Guest Editorial
A further investigation of open source software: community, co-ordination, code quality and security issues

In the first part of this double special issue on open source software (OSS) – *Information Systems Journal* 11(4) – three papers were presented:

- *Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies*, Michael J. Gallivan, Georgia State University, USA.
- *The power of gifts: organizing social relationships in open source communities*, Magnus Bergquist and Jan Ljungberg, University of Gothenburg, Sweden.
- *Putting it all in the trunk: incremental software development in the FreeBSD open source project*, Niels Jørgensen, Roskilde University, Denmark.

In keeping with our intention to investigate the methodological/software engineering, psychosocial, and economic/business issues of OSS (Fitzgerald & Feller, 2001), we have selected four further OSS papers for this issue.

The first paper by Srinarayan Sharma, Vijayan Sugumaran and Balaji Rajagopalan, Oakland University, USA, is entitled *A Framework for Creating Hybrid-OSS Communities*. It considers how the OSS model can be transferred to traditional software development organizations, with the associated benefits that the OSS model provides. The authors derive a framework based on the organization theory concepts of structure, process and culture, and use this framework to analyse OSS in some detail. Based on this analysis, the authors propose a further framework to assist in the creation of hybrid-OSS communities. This latter framework is founded on concepts of community building, community governance and community infrastructure.

The second paper, *Effort, Cooperation and Coordination in an Open Source Software Project: GNOME*, by Stefan Koch and Georg Schneider of the Vienna University of Economics and Business Administration, Austria, presents a useful quantitative analysis of an actual OSS development project, the high profile GNOME (GNU Network Object Model Environment) project. Detailed quantitative studies of OSS have been rare to date, and the findings of this paper can be compared with other widely cited OSS studies such as the Mockus *et al*. (2000) study of the Apache project and the Hermann *et al*., 2000) study of the Linux kernel development community. The paper proposes a number of important findings. For example, it confirms the strict division of labour which has always been a suspected characteristic of OSS –

they observe individuals working in relative isolation on different modules. This strict modularity is critical to achieving the globally distributed, parallel development model of OSS. The findings also reveal no correlation between the length of time developers are associated with the project and the amount of code they contribute. This is in marked contrast to traditional commercial software development where developers work more or less full time on the project. One possible interpretation of these findings is that much OSS development is contributed irregularly as developers work in their spare time, which contrasts a little with the findings of Jorgensen's study of FreeBSD in the first part of this double special issue. The study also reveals the presence of core groups in the GNOME project. Although 52 GNOME developers account for 80% of the code (in contrast to Apache where 15 individuals account for 80% of the code), the GNOME project appears to have an inner core of about 11 developers who are the most active. Also, GNOME experienced a huge increase in coding contributions since 1998, coinciding with the rapid rise in popularity of the OSS concept. One of the most significant contributions of the Koch and Schneider paper is the rigorous estimation of effort on the GNOME project. This is an important research area as the true cost and resource consumption on OSS projects is very difficult to estimate and has not featured in previous research. If the OSS model is to achieve more widespread applicability to software engineering in general, this type of information must be derived.

The third paper, *Code Quality Analysis in Open-Source Software Development*, is by Ioannis Stamelos, Lefteris Angelis, Apostolos Oikonomou and Georgios Bleris, Aristotle University, Greece. While many have written expansively about the high quality of OSS, very few studies have attempted to empirically validate the validity of such claims. This paper sets out to do just that. Using an automated code analysis tool, the authors analysed a random sample of 100 SuSE Linux 6.0 programs, and collected data on 10 metrics related to software quality. Using a weighted sum aggregation, they concluded that 50% of programs were of acceptable quality, 31% required further commenting, 9% were in need of inspection, 4% required further testing, and 6% needed to be completely rewritten. The study also investigated user satisfaction with these programs. It confirms the extreme importance of modularity, and hence, information hiding and structured design. This further illustrates the tight coupling between OSS and fundamental principles of software engineering (cf. Feller & Fitzgerald, 2002).

The final paper, *On the Security of Open Source Software*, by Christian Payne, Murdoch University, Australia, tackles the very important issue of the security of open source software, one of the critical considerations which will determine the long-term sustainability of OSS. He identifies the positive security implications of those integral aspects of OSS such as source code auditing and peer review processes. He also discusses the common argument that the invisibility of source code in closed source systems provides a layer of security through obscurity as bugs or potential security weak points may never be detected. Furthermore, he identifies the fact that in the rush to use open source software, the peer review process does not always have time to take place in an adequate fashion, as users expect others to perform the peer review task. The author conducted a detailed quantitative investigation of the issues on three projects, Debian GNU/Linux, OpenBSD and Sun Solaris the Debian and OpenBSD being open source projects while the Solaris system is closed source. The findings reveal that

OpenBSD scores both highest in terms of security features and lowest in relation to security vulnerabilities, than both Debian and Solaris, with Debian also outperforming Solaris. Thus, open source would appear to be more secure. Also, even though Solaris represents a closed source model, the number of security vulnerabilities found is higher than for the other two. The high score for OpenBSD is argued to be due to the very proactive nature of auditing and peer review in this project.

Brian Fitzgerald and Joseph Feller

REFERENCES

Feller, J. & Fitzgerald, B. (2002) *Understanding Open Source Software Development*, London: Addison-Wesley.

Fitzgerald, B. & Feller, J. (2001) Open Source Software: Investigating the Software Engineering, Psychosocial and Economic Issues. *Information Systems Journal*, 11, no. 4, pp. 273–276.

Hermann, S., Hertel, G. & Niedner, S. (2000) Linux Study Homepage, http://www.psychologie.uni-kiel.de/linux-study/ (accessed 10 May 2001).

Mockus, A., Fielding, R. & Herbsleb, J. (2000) A Case Study of Open Source Software Development: The Apache Server. *Proceedings of the 22nd International Conference on Software Engineering, Pp. 263–272.*