). In the communities, a variety of mature software programs is developed and maintained (e.g., Linux and Apache).

In effect, anyone with access to the Internet can modify the source code and start a new branch thereof. To estimate the number of people that might do so, let us imagine that 1,000 people download a software program—a much too modest figure. When assuming that 10% has programmer skills, of which again 10% actually feels the need or urge to modify software (e.g., they found a bug), this still amounts to at least 10 modifications to the original software (van Wendel de Joode, de Bruijn, & van Eeten, 2003).

There is no entrance fee for participating in OSS development. In most cases, one does not even need to register to download software and modify its source code. Low entry barriers ease the emergence of diversity in OSS (Franck & Jungwirth, 2003; Sharma et al., 2002). The absence of any significant entry barrier—apart from engineering expertise—also explains the variety among those actively involved in OSS development: “[T]here are many different types of contributors to open-source projects: organizations and individual initiators and those who help with other people’s projects; hobbyists and professionals” (Markus, Manville, & Agres, 2000, p. 15).

The different types of involvement and diverse backgrounds imply different requirements with regard to the software being developed. With both motive and opportunity to modify code, one would expect many branches (forks) and variants in open-source software to emerge. Such diversity more likely leads to incompatibility and fragmentation.

As Kogut and Metiu (2001, p. 257) note, “[t]he danger of open-source development is the potential for fragmenting the design into competing versions.” A classic example of fragmentation is UNIX, a multiuser operating system and open-source initiative avant la lettre. It was a de facto standard in the late 1970s. Different UNIX variants developed, which fragmented the market.

An important means to counter diversifying fragmentation and the incompatibility that this brings is standardization (e.g., Linux Standards Base [LSB]; Egyedi & van Wendel de Joode, 2003). However, standardization alone does not explain the relative coherence in certain areas of OSS where an OSS has