Is Evidence Based Software Engineering mature enough for Practice & Policy?

David Budgen
School of Engineering & Computing Sciences
Durham University
Durham, DH1 3LE
U.K.
david.budgen@durham.ac.uk

Barbara Kitchenham & Pearl Brereton
School of Computing & Mathematics
Keele University,
Staffordshire, ST5 5BG
U.K.
{b.a.kitchenham;o.p.brereton}@cs.keele.ac.uk

Abstract

Since their use was first proposed in 2004, evidence-based practices, and particularly systematic literature reviews, are becoming widely used in empirical software engineering. We describe the key concepts of evidence-based software engineering (EBSE) and its use of systematic literature reviews (SLRs) and mapping studies. We then report on the extent to which evidence-based studies are changing our ideas, and the extent to which they have so far addressed major software engineering themes and topics. We discuss how far this can already be used to inform policy and practice, and what further developments could usefully reinforce this role.

1 Introduction

In 2004, Kitchenham, Dybå and Jørgensen proposed that software engineering would benefit from adopting the evidence-based paradigm as a means of enabling research to inform both practice and policy [22].

The purpose of evidence-based studies is to objectively and systematically find and aggregate all relevant evidence about a particular phenomenon. Its origins lie in clinical medicine where it arose in response to a challenge from the Epidemiologist, Archie Cochrane, who was concerned that clinical practice needed a sounder empirical foundation and the means of resolving what could be large volumes of potentially conflicting evidence. Clinical medicine has remained at the forefront of this development, as evidenced by the website of the not-for-profit *Cochrane Collaboration*[1] and by its influence upon both teaching and practice in clinical medicine. The adoption of evidence-based medicine (EBM) was also stimulated by a number of early evidence-based studies that demonstrated results that overturned 'common knowledge' and practice.

Because the role of the human in clinical medicine is essentially one of a *recipient* of a treatment, it is possible to undertake *Randomised Controlled Trials* or RCTs, which in turn makes it possible to employ statistical methods when aggregating the results from different trials. However, since patients with one problem may well have others, RCTs may not always produce consistent outcomes, and the application of the outcomes may not always be straightforward. Despite these factors, the use of the evidence-based paradigm in clinical medicine still forms a 'gold standard' that researchers in other disciplines seek to emulate.

Many healthcare-related disciplines now employ evidence-based practices. although they often need adaptation, not least because when the skills of the practitioner becomes a factor, it is impractical to conduct fully rigorous RCTs. We report on how this has been managed in [7].

Evidence-based studies have also been undertaken in the domains of Education and Social Policy, where the purpose has been to inform policy in particular. (The current U.K. government has been keen to emphasise the adoption of evidence-based policies, although the term 'evidence-informed' may often be more appropriate.) There is now an established journal for this (*J. of Evidence & Policy*) and a number of institutions have been funded to disseminate evidence-based knowledge in such areas as education and healthcare.

For software engineering, the tradition is still largely one of reliance upon expert-based reviews, and an associated reluctance to consider empirical evidence [20]. However, while this is still relatively early days for the development of the paradigm in software engineering, quite solid progress has been made, and there is a growing community of researchers who are working in this area. The aim of this paper is therefore to report on this progress, to identify where some of the results might conflict with expectation and practice, and to address the question as to whether it is now

[1] www.cochrane.org

sufficiently established to be able to inform software engineering practices, organisational policies and international standards—all of which are currently largely formulated on the basis of expert opinion. We also examine what future developments might best enable such a paradigm change to occur.

2 Evidence-Based Software Engineering

In translating the ideas of the evidence-based paradigm to the software engineering domain, Kitchenham *et al.* defined the goal of evidence-based software engineering (EBSE) as being [22]:

> *To provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision making process regarding the development and maintenance of software.*

2.1 Realising EBSE

In [10], and based upon the analogy with clinical medicine, the same group of researchers identified a five-step process for EBSE as being:

1. Converting a relevant problem or information need into an answerable question.
2. Searching the literature for the best available evidence to answer the question.
3. Critically appraising the evidence for its validity, impact and applicability.
4. Integrating the appraised evidence with practical experience and the values and circumstances of the customer to make decisions about practice.
5. Evaluating software development performance and seeking ways to improve it.

This can be viewed as an interpretation of the *Systematic Literature Review* (SLR), which has been widely adopted as the main tool of the evidence-based paradigm [26] and has subsequently been incorporated into the *Guidelines* document, initially produced by Kitchenham for software engineers in 2004, and later updated in 2007 [21].

2.2 Some limitations of EBSE

While conceptually at least, EBSE has a sound basis and the potential to deliver sound, unbiased appraisal of software engineering processes and products, even from the start it has been recognised that there are factors that limit this potential. Some are implicit in the nature of the discipline, while others arise more from resource limitations. Limitations include:

- The role of the participant in empirical studies (and especially experiments). Since this is almost always skill-based, it is impossible to apply techniques such as double-blinding (of participants and experimenters) that are considered an essential feature of medical trials.
- Empirical studies in software engineering make extensive use of laboratory experiments rather than field trials (again, unlike the medical analogy), which makes it difficult to generalise the outcomes. Many of them also use students as participants—and determining whether or not this permits generalisation is a difficult issue.
- For medicine, the evidence is clearly intended for use by clinicians, who will use the outcomes to aid diagnosis and treatment. For education, it is usually the policy-makers at national or possibly regional levels who are the end users. This is less clear for EBSE since in software engineering decisions may be made at many different levels.
- The digital libraries available for software engineering researchers do not provide particularly good (or consistent) searching facilities when seeking papers on a particular topic. In particular, they tend to have quite different interface structures when it comes to determining what elements (title, abstract, body) are to be searched, as well as the forms used for compounding the search terms. This in turn makes it necessary to interpret the chosen search strings differently for each engine.

There are also related issues about the quality and quantity of the 'primary' empirical studies used as inputs to secondary studies such as SLRs.

2.3 Recent progress with EBSE

Much recent progress with EBSE is fairly closely tied up with the various developments that have occurred with empirical software engineering.

Researchers such as Basili have done much pioneering work in empirical software engineering, and there are now established conferences (such as *Empirical Software Engineering & Metrics* (ESEM) and *Evaluation & Assessment in Software Engineering* (EASE)) as well as a journal (*Empirical Software Engineering*). However, issues such as reporting standards still provide a problem for researchers.

In terms of EBSE itself, there is considerable ongoing activity (mainly in Europe and Scandanavia) [18]. Two recent developments are:

- The institution of a separate section for systematic literature reviews in the journal *Information & Software Technology*, recognising both that this type of study may need to be treated a little differently from the standard journal paper, and also the need to build up the international body of knowledge.

- The creation of a web site[2], supported by our own research projects, intended to act as an intenational resource for planning and reporting evidence-based studies (and also primary studies to some degree).

Having discussed some of the processes and mechanisms involved in EBSE, we now go on to examine the knowledge about SE that has so far been obtained through these means.

3 Current state of EBSE knowledge

In clinical medicine, adoption of the evidence-based paradigm in the late 1980s and 90s led to dramatic changes in both research and practice, as well as demonstrating that the clinical judgement of experts compared unfavourably with the results of systematic reviews. For software engineering, as indicated above, the role of the human as *participant* rather than *recipient* in our primary studies makes it more difficult to produce strongly definitive outcomes, but experiences indicate that:

1. Use of the evidence-based paradigm is appropriate and useful for software engineering and can produce repeatable results [24]. However, as discussed above, there are constraints that are imposed by the nature of the discipline and the way that our electronic repositories are organised, that need to be recognised when undertaking secondary studies [3].

2. Although empirical data, in the form of the results from different primary studies, is available for many topics in software engineering (in varying amounts—and quality), secondary studies (such as systematic literature reviews or mapping studies) that aggregate the outcomes from these, have so far only been performed for a few of the possible topics (see below).

3. As for clinical medicine, if less dramatically, where secondary studies have been undertaken, a number of them have demonstrated that a systematic approach to assembling evidence may well produce results that contradict current practices and opinions in the research and practitioner communities. Some examples of these are:

   (a) For many years the original Standish CHAOS report has been widely cited as confirming the parlous state of software engineering, with the majority of projects failing and significantly overrunning. As background to a study of Norwegian projects, Moløkken-Østvold et al. looked at all empirical studies that reported failure rates and over-runs [25]. They found that the Standish report was completely out of line with other contemporary studies, which indicate that average overruns are in the region of 30%, a figure which has not changed much in the last 20 years. The methodological problems with the original Standish report (including sampling by explicitly soliciting reports of failing projects) were the subject of another study [16].

   (b) Jørgensen investigated the results of studies that compared algorithmic cost estimation models with expert opinion estimates [14]. Although research on cost estimation for the past thirty years has been premised upon the inferiority of estimates based upon expert opinion, he found no compelling evidence that algorithmic models are more accurate than expert opinion estimates. In fact, one third of the studies identified algorithmic models as best, one third found expert opinion-based estimates best, and the remaining third found no difference. In a later study he discussed factors that might determine when to choose between an algorithmic model or expert opinion [15].

   (c) Dybå and Dingsøyr have reviewed holistic empirical studies of agile methods (the results of applying an integrated set of agile methods) [9], while Hannay et al. have reviewed empirical studies of pair programming [12]. Both studies suggest that we know a lot less about agile techniques than is popularly thought. Dybå and Dingsøyr highlight the general lack of trustworthy empirical studies, particularly related to management methods—they only found one study investigating SCRUM. Hannay et al. found limited evidence that agile methods exhibit greater quality and are faster than conventional methods but are less productive. However, all of these effects are small, and detailed meta-analysis casts some doubts upon their reliability.

   (d) Our own study of the TAM (Technology Acceptance Model), draws upon 73 primary studies [28]. This has demonstrated that the lack of objective measures for technology use in many studies results in studies of the TAM often measuring *perceived use* rather than *actual use* and

[2]www.ebse.org.uk

has also demonstrated that some of the key components of the TAM are not good predictors of actual use.

4. A number of *mapping studies*, sometimes termed *scoping reviews* [26], a form of secondary study that seeks to scope our empirical knowledge about a topic, have been undertaken as part of our own studies. While in other disciplines these form a useful starting point for a fuller systematic literature review (SLR), when performed for software engineering they have largely revealed the extent of the gaps in our empirical knowledge. Examples have included the UML [27], object-oriented design [2] and design patterns [31]. We discuss some examples from these in more detail in the next section.

Within software engineering, the SWEBOK [1] has become established as an authoritative reference, drawing widely upon expert opinion and experience, and providing an overview of the topics and practices that currently form the knowledge-base for software engineering research and practice.

Table 1 shows the current state of evidence-based experience when mapped on to the structure of the SWEBOK. We have included papers from our original tertiary study of secondary studies [18] together with those additionally identified in the extended-search tertiary study undertaken as part of the EPIC project [19]; and those that we are aware of, but have not yet been formally reported, such as those described in [6]. Also, for this purpose, we have not distinguished between mapping studies (that essentially identify how far a topic is addressed by primary studies) and a full SLR (which aggregates these studies to address a more focussed issue). We have also omitted studies that report research trends rather than addressing questions relating to specific software engineering topics. While the difference between the totals for the first two columns indicates a task well beyond any single project, closer inspection indicates that there is ample scope for selective extension.

SWEBOK heading	No. of sections	Number of sections with SLR coverage	Total No. of SLRs
2. Software Requirements	28	2	2
3. Software Design	30	2	2
4. Software Construction	14	2	3
5. Software Testing	68	3	3
6. Software Maintenance	28	1	1
7. Software Configuration Management	28	0	0
8. Software Engineering Management	24	3	10
9. Software Engineering Process	20	3	7
10. Software Engineering Tools & Methods	13	1	1
11. Software Quality	24	2	3
Total	277	19	32

Table 1. Evidence-based cover of the main SWEBOK topics

4 Informing Practice & Policy

Table 1 shows the coverage of topics against the main SWEBOK headings (chapters and sections), although since the chapters are each edited independently, there is some inconsistency about the structuring at the level of sections that makes it hard to compare chapters. Also, some sections are introductory or concerned with context, and so unlikely to make significant reference to empirical studies. However, the table does clearly show that so far, we only have limited coverage. In the next section we will consider the consequences of this, but for this section, we address the question of what sort of things are covered by existing studies, illustrating them in part from some of our own studies.

4.1 How studies are organised

Although the SWEBOK provides a useful overall structure, based on chapters, that identifies major areas of software engineering, it emerges that the structure becomes less useful when we go below that. At the level of the individual section, the SWEBOK tends to discuss individual techniques, either in isolation, or within some framework. As an example, if we look at software testing, different techniques are grouped based on strategy, and so for a strategy such as *fault-based techniques*, we find two sub-categories of *error guessing* and *mutation testing*. Each of these is discussed as a separate strategy. (We should add that this is an observation rather than a criticism, and indeed, most textbooks are organised in this way.)

However, when we look at the form of primary studies, these are almost always *comparative* in nature. So, for example, if we stay with testing, an experiment might compare the effectiveness of two techniques, possibly using different strategies. Again, this is perfectly reasonable, as most experimental studies (and experiments are generally the most common form used) compare some treatment with a 'control'. If the treatment is some testing technique, then it is quite reasonable that we might use a different technique as the 'control'. In a similar manner, a study of model-based cost estimation might make comparison with expert judgement-based estimation.

In each case, the structure commonly adopted is appropriate to the context, but unfortunately, it makes it harder to draw the two together. It is also largely a reflection of the role of the participant in our studies. For clinical medicine where participants are recipients, the 'control' might be that participants receive no treatment—usually delivered via a placebo to preserve the double blinding. Where the participant performs tasks and uses skill for this, then there is no real equivalent to a control of no treatment.

Unfortunately, it can potentially lead to a possible combinatoric explosion of comparisons if different groups of experimenters choose different treatments for their comparisons. While not a significant effect so far, it does argue that for secondary studies to be effective, we need to encourage more replication studies (and in turn, for journals and conferences to cope with publication of such studies.) There are also other factors that might help bridge between the outcomes from different studies (such as the use of a common outcome measure such as time).

A related issue to this is the use of case studies. This term (and also the word 'experiment') is often used very casually in software engineering papers—and is commonly used to describe what should really be categorised as experience or observation. However, carefully planned case studies, as documented by Yin [30], can be used in software engineering as we have demonstrated in [19]. In particular, this form can provide a much more rigorous option than opportunistic observations, especially in situations where it is impractical to undertake a controlled experiment (which often applies to 'field' studies).

4.2 Example studies

Here we briefly discuss two example studies in order to illustrate the issues discussed above and to help provide an idea of scale. We only provide very outline details, as these are both reported elsewhere.

4.2.1 Design Patterns

Software design patterns are widely promoted as an aid to design and maintenance, and since its publication in 1995, the book by the *Gang of Four* (usually abbreviated to GoF) has provided something of a standard [11]. There are many other patterns in use, but the 23 patterns discussed in that book are probably the most well-known and widely adopted.

When we began a mapping study to investigate how widely patterns were 'validated' empirically, we therefore expected that these would be the ones that experimenters would use, and this was very much born out by our experiences [31]. However, what we also found was that:

- Although many papers discussing patterns exist, the number of papers reporting studies that investigated the effectiveness of patterns in any way was very small. We performed a very rigorous search using electronic databases, manual search and snowballing (following up references from relevant papers found by the other techniques). This identified 480 papers, of which 181 were in some way empirical. However, analysis of these eventually reduced to a set of only 11 papers that reported on good quality empirical studies about patterns.
- Only three of the patterns in the GoF (*Composite*, *Observer* and *Visitor*) had been studied very extensively—without any strong agreement between studies. Also, only 13 of the 23 patterns had been studied at all through experiments.
- Many of the studies were performed by two research groups.
- Where studies were replicated, there were changes to the experimental procedures that made it difficult to draw any comparisons with the original studies (possibly, to help ensure that the work got published!).

We also examined what might usefully be termed the 'observational' papers that reported on the use of patterns in practice (see our previous comments about case studies). Unfortunately, the quality of reporting in those found led to only three being of real use—and unfortunately these tended to report on a different subset of patterns to those covered by the experiments (there was a small overlap). We will return to this issue later, as we did identify one very good example that demonstrated that, even when not formally structured as a case study, such reports could have real value if well organised [29].

4.2.2 Cost Models

As indicated, this is one area where a number of reviews have been performed, examining various aspects. However, for our example here we will briefly examine one published review that was undertaken by Magne Jørgensen and Martin Shepperd [17], entitled "A Systematic Review of Software Development Cost Estimation Studies".

In many ways, this is again something of a mapping study, since it addressed some eight research questions that were concerned with identifying such issues as how cost estimation studies were performed, which were most studied, and where they were published. Their approach was somewhat different to the study described in the previous sub-section and that, together with their findings, is outlined below.

- Rather than use electronic searching, they performed a manual search of the titles and abstracts in more than

100 journals considered likely to be potentially relevant. They found a total of 304 papers in 76 of these journals (up to the date April 2004). They explicitly excluded conference papers for this study.

- Unlike the patterns studies, they found relatively few researchers had longer-term interests in this area, although those researchers did publish across a wide spectrum of topics.
- The distribution of papers across research topics was uneven, with 61% of papers addressing the introduction and evaluation of estimation methods. (Size measures formed the next largest group with 20% of papers.)
- They examined the different estimation approaches studied and found that regression-based approaches dominated (this includes "most common parametric estimation models, e.g. the COCOMO model").
- There was relatively little discussion of expert judgement-based approaches (only 15% of papers). The authors suggest that the "relative lack of focus upon expert judgement-based approaches suggests, we believe, that most researchers either do not cooperate closely with the software industry, or that they believe it is better to focus on replacement, rather than improvement, of judgement-based approaches currently used in industry".

Not surprisingly, one of the recommendations from this study was that researchers should conduct more studies on estimation methods commonly used by the software industry. (A point followed up in [14].)

5 Discussion

For this section we address the following three questions:

1. Is EBSE mature enough to inform practice and policy?
2. If so, what are the barriers to making use of it more widely?
3. How can such barriers be overcome?

Since this is a workshop paper, we see the discussion of these questions, informed by the previous sections, as forming our major contribution to the workshop.

5.1 Is EBSE mature enough to inform practice and policy?

Since 2004 there has been considerable investment of time and effort in exploring the practicality of EBSE, and this has been focused on two main aspects.

- The *methodological* aspects such as adapting the SLR process to software engineering and identifying how both primary and secondary studies can best be performed in this domain. Much of our own work has been in this area, and in an earlier paper we reviewed progress and suggested that there was no evidence that it was *not* appropriate, although still immature in some ways [4].
- The *application* of EBSE in the form of mapping studies and SLRs, with a major contribution coming from the Simula Laboratory in Norway, although many other research groups have also contributed studies, including ourselves.

The preceding sections have addressed various aspects of EBSE, including the scale of its adoption so far. We would argue that there is now ample evidence that *methodologically*, EBSE is sufficiently mature enough in its practices to be able to provide a much more solid basis for decisions than expert reviews and opinions. However, this does need to be tempered by the observation that the set of existing studies is limited in scope, and hence there may only be limited scope for 'off the shelf' use of the existing studies (probably the main exception has to be cost modelling, where there is ample usable material).

5.2 What are the barriers?

In [4] we identified some technical issues that we saw as constraining the acceptance of EBSE for practice and policy. Some of these are activities that the research community needs to address, and indeed, is addressing—including:

- Better *abstracts* to help analysts decide upon inclusion and exclusion criteria (and we applaud the recent decision of *Information & Software Technology* to adopt the use of *structured abstracts* [5]).
- Better quality *reporting* of studies, including observational studies.
- Refinement of our own *methodological framework*, (which is ongoing).
- Better *searching* facilities in our electronic databases, an issue which still needs to be addressed by the providers, who still seem unaware of the need to make systematic searches.

As indicated, progress has been made with two of these (abstracts and methodological framework). The task of encouraging better reporting is an ongoing one—there are some quite widely adopted suggestions for experiments [23], and we have also made some tentative suggestions for observational studies, based on our experiences with using these for the study on design patterns [8]. The outstanding task remains that of improving searching, which is an inconvenience rather than a factor that prevents EBSE in any way.

So, what other barriers do we see at this point? We suggest that the following four are ones that need to be more fully explored.

- At present, researchers tend to investigate *topics* that are either of interest to them, or where they are aware that there is a body of primary studies that can be used. The result is that many of the secondary studies that we have identified are ones that address research issues rather than practical concerns, and there is also a rather uneven spread of studies across the main areas of software engineering, using the SWEBOK as our baseline.

- Papers reporting SLRs and mapping studies rarely provide any guidelines that can be used directly by practitioners and policy-makers. This is not unique of course, and in other disciplines there are organisations that undertake such a role. In the U.K., the *Centre for Research and Dissemination* at York is responsible for disseminating the outcomes of healthcare-related studies in a form that is usable by medical practitioners and others, and there is the *Evidence for Policy and Practice Information and Coordinating Centre* (EPPI-Centre) that undertakes this role for social science (including education), based in London.

- Empirical studies still tend to make extensive use of laboratory studies rather than field studies, and (for obvious reasons) students are often used as the participants in these experiments. Used carefully,the use of students as surrogates for practitioners can be justified (for an example, see [13]), but even where this can be done, there remains a problem of perception from outside academia. For EBSE to be widely accepted, we almost certainly need to find ways of increasing the proportion and quality of field studies (again, see our earlier comments about case studies), and also of using these to validate those studies that do use students.

- Lastly, and complicating the previous points, unlike domains such as clinical medicine and healthcare, knowledge about processes in particular, may well be viewed as being *proprietary* in software engineering, since it may help an organisation with maintaining competitive advantage. So, while this does not impeded academic studies, particularly those using students, it does make it harder to obtain useful observational data and to obtain collaboration for field studies in general.

5.3 How can the barriers be overcome?

For educational studies, government bodies (at least in the U.K.) do sponsor systematic reviews on questions of interest to policy-makers, and some healthcare reviews are likewise commissioned by government agencies. There would clearly be scope for similar practice in software engineering, both from government and also commercial organisations, although in the latter case, it would of course raise the question of commercial benefit versus dissemination. So two very useful topics for discussion at the workshop could well be these issues of both:

- who should be commissioning secondary studies in software engineering and on what topics?

- how researchers can be encouraged to undertake relevant *primary* studies—including experiments, case studies and well-reported observations (and how journals can be persuaded to recognise the importance of the role of replication studies)?

6 Conclusion

The thesis for this workshop paper is that EBSE is fit for purpose as the basis for decision-making in software engineering, whether for practice or policy purposes. However, it is currently limited by the lack of good primary studies, both in the laboratory and in the field. So for EBSE to be able to inform opinion effectively, and for its use to move into the mainstream, we need to find ways both of obtaining good quality inputs from researchers and practitioners and then of delivering relevant findings in a form that will be useful to practitioners and policy-makers.

Acknowledgements

We thank the U.K. Engineering and Physical Sciences Research Council for providing support for this work via project EP/E046983/1, Evidence-based Practices Informing Computing (EPIC).

References

[1] A. Abran, J. W. Moore, P. Bourque, and R. Dupuis, editors. *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, 2004.

[2] J. Bailey, D. Budgen, M. Turner, B. Kitchenham, P. Brereton, and S. Linkman. Evidence relating to Object-Oriented software design: A survey. In *Proceedings of Empirical Software Engineering & Measurement, 2007*, pages 482–484. IEEE Computer Society Press, Sept. 2007.

[3] O. Brereton, B. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the Systematic Literature Review process within the Software Engineering domain. *Journal of Systems & Software*, 80(4):571–583, 2007.

[4] D. Budgen, B. Kitchenham, P. Brereton, M. Turner, S. Charters, and S. Linkman. Employing the evidence-based paradigm for technology-related decision making. *Evidence & Policy*, 4(2):149–169, 2008. ISSN: 1744-2648.

[5] D. Budgen, B. A. Kitchenham, S. Charters, M. Turner, P. Brereton, and S. Linkman. Presenting software engineering results using structured abstracts: A randomised experiment. *Empirical Software Engineering*, 13(4):435–468, 2008.

[6] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham. Using Mapping Studies in Software Engineering. In *Proceedings of PPIG 2008*, pages 195–204. Lancaster University, 2008.

[7] D. Budgen, M. Turner, S. Charters, J. Bailey, B. Kitchenham, and P. Brereton. Lessons from a Cross-domain Investigation of Empirical Practices. In *Proceedings of EASE 2008*, pages 1–12. BCS-eWiC, 2008.

[8] D. Budgen and C. Zhang. Preliminary reporting guidelines for experience papers. Accepted for EASE 2009, 2009.

[9] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information & Software Technology*, 50:833–859, 2008.

[10] T. Dybå, B. Kitchenham, and M. Jørgensen. Evidence-based software engineering for practitioners. *IEEE Software*, 22(1):58–65, 2005.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[12] J. Hannay, T. Dybå, E. Arisholm, and D. Sjøberg. The effectiveness of pair programming. a meta analysis. In press., 2009.

[13] M. Höst, B. Regnell, and C. Wohlin. Using students as subjects-a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201–214, 2000.

[14] M. Jørgensen. A review of studies on expert estimation of software development effort. *Journal of Systems & Software*, 70(1–2):37–60, 2004.

[15] M. Jørgensen. Estimation of software development work effort: Evidence on expert judgement and formal models. *Int. Journal of Forecasting*, 2007. In press.

[16] M. Jørgensen and K. Moløkken-Østvold. How large are software cost overruns? a review of the 1994 CHAOS report. *Information & Software Technology*, 48:297–301, 2006.

[17] M. Jørgensen and M. Shepperd. A Systematic Review of Software Development Cost Estimation Studies. *IEEE Tran. on Software Engineering*, 33(1):33–53, 2007.

[18] B. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering — a systematic literature review. *Information & Software Technology*, 51(1):7–15, 2009.

[19] B. Kitchenham, P. Brereton, M. Turner, M. Niazi, S. Linkman, R. Pretorius, and D. Budgen. The impact of limited search procedures for systematic literature reviews – an observer-participant case study. Accepted for ESEM 2009, 2009.

[20] B. Kitchenham, D. Budgen, P. Brereton, M. Turner, S. Charters, and S. Linkman. Large-Scale Software Engineering Questions–Expert Opinion or Empirical Evidence? *IET Software*, 1(5):161–171, 2007.

[21] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

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

[23] 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.

[24] S. Macdonell, M. Shepperd, B. Kitchenham, and E. Mendes. How reliable are systematic reviews in empirical software engineering? Accepted for publication in IEEE Trans. SE, 2009.

[25] K. Moløkken-Østvold, M. Jørgensen, S. Tanilkan, H. Gallis, A. Lien, and S. Hove. A Survey on Software Estimation in the Norwegian Industry. In *Proceedings 10th International Software Metrics Symposium (Metrics 2004)*, pages 208–219. IEEE Computer Society Press, 2004.

[26] M. Petticrew and H. Roberts. *Systematic Reviews in the Social Sciences: A Practical Guide*. Blackwell Publishing, 2006.

[27] R. Pretorius and D. Budgen. A mapping study on empirical evidence related to the models and forms used in the UML. In *Proceedings of 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2008*, pages 342–344. ACM Press, 2008.

[28] M. Turner, B. Kitchenham, P. Brereton, S. Charters, and D. Budgen. Does the Technology Acceptance Model predict Actual Use? a systematic literature review. Accepted for publication in Information & Software Technology, 2009.

[29] P. Wendorff. Assessment of design patterns during software reengineering: Lessons learned from a large commercial project. In *Proceedings of 5th European Conference on Software Maintenance and Reengineering (CSMR'01)*, pages 77–84. IEEE Computer Society Press, 2001.

[30] R. Yin. *Case Study Research: Design & Methods*. Sage Books, 3 edition, 2003.

[31] C. Zhang and D. Budgen. Assessing the claims for software design patterns. Submitted for publication., 2009.