

Current OSS Research

In recent years, over 34.4% of colleges and universities have implemented some form of an open-source application or architecture in their IT department. The popularity of online education has increased the demand for more interactive systems to accommodate the rising number of distant learning students. Consequently, those 65.6% of organizations that are not using the popular blogs, wikis, or software agents are seeking to deploy them quickly to enhance the attractiveness of their institutional programs (Heigl, 2005). Several factors affect the decision making process in adopting OSS tools and architectural technology. A major obstacle among decision makers is the fiscal responsibilities in the initial phases of the project, including the required funds and support needed once the software becomes operational (Abel, 2006).

Abel (2006) commented that many computer professionals are not knowledgeable about open-source products; therefore, not supportive of migrating from proprietary systems to OSS. Stallman (2007) reported that many IT engineers do not have the expertise to decide what hardware components and system tools are required to initiate an open-source platform. It was also discovered that many IT managers are not knowledgeable of how to evaluate the feasibility of incorporating OSS technology to their current configuration. Several studies have reported on the use of free software in higher education. For example, Meyers (2006) articulated in the ZDNet Open-source Community that distance-learning organizations are very interested in open-source technology and reviewing its possibilities and rewards.

The ability to evaluate, update, and perfect an application is the core ideology of OSS software. There is a potential network effect where many programmers can individually contribute to existing applications in a positive way. For example, if users submit their patches to a central repository, all other users can update their system to include this patch, thus increasing

their own security levels in the process. Given that several users are likely to find different bugs; this allows for many errors and poorly written code to be removed. This process of checks and balances leads to the fast creation of more patches, and consequently more secure code being released back to the user in a timely manner. The mentality of software designers is that given enough eyes on the software, bugs are easier to find and fix (Yu, 2006).

In Hoepman, and Jacobs (2007) research, the openness of OSS code allowed for new patches to be released twice as fast as for closed source software. This approach cuts the software repair and update process in half; therefore, if a user is unable to patch a bug, the open-source approach enables the individual to communicate about bugs with developers more efficiently because both can use the same structure of reference, which is the source code of the application. OSS enables users to add extra security measures that are innovative and not in line with traditional security approaches.

Several tools exist to augment the security of existing systems if the source code is provided to the user. These tools do not rely on static checking of the code; instead, they add common runtime checks to the code to detect, buffer overflows or stack frame corruptions. OSS also permits the user to control the complexity of the application thus increasing its security by removing unnecessary parts. Most importantly, the open-source style of software development requires the software communities to be more careful, and to use the best available tools to secure their systems. The open-source methodology also requires the developers to put more effort into quality and control; whereby pressuring programmers to use clean coding styles and not sloppy syntax that is untrustworthy and unacceptable. It is important for companies and individual programmers alike to keep their respect and credibility as professional software engineers by producing efficient running code. Allowing coders to freely experiment with open

applications will stimulate research and development; therefore, improving the tools used for software development, testing, evaluation, and security verification (Hoepman, and Jacobs, 2007).

OSS Success Criteria

Defining success in OSS is a vague concept that may have different meanings across the development community and with stakeholders. A strong indicator of an application that is well received is the constant activity on the project's website such as downloads of the source code, and the number of individuals who monitor announcements about new releases regarding an application. This type of activity could be seen as user interest or possible adoption of the software. The difference between the user behavior, such as adoption and actual usage of OSS technology, can be viewed as a factor of success (Stewart, Ammeter, and Maruping, 2006).

This research will take into consideration the user interest in an OSS project as a factor of how the public accepts open-source systems in society. In (e.g., Stuart et al., 2006) study, it measured the level of interest in an OSS project as an indicator of how successful the program appeared to be. The actual approach counted the number of individuals who subscribed to a project, or registered to receive electronic new letters about a specific OSS application. A second technique measured the success of the project by how well it attracted interest and input from the development community.

Due to the fact that OSS projects rely on voluntary support, the ability of a project to attract and motivate contributors is a significant factor in the success of a project. The success criteria was measured by evaluating the level of activity on a project, such as how frequently are support requests answered, bugs fixed or new versions of the applications released back to the community. These factors were used to categorize the development activity on a project; the

specific measure used in the study was the number of new releases that were produced during a fixed period (e.g., Stuart et al., 2006).