Directions and Methodologies for Empirical Software Engineering Research

R. Harrison
Department of Computer Science, University of Reading, UK

N. Badoo
Centre for Systems and Software Engineering, South Bank University, UK

E. Barry
Graduate School of Industrial Administration, Carnegie Mellon University, USA

S. Biffl
Institut für Softwaretechnik, University of Vienna, Austria

A. Parra
Software Engineering Laboratory/Computer Sciences Corporation, USA

B. Winter
University of Victoria, Canada

J. Wuest
Fraunhofer Institute IESE, Kaiserslautern, Germany

Abstract

This report summarises and builds on the results of the “Directions and Methodologies for Empirical Software Engineering Research” group discussion. In particular, we considered the strengths, weaknesses, opportunities and threats to empirical software engineering research in light of the discussions and presentations during the workshop. The following sections describe each of these aspects of our discussion in turn. In addition, to finalise our discussion we agreed on a three-point plan for future work.

Strengths

Our interest in empirical studies of software engineering reflects the increasingly popular view that software engineering should have strong foundations as a scientific and engineering discipline, and that techniques for improving software and systems development and evolution processes should be made available to practitioners.

Empirical analysis of software engineering is an important research method that can shed light on areas amenable to process and product improvement. Because it is based on observation, and reflects our actual experience with methods, tools and techniques, empirical research is closer to the real world than analytical or theoretical research. In addition, it enables us to cope with the multidisciplinary and interdisciplinary factors (Harrison and Wells, 1999) that frequently arise during the design of large systems (for example, socio-technical issues such as human factors, communication difficulties, quality of processes and products, etc.). Through reflection and meta-analysis made possible by replicated and repeated studies (where repeated studies are defined as those which are similar to previous ones, rather than being exact replicas), we can pursue more general theories to facilitate software development in a wide range of application

---

1 The workshop was held while Professor Harrison was at the University of Southampton, UK
Due to rapid and continual changes in software and systems there is an increased need for decision support during periods of intense business process changes. In addition, practitioners need models for process improvement and tools for verification of results. Empirical studies of software engineering can provide both quantitative and qualitative results to aid in development of these needed tools and models.

Empirical software engineering research methods such as GQM (Basili and Rombach, 1988) and AMI (Pulford et al, 1996) have been in place for some time and are gaining widespread acceptance. The explicit capture and sharing of knowledge that such methods encourage helps to raise awareness of the improvements needed in both process and product and quietly accentuates the level of skills needed in the workplace.

Weaknesses

Empirical research can lead to a number of problems. Empirical studies of software engineering are relatively difficult to control, primarily because of the large number of variables impacting software and systems evolution. In addition, the sort of multidisciplinary and interdisciplinary research these studies require is extremely difficult, as it calls for groups of people with a range of skills, such as social anthropology, cognitive psychology, etc. Although research methodologies from other disciplines are needed within software engineering, they may need to be tailored or adapted to suit the complex, holistic and fast changing business and industrial systems that are undergoing rapid development and evolution.

Another dilemma faced by empirical researchers is whether to conduct experiments or field studies. While experiments give researchers more control over exogenous variables and allow easier replication of results, there is difficulty in establishing sufficient external validity in the results to allow generalisation of conclusions and transfer of findings to other situations. On the other hand, field studies require researchers to enlist the participation of industrial collaborators. A full empirical investigation of a company's IS department and projects can be very disruptive and time-consuming for the company involved. Studies of this type can require a great deal of effort from both the researchers and the collaborators. Although managers are interested in the benefits that this effort will provide their business, researchers know that definite practical results are often only slowly realised.

Shari Lawrence Pfleeger has described the beneficial effects of having a metrics advocate on a project, who will take responsibility for providing convincing evidence of a payback from data collection and the use of metrics. Using this technique, and starting with problem projects, data collection can then be seen as welcome assistance rather than a time-consuming burden (Pfleeger, 1993).

Despite considerable empirical research in some application domains, and the generation of some large data sets, it is often difficult to generalise results, to understand reasons for conflicting results, and to envisage the way forward given such sets of results.

Opportunities

Empirical software engineering research offers us the opportunity to build and verify theories for software engineering (Lehman and Belady, 1976). Thus it provides steps
towards a better understanding of our discipline. By throwing light on specific aspects of software engineering we may, in the future, be able to generalize these results and so help to construct theorems and laws for software evolution.

The opportunity to evaluate systems from multiple viewpoints, using different techniques together with both qualitative and quantitative data, should be grasped wherever it is available. Such an approach is epitomized by the philosophy of triangulation, which advocates using a variety of research methods, in a variety of settings, with data from a variety of sources in order to find answers to open research problems (Seaman 1999). Careful experimental design can facilitate the design of research projects in order to maximize the impact of deliverables.

There is a wide range of research methodologies: methods such as formal experiments, case studies and surveys are well documented (Fenton and Pfleeger, 1995, Kitchenham 1996). In addition to these, Zelkowitz and Wallace outline a number of other methods, including project monitoring, assertion, field studies, literature searches, legacy data, lessons learned, static analysis, dynamic analysis and simulation (Zelkowitz and Wallace, 1998). Methods such as action research and participant observation are frequently used in information systems research projects (Galliers, 1991), and both involve participation by a researcher in the situation or action under investigation. However, the former involves a researcher who will actively intervene to try to achieve a particular outcome (Nandhakumar and Jones, 1997), whereas the latter involves trying to reduce the impact that results from the researcher being an outsider agent. Awareness of such techniques within the software engineering community seems to be limited.

We need to raise awareness of empirical software engineering research both in academia, through exemplar research projects, research degrees and taught degree programs, and also within the real world. If we fail to do this then metrics may be misinterpreted, spreading fear and alarm within environments in which they may be used (for example) to assess productivity. Managers and developers within business and industry need help in assessing metrics programs and in understanding what empirical research in software engineering has to offer them. We need to find methods of data collection that have a minimal impact on the development and evolution process, and are less invasive than those that are sometimes used.

Empirical software engineering research can help to find the most appropriate method, tool or technique for a particular application domain. There are clearly opportunities for independent evaluation by research organisations. Academics, for example, could provide consultancy to companies wishing to initiate metrics programs.

**Threats**

As far as threats to empirical research in software engineering are concerned, the group agreed that metrics can be seen to be threatening, in certain environments. This hampers the acceptance of empirical software engineering research, as does the publication of flawed empirical research. For example, consider the use of a *shotgun measure* approach that can produce too much data (Courtney and Gustafson, 1993). Outliers and overly complicated experimental design can also lead to inappropriate conclusions (Shneiderman, 1984, Scanlan, 1989). Results from investigations may be biased if the empirical research group is not completely independent of the development or maintenance group. The use of students as experimental participants is particularly
problematic. Students can be capable subjects, as they are often aware of the most recent methodologies, tools and techniques and are often the only available subjects. However, they lack real world experience and their understanding of software and systems can be quite peripheral and patchy. Consequently it is unclear whether or not results generated from such studies are applicable in real world situations. This highlights the need to raise the quality of empirical research, and for this community to reach a critical mass in order to disseminate best practice. Studies into the differences between students and professional programmers are clearly needed.

Conclusions

In general, computer scientists tend to adhere to an ‘ad-hoc’ evaluation of their research, publishing ideas with little or no scientific or objective assessment (Fenton, Pfleeger and Glass 1994, Glass 1994). Software engineers are particularly prone to this phenomenon, which exacerbates the academia/industry divide and delays technology transfer. Empirical software engineering research seeks to address this deficiency, by encouraging the use of objective evaluation techniques throughout the development, dissemination and transfer of new ideas. In so doing, the discipline is intrinsically multi-disciplinary, as solutions to real-world problems are necessarily holistic and complex.

As far as future directions and methodologies for empirical software engineering research are concerned, we agreed on a 3-point plan that encapsulates much of our discussion:

1. Initiate a public clearinghouse of empirical research projects (both in progress and complete), to enable researchers to compare and contrast results, repeat and/or replicate experiments, perform meta-analyses and draw valid conclusions.
2. Produce a classification scheme of research methodologies.
3. Develop a common framework for empirical research, consisting of the research methodologies, a range of applications domains, and sets of variables.

References


