

Taxonomy of Open Source Software Development

Kumiyo Nakakoji^{1,2} and Yasuhiro Yamamoto¹

¹Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma
Nara, 630-0101, Japan

²SRA Key Technology Laboratory Inc.

kumiyo@is.aist-nara.ac.jp, yasuhi-y@is.aist-nara.ac.jp

+81-743-72-5381, 5383(fax)

There have been a number of controversial discussions over open source software development. Such discussions include whether it produces more secure software than traditional one, whether its development style actually adheres to more “cathedral” than “bazaar,” and so on.

Many of those controversial arguments, however, seem to stem from the over-loaded term “open source software development.” In fact, there are many completely different development projects, which are labeled “open source software development” only because they produce open source software under open source licenses, such as GPL, OSI’s, or BSD’s.

We have conducted case studies over four different open source software development projects, which have been conducted within SRA Inc., Japan. SRA is a leading company in the open source movement in Japan, who has been supporting the activities of the Free Software Foundation since 1987, and has carried out a variety of open source software development projects within its Open Source Business Division. The four projects we have looked at are:

1. the Wingnut project, which provides a support for a number of GNU software, such as GCC and Emacs, for Japanese industries who need to port GNU software into their hardware;
2. the Linux project, which as a master SI distributor offers a support for Linux users;
3. the PostgreSQL project, which supports customers who use the PostgreSQL database, which is an open source database; and
4. the Jun project, which is a 3D graphic and multimedia application library for VisualWorks smalltalk and Java.

Those four projects vary in their development styles, communication styles, management styles, evolutionary styles, and in their underlying philosophy. From our case studies, we have found that these differences come from the differences in what aspects each project focuses on.

We identified three major focuses: *archetype*, *security*, and *rapidness*.

- *Archetype*. This type of software, represented by GNU software and the Jun library, is developed so that people can all share the software and modify it if necessary. The purpose is to save time and effort of software developers by not implementing the same functionality over and over again only because the source code is “closed.” Being *archetype*, this type of software must be developed by expert programmers and of very high quality. Coding standard and programming

styles are usually strictly observed. There should be a single archetype software program corresponding to a unit of functionality. Adaptations and bug-fixes carried out by the user community need to be fed-back into the evolution of the archetype software.

- *Security*. This type of software, represented by the PostgreSQL database, is developed as open-source so that it becomes fault-tolerance. It improves its security level by having many programmers examine its source code. As soon as someone reports a fault in its mailing list, PostgreSQL community starts looking for a cause and debugging. This type of software is usually very conservative against evolutionary changes.
- *Rapidness*. This type of software, represented by the Linux operating system (excluding the Linux Kernel, which is developed more like Archetype), needs rapid and prompt adaptation and modification when necessary. To take a hardware driver for the Linux OS, for instance, a programmer develops a driver by necessity and put it on the Web so that other people can take and use it. Users try to find program components on the Web when they face a need for the software. If they do not find one, they will develop it and share it via the Web. This type of software development is a typical bazaar type software development. Because there can be many alternatives and different versions for a single functionality, distribution packages are necessary to identify a typical set of program components chosen among a number of available programs.

We do not mean that these three aspects are mutually exclusive. In fact, many open sources software projects have more than a single focus, probably covers many focuses. However, emphasis on a particular focus determines the project's management, development, communication, and evolutionary styles. Table 1 summarizes our preliminary attempt to characterize each type.

Table 1: A Set of Characteristics for the Three Types of Open Source Software Projects

focus	purpose	control	development	feedback for debug	feedback on evolution	version	code access
<i>Archetype</i>	reference model	central	core member	must	must	single tree	all the time for modification and learning
<i>Security</i>	fast fixes	central	core member	must	not necessarily	single tree	when a bug is reported
<i>Rapidness</i>	timely development	distributed	community	not necessarily	not necessarily	multiple branches	only by interested programmers

Through the case studies, we have identified a wide range of varieties among the software projects, which are all labeled “open source.” We argue that we urgently need taxonomy and better names for different types of open source software development. By having a taxonomy, we would be able to have more constructive, fruitful discussions on how and why open source is a promising approach for the development of a certain type of software, which of existing software engineering frameworks should be used to support open source software development, and on which aspects of open source software development cannot be supported within the existing software engineering approaches.