Investigating the Applicability of the Evidence-Based Paradigm to Software Engineering

David Budgen
Department of Computer Science
Durham University
Durham, U.K.
david.budgen@durham.ac.uk

Stuart Charters
Department of Computer Science
Durham University
Durham, U.K.
s.m.charters@durham.ac.uk

Mark Turner
School of Computing and Maths
Keele University
Staffordshire, U.K.
m.turner@cs.keele.ac.uk

Pearl Brereton
School of Computing and Maths
Keele University
Staffordshire, U.K.
o.p.brereton@cs.keele.ac.uk

Barbara Kitchenham
School of Computing and Maths
Keele University
Staffordshire, U.K.
b.a.kitchenham@cs.keele.ac.uk

Stephen Linkman
School of Computing and Maths
Keele University
Staffordshire, U.K.
s.g.linkman@cs.keele.ac.uk

ABSTRACT

Context: The success of the evidence-based paradigm in other domains, especially medicine, has raised the question of how this might be employed in software engineering.
Objectives: To report the research we are doing to evaluate problems associated with adopting the evidence-based paradigm in software engineering and identifying strategies to address these problems.
Method: Currently the experimental paradigms used in a selected set of domains are being examined along with the experimental protocols that they employ. Our aim is to identify those domains that have generally similar characteristics to software engineering and to study the strategies that they employ to overcome the lack of rigorous empirical protocols. We are also undertaking a series of systematic literature reviews to identify the factors that may limit their applicability in the software engineering domain.
Conclusions: We have identified two domains that experience problems with experimental protocols that are similar to those occurring for software engineering, and will investigate these further to assess whether the approaches used to aggregate evidence in these domains can be adapted for use in software engineering. Our experiences from performing systematic literature reviews are positive, but reveal infrastructure problems caused by poor indexing of the literature.

Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General

WISER'06 May 20, 2006, Shanghai, China.

General Terms

Experimentation

Keywords

Evidence-Based, Systematic Literature Review, Structured Abstracts

1. INTRODUCTION

In the late 1980's and early 1990's, there was a dramatic change in medical research practices, at least for clinical studies, with the adoption of an evidence-based paradigm. More recently, the question has been asked as to whether this might usefully be employed in software engineering too [10]. Indeed as the authors observe in [10], for software engineering the adoption of an evidence-based paradigm is "potentially important because of the central place software intensive systems are starting to take in everyday life".

Although originating in clinical medicine, the *evidence-based paradigm* has subsequently been adopted (and adapted) by a number of other domains that, like software engineering, involve human-centric activities—including education, non-clinical branches of healthcare, and librarianship. In this paper we describe the processes by which we are investigating how we can draw upon the experiences and practices of the spectrum of domains now using the evidence-based paradigm, and how we are seeking to assess its potential usefulness for software engineering.

Software engineering has little tradition of using the experiences of other domains [6, 7, 9]. This is the case even for empirical studies where there is a wealth of experience available from all scientific disciplines. There may be many reasons for this, including the difficulty of performing empirical studies, the limited scope that often accompanies the results, and the lack of overall agreement by the community on suitable empirical practices. Sometimes too, it may simply be that researchers are not familiar with well-established techniques such as *protocol analysis* [13]. Furthermore, despite efforts to provide baselines for performing and report-

ing such studies, such as are described in [11], we still find public disagreement, as demonstrated by the discussion in [1, 14]. This in turn creates a problem for the use of practices such as systematic literature review, that seek to aggregate experimental results for the purposes of meta-analysis.

Indeed, Kitchenham *et al.* point out in [10] that it is important to evaluate whether software engineering is capable of adopting evidence-based practice. They observe, for example, that the success of evidence-based practice in the medical domain is partly due to the infrastructure available to practitioners in terms of medical abstracting services with sophisticated search facilities and groups that undertake systematic reviews and meta-analysis, making their results available to practitioners via not-for-profit web sites. They also identify a number of technical issues that make it difficult to adopt evidence-based practice. In particular, medicine is able to conduct rigorous, realistic experiments with protocols designed to minimise experimenter and subject-related bias. This allows evidence in clinical medicine to be aggregated using systematic literature reviews and meta-analysis. In software engineering we are unable to use such rigorous and realistic experimental protocols. This means the assumptions that hold when aggregating medical evidence do not hold when aggregating software engineering evidence.

One other issue is the question of who will benefit? For medicine, the goal has been to facilitate access by practitioners to the outcomes of research. For education, it would appear that the goal has often been more one of influencing policy-makers and decision-makers. Software engineering has elements of both and beyond: as volumes of research data grow, there is a need to give practitioners ready access to objective summaries; at the same time, the centrality of computing to everyday life requires that decision-makers and standards bodies should be kept well informed; and there are also the needs of research, where the outcomes of evidence-based studies may help focus research on the most relevant questions.

In this paper, we discuss the research that we are doing to identify domains that are both adopting evidence-based practice and that also have similar problems with accumulating empirical evidence. In Section 2, we discuss the evidence-based paradigm. In Section 3 we introduce the *Evidence-Based Software Engineering* (EBSE) project which is investigating the adoption of evidence-based practice in software engineering. In Section 4, we discuss our attempts to characterise domains that are adopting evidence-based practice, in order to assess their similarity to software engineering. In Section 5 we describe our experiences of undertaking systematic literature reviews, as well as the implications of this for the infrastructure needed to support evidence-based practice. Our conclusions and plans for future work are reviewed in Section 6.

2. THE EVIDENCE-BASED PARADIGM

To understand the role of evidence, we need to recognise that, across a wide spectrum of disciplines of study, there is a common requirement to find objective practices that can be employed for aggregating the outcomes of different empirical studies in an objective and consistent manner. The range of forms and issues is very wide: at the one end, aggregating (say) experimental studies measuring the mass of the electron is largely a matter of using mathematically-based transformations to adjust for variations in experimental conditions; whereas drawing together the results from a set of surveys, that may have employed different sets of questions and been administered to rather different populations, presents a much less mathematically tractable problem. One of the key issues underlying this difference is the role of the human in the process of data collection: in the former the only involvement is as an external *observer*, while in the latter, the human is a *participant* in the treatment itself.

This process of *aggregation* is a fundamental one for any evidence-based approach that is seeking to provide objective summaries of empirical data. The area of medicine occupies an intermediate position between the two examples above. Clinical medicine (at least) is able to make extensive use of *randomised controlled trials* (RCTs) as its experimental paradigm, and in the RCTs used in clinical studies, the role of the human is as a subject, being a *recipient* of the experimental treatment. This makes it possible to adopt the use of statistical techniques for aggregation of the outcomes, and this, along with the nature of some of the outcomes, has helped to cement the success of evidence-based medicine.

In [10], the authors suggest that the following form the key steps in evidence-based practice for both medicine and software engineering.

1. Converting the need for information into an answerable question.
2. Tracking down the best evidence with which to answer that question.
3. Critically appraising that evidence for validity, impact and applicability.
4. Integrating the critical appraisal with domain expertise and the stakeholder's values.
5. Evaluating the effectiveness and efficiency in performing steps 1–4 and seeking ways to improve these.

Steps 2 and 3 are evidently major factors in the success (or otherwise) of the approach. Both of these are being investigated by the EBSE project and discussed in the following sections, with particular emphasis upon the use of *Systematic Literature Reviews* for the first three steps.

3. THE EBSE PROJECT

The EBSE project (Evidence-Based Software Engineering) is a two-year project supported by the UK's Engineering and Physical Sciences Research Council (EPSRC), with four academic investigators and two full-time research assistants. The project began in April 2005.

The project has been able to build upon previous work that has been undertaken in a number of ways. One of these has been the series of EASE conferences (Evaluation and Assessment in Software Engineering)[1], run annually since 1996, that has helped to build up and explore issues related to the use of empirical studies in software engineering. Members of the team have also run two workshops exploring ideas about evidence-based software engineering at the STEP series of conferences in 2002 and 2003 as well as one at ICSE in 2005 [3, 4].

[1] http://ease.cs.keele.ac.uk

The goal of the project is to "investigate the viability of adopting the evidence-based paradigm for software engineering". A supporting objective is to establish the software engineering equivalent of a *Cochrane Group* that would provide systematic reviews of evidence relating to a major topic in software engineering (the proposed topic is component-based software engineering). In medicine, the *Cochrane Collaboration*[2] coordinates the practice of evidence-based medicine and within this, a Cochrane Group comprises a group of researchers who take responsibility for performing and maintaining systematic reviews associated with a particular medical topic. Hence, specific objectives for the EBSE project include:

- determining and codifying appropriate procedures for undertaking systematic reviews of evidence in software engineering;
- creating an infrastructure to support evidence-based software engineering;
- performing a number of exemplars of systematic reviews and publishing these.

In this paper we report on two specific activities that emerge from this:

- our progress with the task of *characterising* the domain of software engineering and a set of the domains using evidence-based practices, to identify practices in similar domains that can potentially be adapted for software engineering;
- our experiences with undertaking a pilot systematic literature review.

We then assess these experiences in order to draw some preliminary conclusions about the applicability of the evidence-based paradigm in the domain of software engineering.

4. CHARACTERISING THE DOMAIN

The evidence-based paradigm and its associated practices originated in medicine which is therefore viewed as being the 'classical' model of evidence-based practice. However, medical standards are based on the assumption that most empirical studies will be RCTs. RCTs are comparative trials of a proposed treatment (e.g. a new drug) against either a placebo or a current standard treatment under extremely strict conditions. In particular:

- subjects are real patients with real diseases recruited by their medical practitioners to take part in the experiment;
- neither experimenters nor subjects know which treatment a subject has received (blinding).

Such studies are virtually impossible in software engineering. We have therefore sought to make an initial study of the domains that are now using the evidence-based paradigm in some way, in order to identify the ones that are most 'similar' to software engineering and hence that would merit a more detailed examination of their practices. So, while our goal for the initial study has been to be both systematic and objective, it was not intended to provide an in-depth study of all domains that are using evidence-based practices.

[2]http://www.cochrane.org

4.1 Domain Similarity Assessment

In order to develop guidelines more suited to the type of evidence we are able to obtain in software engineering (i.e. toy laboratory experiments, uncontrolled observational studies and non-probabilistic surveys), we need to consider the procedures for aggregating evidence that have been adopted in domains that exhibit experimental limitations similar to those of software engineering. As a first step towards identifying such domains, we developed the questionnaire shown in Appendix A. (The terms used in this are defined in Appendix B.)

The comparison of software engineering with other domains using the questionnaire in the appendix is based on the following assumptions.

- Software engineering primarily makes use of laboratory experiments, observational field studies and convenience samples. It makes little use of qualitative methods and formal field experiments.
- Software engineering has major difficulties with blinding either the experimenter or the subjects because most software engineering tasks require human expertise.
- Because using a technology requires expertise, subjects need to be trained in any techniques being evaluated. This is a possible cause of bias since it may be difficult to train people to the same level of competence in different techniques.
- In real software projects, the difficulty of tasks and the quality of materials used as inputs to tasks can have a major effect on performance. Thus, software experiments ought to randomise with respect to tasks and materials as well as subjects.

We are in the process of using a questionnaire to collect information from a selection of domain experts in domains other than clinical medicine that use the evidence-based paradigm. Our initial selection of experts is based on a convenience sample of experts from Keele University and Durham University. However, we are using the "snowball" approach of asking experts we have contacted to recommend other people we might involve, and following this pilot we will use semi-structured interviews with a wider sample of experts.

4.2 Results to date

We had originally hoped that we could simply score the responses from different domains and obtain a numerical measure of similarity to software engineering. However, this was far too simplistic since the answers for each one need to be interpreted within the context of the standards for empirical studies in the domain. For example, like software engineering, organic chemistry relies heavily on laboratory experiments. However, while this is sensible for organic chemistry where the properties of chemicals do not change outside the laboratory, it is less so for software engineering where there are serious concerns that laboratory experiments are not representative of real software engineering activities. Thus, we have a similarity in experimental practice that does not reflect a meaningful similarity between the domains.

Table 1 summarises our results to date. (Note that we ourselves made the assessments for both software engineering

Table 1: Domain similarity results

Domain	Utilise field experiments or quasi-random experiments	Utilise Laboratory Experiments	Utilise other types of empirical study	Involve human expertise	Possible to blind subjects	Possible to blind experimenters
Nursing & Midwifery	No	No	Yes	Yes	No	No
Primary Care	Yes	No	Yes	Yes	Yes	Yes
Criminology†	No	No	Yes	Yes	No	No
Organic Chemistry†	No	Yes	No	Yes	No	No
Empirical Psychology	No	Yes	Yes	Yes	Yes	Yes
Clinical Medicine	Yes	No	Yes	No	Yes	Yes
Education	No	No	Yes	Yes	No	No
Software Engineering	No	Yes	Yes	Yes	No	No
† These used an earlier version of the questionnaire						

and clinical medicine.) It confirms that software engineering and organic chemistry share many similarities in terms of study types and characteristics. As regards study types, neither domain performs RCTs and both rely upon laboratory experiments. They also involve human expertise and share difficulties in blinding subjects and experimenters. However, the major domain difference between organic chemistry and software engineering is that in organic chemistry RCTs are unnecessary because laboratory experiments are sufficient to establish research results, whereas in software engineering, RCTs are not performed because of technical difficulties. This is why organic chemistry does not utilise other types of study that are frequently used in software engineering, such as field observations and surveys.

Nursing & midwifery, education and criminology all share many characteristics with software engineering. Unlike software engineering they do not use laboratory experiments, but it could be argued that software engineering ought to rely far less on laboratory experiments. Like software engineering, none of these can undertake RCTs and therefore rely to some extent upon other types of empirical study such as field observational studies and surveys. Also, problems with blinding and human expertise occur in all three domains. Overall, our current results suggest that nursing & midwifery, and education are the domains that are most similar to software engineering. We will therefore be particularly interested in the approaches used in these domains to assessing the quality of primary studies such as surveys and observational studies and aggregating evidence from such studies.

To further illustrate how the chosen domains compare, Figure 1 compares the characteristics of software engineering with those of nursing & midwifery, while Figure 2 compares them with those of primary care. (In both cases, the bars with diagonal lines represent software engineering characteristics.) Table 2 provides the keys to the labels used in these charts. The scales used are based upon the weightings employed in the questionnaire.

Figure 1 indicates that Nursing & Midwifery has similar problems to software engineering with respect to the difficulty it has in performing rigorous experiments (i.e. randomised field experiments and quasi-random field experiments). Unlike software engineering, the domain does not perform laboratory experiments, it gathers evidence from more rigorous surveys (usually based on sampling, not convenience) and document analysis. In contrast, Figure 2

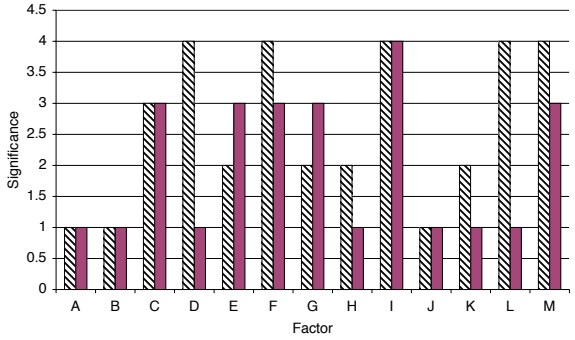

Figure 1: Contrasting Software Engineering with Nursing & Midwifery

shows that software engineering and primary care differ on almost all factors.

When viewed in this way (which includes a ranking of the degree to which particular forms are employed), there is obviously a wider degree of variation than might be expected from the results shown in Table 1. However, some care needs to be taken here, since the values used in each chart are essentially the outcome of one (experienced) researcher's view of a particular domain and so should be treated as indicative rather than definitive.

Nonetheless, the most evident conclusion from this exercise is that the evidence-based paradigm is currently employed with a widely-varying set of experimental profiles.

4.3 Future Plans

Our initial exploration has revealed that the use of a questionnaire, while helpful, can result in misunderstandings and that it is best if the knowledge elicitation process is moderated in some way. We are therefore in the process of repeating our initial study as follows.

- Using a structured interview driven by an expanded set of questions, which will allow scope for clarification as well as exploration of any issues that arise.

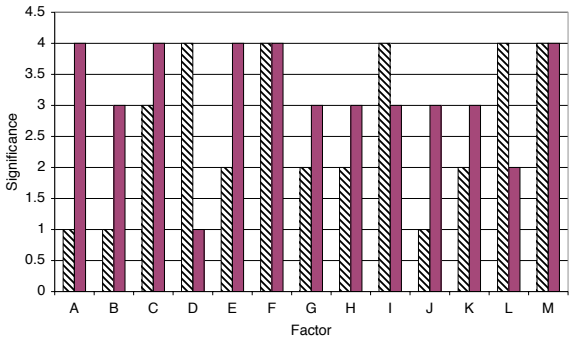

Figure 2: Contrasting Software Engineering with Primary Care

Table 2: Keys used in the Charts

Key	Experimental Technique
A	Field Experiments
B	Quasi-Random Field Experiments
C	Field Observational Studies
D	Laboratory Experiments
E	Surveys based on Random Sampling
F	Surveys based on Convenience Samples
G	Document Analysis
H	Discourse Analysis
I	Human Expertise a factor
J	Possible to Blind Subjects
K	Possible to Blind Experimenters
L	Training Bias
M	Randomisation for Materials

- Interviewing at least two experts from each domain in order to limit possible bias—if there are any inconsistencies we can explore these with the experts and if necessary, obtain further views.

We will be re-interviewing those who helped us with the initial questionnaire wherever possible.

5. SYSTEMATIC LITERATURE REVIEWS

The first three steps of the evidence-based paradigm as described above, essentially constitute a *systematic review* of the literature related to a particular phenomenon. (Although these are strictly 'systematic *literature* reviews', we will use the term 'systematic review' in the rest of this paper in the interest of readability.) While the use of systematic reviews is well established in medicine [8], and (at least partially) established in information systems research [12, 15], such secondary studies are not commonly performed in software engineering and hence one of the tasks undertaken by the project has been to establish suitable procedures, to apply these to some sample questions, and to evaluate their effectiveness. In the rest of this section we briefly discuss our work on establishing procedures and then describe our experiences to date with conducting such a review.

5.1 The Systematic Review Process

The process involved in conducting a systematic review is organised around three major phases: planning the review; conducting it; and documenting the outcomes. Each of these involves a sequence of specific steps, as illustrated in Figure 3 and described more fully in [9].

An important element in conducting a systematic review is to develop a *protocol*, which should be agreed before the study begins. The role of the protocol is to establish all of the criteria under which the study is to take place, such as keywords and combinations, the range of sources to be included (as well as the start date), procedures to be followed (for example, the tasks of different reviewers), and the information needing to be abstracted. A protocol can be a substantial document (that developed for the review described below runs to over twenty pages), and any revisions must themselves be documented. The protocol is a key element in establishing both the validity of a study and also the limitations upon its scope and hence the scope of its outcomes.

5.2 Results to date

To establish the practicalities of performing systematic reviews in the software engineering domain, a study of the *Technology Acceptance Model* (TAM) [5] is being conducted to assess its predictive capabilities. Our choice of the TAM stems from the experience of a research student being supervised by one of the authors of this paper (BAK), who found unpredictable results from its use. This in turn led to a pilot systematic review by another student, with results that encouraged us to investigate this further as one of the topics for the EBSE project.

Conducting such a systematic review allows us to explore the infrastructure available for providing access to the relevant software engineering publications. More comprehensive details of our experiences so far with this are reported in [2], but three key findings that emerged (based partly on a pilot study and also on our development of the protocol for the review) are that:

- software engineering has no comprehensive abstracting service;
- it also has no comprehensive indexing service;
- and there is no comprehensive pre-print facility.

When searching for papers using various library systems, such as CiteSeer, and those provided by IEEE and ACM, we encountered many issues relating to the ordering of keywords and the use of boolean operators. Another, related, issue was the poor quality of many of the abstracts. In order to determine whether a paper met the criteria established in the protocol, it was often necessary to read further parts of the paper such as the introduction and the conclusions.

Perhaps less surprisingly, despite the topic itself, the percentage of papers that were based upon empirical studies and that also met the inclusion criteria was very low.

As indicated above, the protocol for this study is also a substantial document. Considerable time and effort has had to be expended on developing and maintaining it, and its value to the team in defining the parameters of the study and raising questions about the processes involved has been immeasurable.

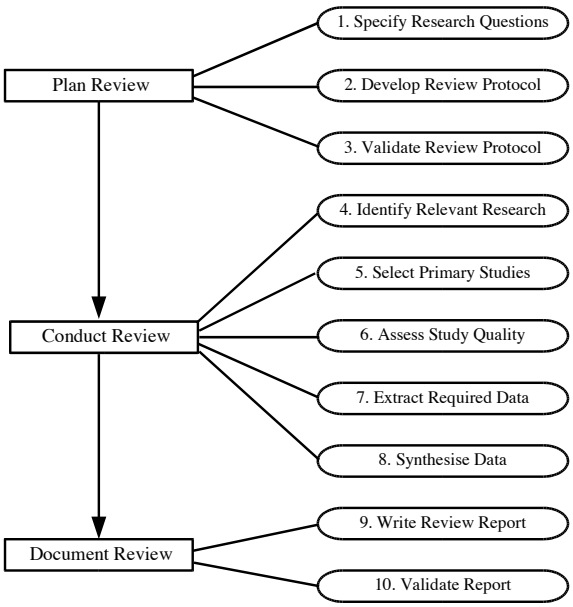

Figure 3: Systematic Literature Review Process

5.3 Future Plans

The work on this aspect has led to plans for both additional reviews as well as other studies. For future reviews we plan to include at least one review topic that will involve study of the 'grey literature' (workshop and conference papers, technical reports etc.) to investigate the issues of searching and extracting data from such sources. Our experiences with the problems of searching have also led us to begin developing protocols for further studies in the area of abstracting and indexing papers.

6. CONCLUSIONS

Our aim in this paper has been to describe our experiences with the process of adapting the practices of evidence-based studies to software engineering. Adapting a paradigm for use in a new and different domain does require some understanding of the differences that arise in that domain. The work of our project, that we have reported here, has therefore involved seeking ways to characterise the empirical aspects of the software engineering domain as well as to adapt the practices of systematic review to these characteristics. We have also described our experiences so far with applying these and conducting a systematic review.

Our study of domain characteristics indicates that the evidence-based paradigm has been successfully adopted within a wide range of disciplines. While (not surprisingly) none of these has characteristics that closely match those of software engineering, the degree of variation exhibited, and the variety of experimental forms employed, does at least indicate that there are no obvious reasons why this should not be appropriate for software engineering research. It has also identified a number of domains where the characteristics are close enough to suggest that we should examine how some of their practices might be deployed in software engineering. Interestingly, be perhaps not surprisingly, these tend to be at the 'social' end of the subject spectrum rather than the 'science' end.

This theme of adaptation is continued in the second part of our work, which is that of developing appropriate practices for systematic reviews in software engineering, creating a support infrastructure, and conducting some field trials of these. While our experiences with developing protocols for this are generally good, the experiences from our pilot study as well as from the protocol development reveals significant problems with indexing and abstracting of the software engineering literature.

Overall, our work so far is encouraging as regards use of the evidence-based paradigm in software engineering. However, it also indicates that its widespread application will only be practical if we as a discipline are willing to make substantial revisions to software engineering's infrastructure, ranging from the adoption of structured abstracts to the provision of much better searching mechanisms.

In general, we conclude that moving a technology from one discipline to another requires a careful assessment of the context in which the technology works successfully. If there are differences in context, it is important to understand the implications of those differences and to properly adjust the technology to the constraints of the new domain.

7. ACKNOWLEDGMENTS

The authors would like to thank all those who helped us with characterising the different domains, and the support of the EPSRC for funding the EBSE project (EP/C51839X/1). Dr Mohamed Khalil was a research assistant on the project for the first five months, and we also wish to acknowledge his very valuable contributions.

8. REFERENCES

[1] D. Berry and W. Tichy. Comments on "formal methods application: An empirical tale of software development". *IEEE Transactions on Software Engineering*, 29:567–571, 2003.

[2] O. Brereton, B. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Employing systematic literature review: An experience report. Technical Report TR05/01, Department of Computer Science, Keele University, 2005. Submitted for Publication.

[3] D. Budgen, J. Bøegh, and A. Mohan. Organising Evidence to support Software Engineering Practice. In *Proceedings 11th International Workshop: Software Technology and Engineering Practice*, pages 25–32, 2004.

[4] D. Budgen and B. Kitchenham. Realising Evidence-Based Software Engineering: A Report from the Workshop held at ICSE 2005. *Software Engineering Notes*, 30(5):44, 2005. Full report published electronically by ACM Press.

[5] F. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technologies. *MIS Quarterly*, 13:319–340, 1989.

[6] R. Glass, V. Ramesh, and I. Vessey. An Analysis of Research in Computing Disciplines. *Communications of the ACM*, 47:89–94, June 2004.

[7] R. Glass, I. Vessey, and V. Ramesh. Research in software engineering: An analysis of the literature. *Information & Software Technology*, 44:491–506, 2002.

[8] K. Khan, R. Kunz, J. Kleijnen, and G. Antes. *Systematic Reviews to Support Evidence-Based Medicine: How to review and apply findings of healthcare research*. Royal Society of Medicine Press Ltd., 2003.

[9] B. Kitchenham. Procedures for undertaking systematic reviews. Technical Report TR/SE-0401, Department of Computer Science, Keele University and National ICT, Australia Ltd, 2004. Joint Technical Report.

[10] B. Kitchenham, T. Dybå, and M. Jørgensen. Evidence-based software engineering. In *Proceedings of ICSE 2004*, pages 273–281. IEEE Computer Society Press, 2004.

[11] B. Kitchenham, S. L. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. E. Emam, and J.Rosenberg. Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Transactions on Software Engineering*, 28:721–734, 2002.

[12] B. J. Oates. *Researching Information Systems and Computing*. SAGE Publications, 2006.

[13] S. Owen, D. Budgen, and P. Brereton. Protocol Analysis: A Neglected Practice. *Communications of the ACM*, 49:117–122, Feb. 2006.

[14] A. K. Sobel and M. Clarkson. Response to "comments on 'formal methods application: An empirical tale of software development' ". *IEEE Transactions on Software Engineering*, 29:572–575, 2003.

[15] J. Webster and R. Watson. Analysing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, 26:xiii–xxiii, 2002.

APPENDIX

A. QUESTIONNAIRE TO DETERMINE SIMILARITY TO SOFTWARE ENGINEERING

1. Does the domain undertake the following types of empirical study:
   - Field experiments (with random allocation)
     Frequently (4)/ Occasionally (3)/ Rarely (2)/ Never (1)
   - Quasi-random field experiments
     Frequently (4)/ Occasionally (3)/ Rarely (2)/ Never (1)
   - Field observational studies (including case studies)
     Frequently (1)/ Occasionally (2)/ Rarely (3)/ Never (4)
   - Laboratory experiments
     Frequently (1) /Occasionally (2)/ Rarely (3)/ Never (4)
   - Surveys based on random sampling of a population
     Frequently (4)/ Occasionally (3)/ Rarely (2)/ Never (1)
   - Surveys based on convenience samples
     Frequently (1)/ Occasionally (2)/ Rarely (3)/ Never (4)
   - Document analysis (Content analysis)
     Frequently (1)/ Occasionally (2)/ Rarely (3)/ Never (4)
   - Discourse analysis
     Frequently (1)/ Occasionally (2)/ Rarely (3)/ Never (4)
2. Is human expertise a factor in techniques, tools or methods that are evaluated in empirical studies?
   Always (1) /Often (2)/Sometimes (3)/Rarely (4)/Never (5)
3. Is it possible to blind subjects as to the techniques, tools, methods that are evaluated in empirical studies?
   Always (5)/Often (4)/Sometimes (3)/Rarely (2)/Never (1)
4. Do experimenters use procedures to blind experimenters as to the allocation of subjects to the techniques, tools, methods that are evaluated in empirical studies?
   Always (5)/Often (4)/Sometimes (3)/Rarely (2)/Never (1)
5. Is subject training a significant cause of potential bias?
   Always (1)/Often (2)/Sometimes (3)/Rarely (4)/Never (5)
6. Is randomisation required for materials, tasks and settings as well as subjects for valid generalisation?
   Always (1)/Often (2)/Sometimes (3)/Rarely (4)/Never (5)

B. DEFINITIONS

The terms used in the questionnaire are defined as follows:

- **Field experiments with random allocation** are formal experiments undertaken in a software engineering company by software engineers performing standard project activities (i.e. excluding experiments taking place as part of training exercises). Participants are allocated to the software engineering method at random. This would be equivalent to a **Randomised Controlled Trial** (RCT) in medicine.
- **Quasi-random field experiments** are studies where randomisation is not possible for practical or ethical reasons. They are used when different groups of interest occur naturally so that no random assignments can be performed, such as successful or unsuccessful projects.
- **Field observational studies** are observational studies undertaken in a software engineering company. They are neither experiments nor quasi-experiments and may be qualitative rather than quantitative. These include correlation and regression studies such as cost estimation studies.
- **Laboratory experiments** are randomised experiments performed by subjects (whether students or practitioners) that are not part of a real software engineering project.
- **Surveys** are questionnaire based or interview based exercises where opinions or data are extracted from a relatively large number of participants. Surveys based on random sampling are based on having a defined population and ensuring that each member of the population has an equal chance of being requested to participate. Convenience sampling is based on non-random sampling. Participants are those people accessed by the researchers who choose to take part.
- **Document analysis**, and in particular, **content analysis**, is a semi-qualitative method that codes words or phrases in a document and counts the number of incidences of the coded phases/words.
- **Discourse analysis** is a range of methods relating to a theoretical position that are applied to different forms of discourse, including interviews.