Chapter 7.16
Morality and Pragmatism in Free Software and Open Source

Dave Yeats
Auburn University, USA

ABSTRACT

This chapter analyzes the differences between the philosophy of the Free Software Foundation (FSF) as described by Richard Stallman and the open source movement as described in the writings of Eric Raymond. It argues that free software bases its activity on the argument that sharing code is a moral obligation and open source bases its activity on a pragmatic argument that sharing code produces better software. By examining the differences between these two related software movements, this chapter enables readers to consider the implications of these differences and make more informed decisions about software use and involvement in various software development efforts.

INTRODUCTION

As governments around the world search for an alternative to Microsoft software, the open source operating system Linux finds itself in a perfect position to take market share from Microsoft Windows. Governments in France, Germany, The Netherlands, Italy, Spain, and the United Kingdom use Linux to encourage open standards, promote decentralized software development, provide improved security, and reduce software costs (Bloor, 2003). The Chinese government strongly supports Linux as its operating system of choice because Chinese experts have complete access to the source code and can examine it for security flaws (Andrews, 2003). In Brazil, leftist activists gathered to promote the use of open source software (OSS) (Clendenning, 2005).

There is a connection between the technological reasons for choosing open source software and the political ones. Many governments see open source as a way to promote a socialistic agenda in their choices of technology. Open source advocates, however, do not necessarily make these connections between the software development methods involved in open source and political movements of governments. There is evidence, however, that leaders in the open source movement have expressed their rationale for advocating opening the source code of software.

The open source movement can trace its roots back to an alternate, still very active, software movement known as free software. While open source and free software can (and do) coexist in many ways, there are some essential differences that distinguish the two groups from one another. Perhaps most notably, the free software movement is based on a belief in a moral or ethical approach to software development, while open source takes a much more pragmatic view. While both groups argue for the open sharing of source code, each has its own reason for doing so. Understanding the differences between open source and free software can help open source researchers use more precise terminology and preserve the intent of each of these groups rather than assuming that they are interchangeable.

The following chapter begins with a brief historical overview of the free software and open source movements and highlights some of the main beliefs of each. The chapter then offers an examination of both the moral and pragmatic aspects of open source software. The conclusion invites readers to consider the implications of the differences between the two viewpoints and suggests ways for readers to apply this information when making choices about software.

BACKGROUND

The open source movement grew out of the software development practices in academic settings during the 1970s. During those early years of software development, computer scientists at colleges and universities worked on corporate-sponsored projects. The software developed for these projects was freely shared between universities, fostering an open, collaborative environment in which many developers were involved in creating, maintaining, and evaluating code (Raymond, 1999).

In his *A Brief History of Open Source* article, Charlie Lowe (2001) describes the end of open and collaborative methods of developing computer software in the 1980s when the corporate sponsors of academic software projects began to copyright the code developed for them. Corporations claimed that the university-run projects created valuable intellectual property that should be protected under law. This, of course, was just one of the signs of the shift from the commodity-based economy in the U.S. to a knowledge-based one. The wave of copyrights threatened to end the collaboration between computer scientists and slow the evolution of important projects. It looked as if the computer scientists would be required to work in smaller groups on proprietary projects.

Richard Stallman (1999) reports that he created the GNU General Public License (GPL) to maintain the ability to collaborate with other computer scientists on software projects, without restriction. The name GNU is a self-reflexive acronym meaning "GNU's Not UNIX," a play on words that pays homage to and differentiates itself from the UNIX legacy.[1] Stallman was concerned that the UNIX operating system, created during the collaborative era of the 1970s, would no longer be supported by new programs that used its stable and robust architecture when access to the source code was cut off. Stallman started the GNU initiative (which enabled the establishment of the Free Software Foundation [FSF]) to ensure that new software would be freely available.

The GNU GPL gave programmers the freedom to create new applications and license them to be freely distributable. Specifically, the GNU GPL gives anyone the right to modify, copy, and redistribute source code with one important restriction: Any new version or copy must also be published under the GNU GPL to insure that the improved code continues to be freely available. Many programmers (both those accustomed to the academic practices of the 1970s and new computer enthusiasts) adopted the GNU GPL and continued to work in open, collaborative systems.