Guest Editorial:

Empirical Studies of Software Development and Evolution

Rachel Harrison¹

School of Computer Science, Cybernetics and Electronic Engineering
University of Reading
Reading RG6 6AY, UK

In some ways it seems extraordinary that it has taken the software engineering community until recently to consider the ways in which we do research. Empirical software engineering has given impetus to the search for research methodologies that will help us to evaluate and transfer technology in a timely manner. The growing interest in the area of empirical research is clear from papers published in journals such as this, as well as the success of workshops such as the IEEE Workshop on Empirical Studies of Software Maintenance (WESS), and the Workshop on Process Modelling and Empirical Studies of Software Evolution (PMESSE) that was held in 1997.

Following on from PMESSE, the Workshop on Empirical Studies of Software Development and Evolution (ESSDE) was held in LA in May 1999, co-located with the International Conference on Software Engineering and in co-operation with the IEEE Computer Society. The papers that comprise this special section were selected from those presented at ESSDE.

The aim of the Workshop was to bring together researchers and practitioners working in empirical software engineering and to raise awareness of such work. The workshop was intended to identify progress in the area of empirical studies and to inspire and encourage delegates to continue their empirical investigations.

The workshop papers, which were of a high standard, fell neatly into five main themes revolving around empirical studies: process modelling, systems evolution, process improvement, object-oriented technology, and directions and methodologies for empirical research.

Process modelling continues to excite interest from both academics and practitioners alike. David Raffo and Marc

¹ Tel: +44 118 931 8615; fax: +44 118 975 1994; email: rachel.harrison@reading.ac.uk
Kellner investigate the use of empirical analysis in software process simulation modeling. The authors discuss a number of issues concerning the use of real-world data and problems with sample sizes, outliers, lack of desired data, loosely defined metrics, etc.

Systems evolution is becoming recognised as a research field in its own right, widening the maintenance focus to include topics such as business and systems co-evolution, emergent information systems, designing for evolvability, and so on. Liz Burd and Malcolm Munro describe their investigations into the software evolution process using a number of applications written in C and COBOL. Change histories were plotted for sequential versions, giving graphical illustrations of the evolution process. The localisation of changes throughout the lifetime of the software was also tracked. The implications for reverse and re-engineering technologies are considered.

Connected to both software evolution and the object-oriented paradigm, the paper by Erik Arsiholm and Dag Sjøberg explores the problems that can arise in object-oriented software through the adoption of continuous incremental change that is typical of evolutionary development. The authors define changeability decay and discuss three possible approaches for assessing it.

Software effort prediction systems can help managers to decide how to allocate resources. However, current prediction systems leave a lot to be desired. Carolyn Mair et al. discuss the use of machine learning methods to construct software effort prediction systems. Artificial neural networks are shown to have superior accuracy to both case-based reasoning and rule-induction techniques, although the authors acknowledge that further research is needed to explore the advantages and disadvantages of neural networks in this context.

Encapsulation was once considered a cornerstone of software engineering. But encapsulation can easily be circumvented, particularly through the use of friends in languages such as C++. Steve Counsell and Peter Newson ask whether there is any evidence to show that friends are used in place of inheritance or to facilitate overloading.

The success of the ESSDE workshop, together with the WESS workshops held in conjunction with ICSM, suggests that there is an increasing recognition that it is important to
evaluate what software engineers do, and to provide feedback to researchers and developers. We hope that this trend will continue, and that the growing community will successfully abstract and disseminate the best practice that is being fostered in this area.