Chapter 85
The Cultural and Institutional Barrier of Knowledge Exchanges in the Development of Open Source Software

Ikbal Maulana
https://orcid.org/0000-0002-3727-3809
Indonesian Institute of Sciences, Indonesia

ABSTRACT

Open source software (OSS) gives developing countries inexpensive or free alternatives to proprietary software. It gives them the opportunity to develop software and software industry without starting from scratch. This chapter discusses the diffusion and development of OSS in Indonesia especially after the government took "Indonesia, Go Open Source" (IGOS) initiative. This initiative united government organizations, communities, R&D institutions, and universities. While the government's concern was to tackle piracy by replacing illegal software with OSS, the others sought to develop their own OSS. However, the openness of their software is only in terms of that they were developed using OSS development tools, while their mode of development remained closed, which was caused by cultural barrier and institutional incompatibility between government's regime of project administration and the governance of OSS development.

INTRODUCTION

Industry has developed to be increasingly relying on technology rather than on workers. In the past or today in craftsmanship traditions, technology is manifested as tools that are only useful in hands of skillful workers, whereas in modern production systems workers "becomes a mere appendage to an already existing material condition of production" (Marx, 1906, p. 421). As machineries becoming more sophisticated, because knowledge which was previously possessed by workers is increasingly embedded into them, they can be operated by much less skillful workers. This development allows capitalists to better

DOI: 10.4018/978-1-6684-3702-5.ch085

control their production systems and even transport their machineries to any developing country to find low-cost labors to operate them. The transfer of sophisticated machineries to developing countries does not automatically lead to the transfer of knowledge of production. When the machineries are moved to other countries, the workers' involvement in production does not develop necessary skills that allow developing countries to make the same production. Their skills of operating machineries apparently are necessary but easily replaceable parts of production process.

Software industry has given new promises to developing countries, because software can be produced on inexpensive machines. Software programmers can develop their skills by experimenting on widely available computers. Software industry, as part of information and communication technology (ICT) industries, seems to give the sense of promise because it periodically experiences transformation caused by the emergence of, what Christensen (Christensen, 2000) calls, disruptive technology. The disruption of technology demands industry players to play with new knowledge and technology, because the old one is not only irrelevant, but can be a liability to those who use it. It also explains why the innovations that direct this industry has been created by young people who have no significant experience in business and industry. This fact seems to give hope to entrepreneurs in developing countries as well. "Although it is dominated by firms based in major industrialized countries of the world, it continues to offer great prospects for economic growth and industrial development within developing economies" (UNCTAD, 2002, p. 3).

The strong reliance on knowledge rather than on technology may deceive people to underestimate the complexity of the development of software industry, as if knowledge can be easily acquired through simple learning and softwares can be easily produced through mere thinking in front of a computer. The intangibility of software gives them the illusions that the development of software and software industry are easy and inexpensive, and that both developed and developing countries have the same opportunities because this industry does not rely on expensive production capital. Indeed, if every country has the same opportunity, then the competition must be very high, and even higher for software industry, which produces products that are "costly to *produce* but cheap to *reproduce*" (Shapiro & Varian, 1999, p. 3), and can be inexpensively distributed throughout the world. Consequently, having the capability to create a working software is not sufficient, competition demands producers to create it better than similar products competing in the same marketplace. India has often been mentioned as the best exemplar of a developing country that can take advantage of the opportunity in global software industry (Nagala, 2005), but most of other developing countries can only dream of that achievement.

Opportunities in software industry for developing countries have become elusive if we see that the market leaders, which are from advanced countries, can make their clients dependent on them, because users of software "are notoriously subject to switching costs and lock-in: once you have chosen a technology, or a format for keeping information, switching can be very expensive" (Shapiro & Varian, 1999, p. 11). And as current softwares are also very complex, consisting of thousands to hundred thousand of codes which require time consuming, hence expensive, development and testing, it becomes more difficult for new software companies to challenge market leaders. So proprietary software industries from advanced countries have erected a high barrier for new entrants from developing countries.

The above barrier is irrelevant in the development of open source software (OSS). One of the basic tenets of OSS is that you do not have to develop anything from scratch, because it is legal for you to modify what others have developed, and, hence, "open source developers enjoy a great productivity gain from code reuse" (DiBona, 2006, p. 22). In OSS development, you do not need to reinvent the wheel, rather use the wheel invented by others and modify it when it is necessary. This principle is practiced