UPGRADE is the European Journal for the Informatics Professional, published bimonthly at <http://www.upgrade-cepis.org/>

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by **Novática** <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática,* <http://www.ati.es/>)

UPGRADE monographs are also published in Spanish (full version printed; summary, abstracts and some articles online) by **Novática**

UPGRADE was created in October 2000 by CEPIS and was first published by **Novática** and **INFORMATIK/INFORMATIQUE**, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>)

UPGRADE is the anchor point for UPENET (UPGRADE European NETwork), the network of CEPIS member societies' publications, that currently includes the following ones:
- **Informatica**, journal from the Slovenian CEPIS society SDI
- **Informatik-Spektrum**, journal published by Springer Verlag on behalf of the CEPIS societies GI, Germany, and SI, Switzerland
- ITNOW, magazine published by Oxford University Press on behalf of the British CEPIS society BCS
- **Mondo Digitale**, digital journal from the Italian CEPIS society AICA
- **Novática**, journal from the Spanish CEPIS society ATI
- **OCG Journal**, journal from the Austrian CEPIS society OCG
- **Pliroforiki**, journal from the Cyprus CEPIS society CCS
- **Tölvumál**, journal from the Icelandic CEPIS society ISIP

ISSN 1684-5285

Monograph of next issue (December 2009)

"Privacy and Identity Management"

(The full schedule of **UP**GRADE is available at our website)

CEPIS UPGRADE

The European Journal for the Informatics Professional
http://www.upgrade-cepis.org

Vol. X, issue No. 5, October 2009

Monograph: Experiences and Advances in Software Quality
(published jointly with Novática*)
Guest Editors: *Darren Dalcher and Luis Fernández-Sanz*

* This monograph will be also published in Spanish (full version printed; summary, abstracts, and some articles online) by **Novática**, journal of the Spanish CEPIS society ATI (*Asociación de Técnicos de Informática*) at <http://www.ati.es/novatica/>.

Evidence-based Software Engineering and Systematic Literature Reviews

Barbara Kitchenham, David Budgen, and O. Pearl Brereton

In 2004-5, Kitchenham, Dybå and Jørgensen wrote three papers discussing the concept of evidence-based software engineering (EBSE). EBSE is concerned with the aggregation of empirical evidence and uses systematic literature reviews (SLRs) as a methodology for performing unbiased aggregation of empirical results. This paper presents the concepts of EBSE and SLRs. In order to access the current impact of these concepts we relate existing systematic reviews to the software engineer's body of knowledge (SWEBOK) structure. Our long term goal is to see the SWEBOK supported by a software engineer's body of evidence.

Keywords: Evidence-based Practice, Evidence-based Software Engineering, Systematic Literature Review.

1 Introduction

In 2004, Kitchenham et al. [1] first suggested that software engineering should consider adopting evidence-based practice as pioneered in the field of medicine. The ideas were further considered in two subsequent papers, one discussing the implications for practitioners, the other discussing how evidence-based software engineering could be taught ([2], [3]).

Kitchenham et al. [1] defined the goal of evidence-based software engineering (EBSE) as:

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.

By analogy with medicine, Dybå et al. [2] identified evidence-based software engineering as a five step process:

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.

Although these papers were sympathetic to the idea of EBSE they pointed out some potential problems, for example:

■ Human-centred software engineering experiments are skill-based. They cannot have the same objectivity as medical trials which can utilize double-blind protocols (i.e. experiments where neither the experimenter nor the subject knows which treatment the subject has received) to avoid subject or experimenter bias.

■ Most software experiments are laboratory experiments which are difficult to generalize, whereas medical

Authors

Barbara Kitchenham is Professor of Quantitative Software Engineering at Keele University in the UK. She has worked in software engineering for over 30 years both in industry and academia. Her main research interest is software measurement and its application to project management, quality control, risk management and evaluation of software technologies. Her most recent research has focused on the application of evidence-based practice to software engineering. She is a Chartered Mathematician and Fellow of the Institute of Mathematics and Its Applications, a Fellow of the Royal Statistical Society and a member of the IEEE Computer Society. <b.a.kitchenham@cs.keele.ac.uk>

David Budgen is Professor of Software Engineering at Durham University. He received BSc and PhD degrees in theoretical physics from the University of Durham in 1969 and 1973. After a period working on naval command and control systems, where he developed an interest in software design issues, he was appointed to a post at the University of Stirling, moving to a chair of software engineering at Keele in 1991, and then to Durham in 2005. He is the author of many papers on software design as well as of the textbook Software Design. His research has addressed both the development and evaluation of software development environments and also the actual processes employed in designing software. He has also investigated design strategies appropriate for component-based development, and more recently, service-based systems. He has been a member of the EPSRC computing college since its foundation in 1997. <david.budgen@durham.ac.uk>

O. Pearl Brereton is Professor of Software Engineering in the School of Computing and Mathematics at Keele University. She was awarded a BSc degree in Applied Mathematics and Computer Science from Sheffield University (1970) and a PhD in Computer Science from Keele University (1977). After a period in industry and working for the UK's Science and Engineering Research Council she moved to a research post at Keele University in 1980. She was awarded a personal chair in 2003. Her research focuses on evidence-based software engineering and component-based /service-oriented systems. She is a member of the EPSRC computing college, IEEE Computer Society, the ACM, and the British Computer Society. <o.p.brereton@cs.keele.ac.uk>

controlled trials are field experiments where real patients receive real (or sometimes placebo) treatments.

■ It is not clear who the "practitioner" is in the context of EBSE. Unlike medical practitioners, software engineers are not usually responsible for selecting and using appropriate methods, this is usually the task of senior managers. In our opinion, we should also regard authors of text books and developers of standards as "practitioners" in the sense that they are individuals that may want to make use of EBSE results.

■ Facilities for searching digital libraries are not as good as those available to medics.

Also, EBSE relies on the existence of good quality empirical studies and the ability to aggregate the results of those studies. With respect to empirical studies in general, for over 30 years, Basili's pioneering work on empirical software engineering has lead to an increasing acceptance of the need for empirical studies (e.g. ([4], [5], [6]).

Furthermore, since the mid 90's there have been many initiatives aimed at improving the quality of empirical software engineering research. In 1995, the Empirical Software Engineering Journal was established. Several books on empirical software engineering were published ([7], [8], [9]). Kitchenham et al. published a paper specifying guidelines for empirical studies in software engineering [10] and Jedlitschka et al. presented a set of guidelines for reporting experiments [11]. The International Symposium on Empirical Software Engineering (ISESE) was established in 2003 and merged with the Metrics Symposium in 2008 to form the Empirical Software Engineering and Measurement Conference. Since 2005, researchers at the Simula Research Laboratory have published a series of systematic literature reviews investigating the nature and quality of experiments in software engineering ([12], [13], [14], [15]). This suggests that one basic requirement for EBSE is achieved to some extent.

With respect to the ability to aggregate good quality research results, evidence-based practice usually adopts the systematic literature review methodology. This methodology was first developed in the field of medicine (see for example [16]), but has also been adopted by social scientists ([17], [18]). In 2004, Kitchenham summarized current medical guidelines and attempted to adapt them for software engineering research [19]. The software engineering guidelines were updated in 2007 [20].

We are currently undertaking a research project "Evidence-based Practice Informing Computing" (EPIC) aimed at assessing the use of evidence-based practice and systematic literature reviews in software engineering <http://www.ebse.org.uk>. The results of the EPIC studies are addressing some of the issues raised by the guidelines for example:

■ What problems do novice researchers face when performing systematic reviews [21]?

■ How can we determine the quality of software engineering empirical studies that include technology-centric as well as human-centric empirical studies ([22], [23])?

■ Are complex search strings as suggested by medical standards more efficient than simple search strings?

■ Are broad searches rather than targeted searches necessary in software engineering [24]?

This paper aims to introduce the concepts of Evidence-based Software Engineering and in particular the systematic review methodology. Section 2 describes systematic literature reviews. Section 3 presents a summary of the distribution of systematic reviews across the SWEBOK categories [25] and presents our vision for the future of software engineering.

2 Systematic Reviews - Background

We start by explaining the basic methodology and rationale for systematic reviews. Then we discuss the different types of literature review that can broadly be called "systematic".

2.1 Basic Methodology

Systematic reviews (sometimes called systematic literature reviews in software engineering to ensure they are not confused with inspection research) are a form of *secondary study* that assess the impact of individual empirical studies (referred to as *primary studies)*. A systematic review involves several discrete activities. Existing guidelines for systematic reviews have slightly different suggestions about the number and order of activities. However, medical guidelines and sociological text books (e.g. [16], [17], [18]) are broadly in agreement about the three major stages in the process: Planning the Review, Conducting the Review, Reporting the Review.

The stages associated with *planning the review* are:

- ■ Identification of the need for a review.
- ■ Commissioning a review.
- ■ Specifying the research question(s).
- ■ Developing a review protocol.
- ■ Evaluating the review protocol.

The stages associated with *conducting the review* are:

- ■ Identification of research.
- ■ Selection of primary studies.
- ■ Study quality assessment.
- ■ Data extraction and monitoring.
- ■ Data synthesis.

The stages associated with *reporting the review* are:

- ■ Specifying dissemination mechanisms.
- ■ Formatting the main report.
- ■ Evaluating the report.

We consider all the above stages to be mandatory except:

■ *Commissioning a review* which depends on whether or not the systematic review is being done on a commercial basis.

■ *Evaluating the review protocol* and *Evaluating the report* which are optional and depend on the quality assurance procedures decided by the systematic review team (and any other stakeholders).

The stages listed above may appear to be sequential, but it is important to recognize that many of the stages involve iteration. In particular, many activities are trialled during the protocol development stage, and refined when the review proper takes place.

2.2 Goals and Rationale

Systematic reviews aim to search for, and aggregate, all relevant information on a specific research topic, where in the context of evidence-based software engineering, "information" implies empirical evidence. (However, there is no reason not to use a systematic approach if you need to aggregate other forms of research.)

The use of the term "all relevant" qualifying "information" implies that all primary studies addressing the research topics should be included in the review. However, many researchers prefer to restrict the aggregation process to "best quality" research. For example, in the context of software engineering research, researchers may identify many "lessons learnt" papers that were conducted without using any well-defined research methodology. Such studies may be excluded from any aggregation of primary study results due to poor quality (see [26] for a discussion of how to improve experience reports).

The critical issue for systematic reviews compared with conventional reviews is that a systematic review uses a *defined methodology* that aims to ensure that the review is both fair and seen to be fair. In particular, systematic reviews aim to be:

- *Open* i.e. all the review procedures are reported initially in a study plan (referred to as a protocol) and in the final report of the study (where deviations from the protocol need to be reported).
- *Unbiased* i.e. (as far as possible) all relevant primary studies are included and the results of the included primary studies are fairly aggregated.
- *Repeatable* i.e. (as far as possible) the review could be replicated by other researchers. Note, however, when searching digital libraries, even if electronic search strings are specified, search results may not be completely repeatable.

The goals support one another. The requirement for an open methodology supports the goal for unbiased search and selection of primary studies and helps to support the requirements for repeatability. The detailed procedures used in a systematic literature review also support these goals. For example, the use of a study protocol not only improves study conduct but also reduces the opportunity for researcher bias, since stating in advance what sources will be searched and how avoids relying on the personal knowledge of individual researchers. Furthermore, a protocol helps to ensure that the study is open and repeatable.

2.3 Systematic v. Non-systematic Reviews

Kitchenham and Charters [20] and Brereton et al. [27] both make it clear that the rigour of the systematic reviews process means that it uses more time and effort than typical "expert opinion" based reviews where the included studies and the aggregation process depends on the expertise of the authors. However, results in other disciplines confirm that reviews may miss relevant papers. For example, Oakley and colleagues undertook two investigations of reviews, one related to health promotion for the elderly [28], the other related to anti-smoking education among young people ([29] reported in [17]). In the first case there were six reviews of differing rigour identifying a total of 137 studies, but only 33 studies were common to at least two reviews. In the second case there were two reviews identifying 27 studies of which only three studies were common to both reviews.

Missing relevant studies can lead to drawing incorrect conclusions even when undertaken by experts. For example, after an informal review, the Nobel Laureate Linus Pauling was convinced that mega-doses of vitamin C would protect against the common cold. However a systematic review contradicted this conclusion and identified that Pauling had not cited 5 of the 15 methodologically sound studies [30]. In addition, authors can be biased in their selection of papers to cite. Shaddish surveyed authors of over 280 articles in psychological journals and found studies were often cited because they supported the author's argument, not because they were good quality studies [31].

In the field of software engineering, there are examples of systematic reviews that have overturned "common knowledge" or suggested we are not as well-informed as we think about our methods and tools:

- Jørgensen [32] investigated the results of studies that compared algorithmic cost estimation models with expert opinion estimates. Although cost estimation research for the past thirty years has been premised upon the inferiority of expert opinion estimates, he found no compelling evidence that algorithmic models are more accurate than expert opinion estimates. In fact, one third of the studies said algorithmic models were best, one third said expert opinion-based estimates are best and one third found no difference.
- For many years the original Standish Chaos report was taken 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. [33] looked at all empirical studies that reported failure rates and over-runs. They found that the Standish report was completely out of line with other contemporary studies, which suggested average overruns in the region of 30%. This figure has not changed much in the last 20 years. The methodological problems with the original Standish report (including explicitly soliciting reports of failing projects) were the subject of another study [34].
- Dybå and Dingsøyr recently reviewed holistic empirical studies of agile methods i.e. the results of applying an integrated set of agile methods [35]. In contrast, Hannay et al. have reviewed empirical studies of pair-programming [36]. Both studies suggest we know a lot less about agile techniques than we might think. 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 these effects were small and detailed meta-analysis casts some doubts on the reliability of the effects.

■ Our own study of the TAM (Technology Acceptance Model) draws upon 73 primary studies [37]. This has demonstrated the lack of objective measures for *technology use* in many studies showing that studies of the TAM often measure *perceive*d *us*e rather than *actual us*e and also that some of the key components of the TAM are not good predictors of actual use.

■ A number of *mappin*g *studies*, a form of secondary study that seeks to scope our empirical knowledge about a topic (see Section 2.4), have been undertaken as part of our own studies. While in other disciplines these form a useful start point for a fuller SLR, when performed for software engineering they have largely revealed the major gaps in our empirical knowledge. Examples have included UML [38], object-oriented design [39] and design patterns [40].

2.4 Types of Systematic Review

A standard systematic review is driven by a very specific research question that can be answered by empirical research, for example "Are algorithmic cost models more accurate than expert judgement-based estimates?" This research question drives the identification of appropriate primary studies, informs the data extraction process applied to each included primary study, and determines the aggregation of the extracted data. This type of secondary study contributes most to evidence-based software engineering. If the research question is of interest to industry, it provides an unbiased summary of current empirical knowledge that can support the identification of good practice guidelines.

There are however two other types of rigorous secondary study found in software engineering journals that use a similar methodology to a standard systematic review but have slightly different goals:

■ *Research method studies*. This type of secondary study reviews a set of primary studies with the aim of assessing the methods by which a discipline undertakes its research, see for example [41]. Such a study uses a systematic process for selecting primary studies (in this case a random sample from all papers published in a specific set of publications in a specified time period). Thus, repeatability rests on a statistical argument. Each primary study is classified and the results are aggregated by enumerating the percentages of primary studies in various categories. Although the basic secondary study process is consistent with a systematic review, such studies do not aim to address specific empirical research questions. They are not usually of direct interest to researchers concerned with software engineering topics nor to industry practitioners.

■ *Mapping studies*. This type of secondary study reviews a specific software engineering topic and classifies the primary research papers in that specific domain. The research questions for such a study concern which sub-topics have been addressed, what empirical methods have been used, and what sub-topics have sufficient empirical studies for a more detailed systematic review. Mapping studies are of great potential importance to software engineering researchers. For example, two particularly influential mapping studies looked at cost estimation studies [42] and software experiments [12]. These studies catalogued the primary studies and their results were used by later more detailed systematic reviews ([13], [14], [15], [43]). The subsequent studies were easier to do because the original studies had already identified the relevant literature.

Although at the extreme, mapping studies and systemic reviews have rather different goals, there is often an overlap. Some systematic reviews include a classification system to organize relevant literature followed by a more detailed description of the research within each category. The important difference is that a conventional systematic review makes an attempt to aggregate the primary studies in terms of the research outcomes and investigates whether those research outcomes are consistent or contradictory. In contrast, a mapping study usually aims only to classify the relevant literature.

Finally, a *meta-analysis* is also a type of systematic review. Basically, a meta-analysis is an add-on to a conventional systematic review. It relies on systematic review procedures to search and select the relevant primary studies and define the data extraction process, but uses statistical methods to aggregate primary study outcomes rather than simple tabulation or narrative descriptions (see for example [36]).

A systematic review assessing empirical evidence related to a specific software engineering issue can save time and effort for researchers and practitioners. If it identifies areas where no further research is necessary, there is the opportunity to give clear-cut advice to practitioners and to provide a reliable empirical basis for our international standards and text books. If it identifies an import research question where empirical research is sparse, it provides a clear justification for more detailed primary research.

A systematic mapping study is usually of most relevance to researchers. A problem with some mapping studies is that they only present summary results without the full list of categorized primary studies (e.g. [38], [39], [40]). However, a mapping study which includes all the references provides an excellent starting point for other researchers.

Finally we must emphasize that systematic reviews are not meant to be solely of interest to researchers. Results of good quality systematic reviews provide an *unbiased* assessment of the empirical evidence supporting proposed software engineering methods. Such studies should provide a welcome dose of realism compared with non-peer reviewed "white papers" prepared by vendors and consultants with vested interests in selling products and services. Industry is often critical of academic research, but a significant benefit of high quality academic research is its impartiality. Systematic reviews provide a framework for providing impartial summaries of empirical results which are the basis for evidence-based practice (see for example [44], [45]). Furthermore, researchers have a responsibility to make their results accessible to industry. They should ensure that the results of systematic reviews are explained (whenever possible) in terms of their implications for practice.

However, evidence-based practice also depends on practitioners appreciating the need for high quality evidence and understanding the role researchers have in the development (via primary studies) and aggregation (via secondary studies) of empirical evidence. This would be helped if more of our text books and international standards were based on evidence rather than simply "expert" opinion.

3 Systematic Reviews of Software Engineering

Within software engineering, the SWEBOK [25] 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. In this section we use the structure of the SWEBOK to summarize existing software engineering systematic reviews.

We recently published the results of a mapping study aimed at identifying software engineering systematic reviews [46]. The goal of the study was to identify how many systematic reviews had been published, what research topics were being addressed, and the limitations of current systematic reviews. For this study we used a manual search of a targeted set of 13 conferences and journals published during the period January 1 2004 to 30 June 2007. The sources were selected because they were known to include either empirical studies or literature surveys, and had been used as sources for other systematic mapping studies (e.g. [12], [41]).

We found 20 studies (two of which were found by consulting known researchers or their web sites). We have recently updated this study using a broader automated search strategy (searching 6 electronic sources) from January 2004 to July 2008 [24]. We found an extra 12 systematic reviews in the period January 2004-June 2007, and 19 systematic reviews in the period July 2007-June 2008, plus 1 extra review published in July 2008. In addition, the Information and Software Technology journal has just published an online Virtual Special Issue of systematic reviews (linked by a banner to the IST home page, <http://www.elsevier.com/wps/find/journaldescription.cws_home/525444/description# description>). This special issue includes eight recent systematic reviews of which only one was included in our search (because it was published prior to June 2008). Thus, in total we have found 59 systematic reviews of which 34 addressed software engineering questions rather than mapping general research trends or software engineering topics. Table 1 shows the current state of evidence-based knowledge when mapped on to the structure of the SWEBOK. The total number of relevant reviews totals 32 not 34 because one review [47] was an intentional replication of another (i.e. it addressed exactly the same research question as another) and another review addressed the motivation of individual software engineers [48] and so was outside the scope of the SWEBOK which includes software project management issues but not general management issues.

Because Table1 is coarse-grained at the level of SWEBOK chapter headings, it can be slightly misleading. In particular, although the largest number of secondary studies is reported under *Software Engineering Management*, 8 of these are essentially under just one of its sub-headings (*Effort, schedule and cost estimation*). More important though, as indicated above, the results from a number of these studies contradict established practices and opinions. Under *Software Engineering Process*, three papers relate to *Software Lifecycle Processes* and three relate to *Process Assessment Models*. Furthermore, our results confirm that even at the level of subsections, the SWEBOK is itself very coarse grained in places, so there can be many different systematic reviews that apply to a single subsection.

The number of systematic reviews is encouraging. However, overall the coverage of the SWEBOK is limited, although this reflects a lack of basic empirical studies as well as a lack of systematic reviews.

There is a very real need to provide a better basis for making IT/Computing related decisions than simply using expert judgement, however well this is codifed ([49], [50]). Questions about the effectiveness of large-scale government IT procurement indicate that there is also a need to provide a firmer basis for such projects wherever possible.

We believe that the use of the evidence-based paradigm in software engineering provides a means to address these needs. Furthermore, a goal of an empirical software engineering BOK would provide an ideal framework for researchers and practitioners. It would provide a basic research agenda for researchers undertaking both primary and secondary studies, as well as a structure for making results available to practitioners (engineers and managers), standards and text book authors, and students.

Acknowledgements

This study was funded by the UK Engineering and Physical Sciences Research Council project EPIC/E046983/1.

References

[1] B.A. Kitchenham, T. Dybå, M. Jørgensen. Evidence-based Software Engineering. Proceedings of the 26th International Conference on Software Engineering, (ICSE ’04), IEEE Computer Society, Washington DC, USA, 2004, pp. 273-281.

[2] T. Dybå, B.A. Kitchenham, M. Jørgensen. Evidence-based Software Engineering for Practitioners, IEEE Software, 22 (1), 2005, pp. 58-65.

[3] M. Jørgensen, T. Dybå, B.A. Kitchenham. Teaching Evidence-Based Software Engineering to University Students, 11th IEEE International Software Metrics Symposium (METRICS’05), 2005, p. 24.

[4] V.R. Basili, M.V. Zelkowitz. The software engineering laboratory: Objectives, Proceedings of the fifteenth

SWEBOK Chapters	Title	Number of Sections	Number of Sections with SLR coverage	Total number 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 and Methods	13	1	1
11	Software Quality	24	2	3
Total		277	19	32

Table 1: Distribution of Existing Software Engineering Systematic Reviews.

annual SIGCPR conference, 1977, pp. 256-269.

[5] V.R. Basili, R.W. Reiter. A Controlled Experiment Quantitatively Comparing Software Development Approaches. IEEE Trans on Software Eng., 1981, pp. 299-320.

[6] S. Vegas, N. Juristo, V. Basili. Maturing Software Engineering Knowledge through Classifications: A Case Study on Unit Testing Techniques. IEEE Trans on Software Engineering, 2009, in press.

[7] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslé. Experimentation in Software Engineering. An Introduction. Kluwer Academic Press, 2000.

[8] N. Juristo, A. Moreno. Basics of Software Engineering Experimentation. Kluwer Academic Publishers, 2001, Boston, MA. ISBN-10: 079237990X.

[9] F. Shull, J. Singer, D.I.K. Sjøberg (Eds). Guide to Advanced Empirical Software Engineering, 2008, Springer.

[10] B. Kitchenham, S.L. Pfleeger, L.M. Pickard, P. Jones, D. Hoaglin. K. El Emam, J. Rosenberg. Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering, 28(8), August 2002, pp. 721-734.

[11] A. Jedlitschka, M. Ciolkowski, D. Pfahl. Reporting Experiments in Software Engineering, in Guide to Advanced Empirical Software Engineering, eds. F. Shull, J. Singer, D.I.K. Sjøberg, Springer, 2008.

[12] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.K. Liborg, A.C. Rekdal. A survey of controlled experiments in software engineering. IEEE Trans. on Software Eng., 31 (9), 2005, pp. 733-753.

[13] T. Dybå, V.B. Kampenes, D.I.K. Sjøberg. A systematic review of statistical power in software engineering experiments, Information and Software Technology, 48(8), 2006, pp. 745-755.

[14] V.B. Kampenes, T. Dybå, J.E. Hannay, Dag I. K. Sjøberg. A Systematic Review of Effect Size. Software Engineering Experiments, Information and Software Technology, 49 (11-12), 2007, pp. 1073-1086.

[15] V.B. Kampenes, T. Dybå, J.E. Hannay, Dag I. K.

Sjøberg. A Systematic Review of Quasi-Experiments in Software Engineering, Information and Software Technology, 51 (1), 2009, pp. 71-82.

[16] K.S. Khan, , R..Kunz, J. Kleijnen, G. Antes. Systematic Reviews to Support Evidence-based Medicine. The Royal Society of Medicine Press Ltd., 2003.

[17] M. Petticrew, H. Roberts. Systematic Reviews in the Social Sciences: A Practical Guide. Blackwell Publishing, 2005. ISBN-10: 1405121106.

[18] A. Fink. Conducting Research Literature Reviews. From the Internet to Paper. Sage Publication, Inc., 2005. ISBN-10: 141290904X.

[19] B.A. Kitchenham. Procedures for Undertaking Systematic Reviews. Joint Technical Report, Computer Science Department, 2004, Keele University (TR/SE-0401) and National ICT Australia Ltd (0400011T.1).

[20] B.A. Kitchenham, S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, 2007.

[21] P. Brereton, B. Kitchenham, Z. Li, D. Budgen, F. Bian. Planning problems facing novice systematic literature reviewers: a participant-observer case study, 2009, unpublished manuscript.

[22] B. Kitchenham, P. Brereton, D. Budgen, Z. Li. An Evaluation of Quality Checklist Proposals – A participant Observer case study. 13th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2009, BCS eWiC.

[23] B. Kitchenham, A.J. Burn, Z. Li. A Quality Checklist for Technology-Centered testing Studies. 13th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2009, BCS eWiC.

[24] B. Kitchenham, P. Brereton, M. Turner, M. Niazi, P. Pretorius, D. Budgen. The impact of search procedures for systematic literature reviews – An observer-participant case study. 2009, ESEM 09.

[25] A. Abran, J.W. Moore, P. Bourque, R. Dupuis (eds). Guide to the Software Engineering Body of Knowledge. IEEE Computer Society, 2004.

[26] D. Budgen, C. Zhang. Preliminary Reporting Guidelines for Experience Papers. 13th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2009, BCS eWiC.

[27] O.P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. J. Sys. Software, 80(4), 2007, pp. 571-583.

[28] A. Oakley, D. Fullerton. A systematic review of smoking prevention programmes for young people. London: EPPI Centre, Institute for Education, 1995.

[29] S. Oliver, G. Peersman, A. Harden, A. Oakley. Discrepancies in findings from effectiveness reviews: The case of health promotion for older people in accident and injury prevention. Health and Education Journal 58, 1999, pp. 66-77.

[30] P. Knipschild. Some examples of systematic reviews. British Medical Journal, 309, 1994, pp. 719–721.

[31] W. Shadish. Author judgements about work they cite: Three studies from psychological journals. Social Studies of Science 1995, 25, pp. 477-498.

[32] M. Jørgensen. A review of studies of expert estimation of software development effort. J. Sys. Software, 70, 2004, pp. 37-60.

[33] K.J. Moløkken-Østvold, M. Jørgensen, S.S. Tanilkan, H. Gallis, A.C. Lien, S.E. Hove. A Survey on Software Estimation in the Norwegian Industry, Proceedings Software Metrics Symposium, 2004.

[34] M. Jørgensen, K.J. Moløkken-Østvold. How large are software cost overruns? A review of the 1994 CHAOS report. IST, 48, 2006, pp. 297-301.

[35] T. Dybå, T. Dingsøyr. Empirical studies of agile software development: A systematic review. Information and Software Technology, 50 (9-10), 2008, pp. 833-859.

[36] J.E. Hannay, T. Dybå, E. Arisholm, D.I.K. Sjøberg. The effectiveness of pair-programming: A meta-analysis, Information and Software Technology, 2009, in press.

[37] M. Turner, B. Kitcheham, P. Brereton, S. Charters, D. Budgen. Does the Technology Acceptance Model predict Actual Use? A Systematic Literature Review. 2009. Accepted for publication in Information and Software Technology.

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

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

[40] C. Zhang, D. Budgen. Assessing the claims for software design patterns. 2009, submitted for publication.

[41] R.L. Glass, I. Vessey, V. Ramesh. Research in software engineering: an analysis of the literature. Information and Software Technology, 44(8), 2002, pp. 491-506.

[42] M. Jørgensen, M. Shepperd. A Systematic Review of Software Development Cost Estimation Studies. IEEE Trans. on Software Eng., 33(1), 2006, pp. 33-53.

[43] S. Grimstad, M. Jorgensen, K. Molokken-Ostvold. Software effort estimation terminology: The tower of Babel. Information and Software Technology, 48 (4), 2006, pp. 302-310.

[44] M. Jørgensen. Practical Guidelines for Expert-Judgment-Based Software effort estimation. IEEE Software, 22(3), 2005, pp. 2-8.

[45] M. Jørgensen. Evidence-based Guidelines for Assessment of Software Development Cost Uncertainty. IEEE Trans. on Software Eng., 31(11), 2004, pp. 942-954.

[46] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner,

J. Bailey, S. Linkman. Systematic Literature reviews in software engineering – a systematic literature review. Information and Software Technology, 51, 2009, pp. 7-15.

[47] S. MacDonell, M. Shepperd. Comparing local and global effort estimation models reflections on a systematic reviews. Proceedings of Empirical Software Engineering and Measurement, IEEE Computer Society Press, 2007.

[48] S. Beecham, N. Baddoo, T. Hall, H. Robinson, H. Sharp. Motivation in Software Engineering: A systematic literature review. Information and Software Technology, 50, 2008, pp. 860-878.

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

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