What Scope Is There for Adopting Evidence-Informed Teaching in SE?

David Budgen & Sarah Drummond
School of Engineering & Computing Sciences
Durham University
Durham DH1 3LE, U.K.
{*david.budgen;sarah.drummond*}*@durham.ac.uk*

Pearl Brereton & Nikki Holland
School of Computing & Maths
Keele University
Staffordshire ST5 5BG, U.K.
{*o.p.brereton;n.k.holland*}*@cs.keele.ac.uk*

***Abstract*—Context: In teaching about software engineering we currently make little use of any empirical knowledge. Aim: To examine the outcomes available from the use of Evidence-Based Software Engineering (EBSE) practices, so as to identify where these can provide support for, and inform, teaching activities. Method: We have examined all known secondary studies published up to the end of 2009, together with those published in major journals to mid-2011, and identified where these provide practical results that are relevant to student needs. Results: Starting with 145 candidate systematic literature reviews (SLRs), we were able to identify and classify potentially useful teaching material from 43 of them. Conclusions: EBSE can potentially lend authority to our teaching, although the coverage of key topics is uneven. Additionally, mapping studies can provide support for research-led teaching.**

***Keywords*-empirical; education; evidence-based**

I. Introduction

During a history that now spans more than 40 years, the teaching of software engineering has been largely structured around descriptions and models of procedures (methods) and technical products. To some extent, that emphasis reflects our research culture. As Glass *et al.* have observed from their study of the research literature, software engineering research is dominated by the use of *descriptive* and *formulative* approaches, with a much lower proportion of *evaluative* studies [1]. Methodologically, our research is also overwhelmingly based upon the use of concept analysis and concept implementation as research methods. Where 'real' data is employed (for example, in the formulation of COCOMO), the details of this are often abstracted out when presented to the user (student). Our textbooks likewise present largely descriptive formulations for (mostly idealised) models of the topics that characterise software engineering.

Since it is unrealistic to expect that any teacher should have extensive experience of all aspects of software engineering, one question this leaves is how both teachers and students can be informed about what actually works in practice, and under what conditions? Indeed, even where a teacher may have experience related to particular topics (e.g. testing), they may not have any means of knowing how representative their experience and knowledge actually is.

The models we use in teaching are largely derived from expert knowledge, and while we have argued elsewhere that software engineering practices are excessively dependent upon expert judgement [2], we do also recognise that there are good reasons for using this in teaching. In particular:

- the very wide range of software applications often makes it difficult to identify 'representative' examples of the use of a method or technique;
- the (possibly excessive) concern of companies about proprietary information makes it difficult for the teacher to obtain unbiased exemplars from real life;
- empirical research in software engineering has only recently begun to make a significant impact, and even now, empirical data about major topics may be hard to find or non-existent—and in addition, what is available may be difficult to interpret in a teaching context.

However, while these still remain true, in the case of the third point, there is one development that is likely to influence the way that we teach our subject. The emergence of the *evidence-based paradigm* has significantly changed the way that clinical medicine is taught and practised. Its adoption (and adaptation) for use in software engineering through Evidence-Based Software Engineering (EBSE) [3], [4] has potential to make an impact upon our own teaching too.

The core objective of EBSE (and of evidence-based studies in general) is to find all relevant data related to a topic in a systematic, unbiased and objective manner, to aggregate this data, and then to combine it with experience and context to provide the best possible evidence about the given topic. We describe some further details about EBSE and its practices in the next section, for the moment, this description is sufficient for us to identify our research question as being:

> "*What is available to enable evidence-informed teaching for software engineering?*"

Note that we have preferred to use the term 'evidence-informed' rather than 'evidence-based' here, as we recognise that the range of factors involved in empirical software engineering make it difficult, if not impossible, to provide strong and authoritative evidence on most topics, and also that the use of evidence is likely to be context-specific. Underpinning this question there is also, of course, the assumption that

providing them with evidence will create a better and more satisfying experience for our students, and also prepare them more effectively for their future careers, whatever form these might take. For the moment this has to remain an assumption, since we lack suitable longitudinal studies that would allow a more objective assessment. However, given its impact elsewhere, not least in the teaching of evidence-based medicine (EBM) [5], it is not unreasonable to expect that the impact of EBSE is likely to be beneficial, even if at present we cannot effectively estimate the extent of this.

To address our research question we examine the nature of EBSE and the extent of its adoption by the empirical research community over the past eight years. We describe the research method underpinning our paper; present the outcomes; and then interpret these outcomes in the light of our own experiences as teachers. Finally, we return to consider our original research question.

II. Evidence-Based Software Engineering

In interpreting the evidence-based paradigm for the software engineering domain, Kitchenham *et al.* defined the goal of evidence-based software engineering (EBSE) as being:

> *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.* [3]

This section examines the procedures for doing this, identifies some limitations upon their effectiveness, and describes current progress with realising EBSE.

A. The EBSE Process

Drawing upon the experiences from other domains that have adopted the evidence-based paradigm for research, including clinical medicine, education, and various healthcare specialisations, the same group of researchers identified the following five-step process for EBSE [6].

1) Convert a relevant problem or information need into an answerable question.
2) Search the literature for the best available evidence to answer the question.
3) Critically appraise the evidence for its validity, impact and applicability.
4) Integrate the appraised evidence with practical experience and the values and circumstances of the customer to make decisions about practice.
5) Evaluate software development performance and seek ways to improve it.

The first three steps in this sequence of activities are encapsulated in the procedures used for a *Systematic Literature Review* (SLR), which has been widely adopted as the main tool for evidence-based research across many disciplines, including education and social sciences [7]. Recommendations for conducting an SLR in the context of software engineering have subsequently been incorporated into the *Guidelines* document, updated in 2007 [4][1].

What particularly distinguishes a systematic review from a conventional (expert) review is the use of a *defined methodology* that ensures that the review is both fair and objective, and that it can also be seen to be so. In particular, the conduct of a systematic review should be:

- *Open* in that all the procedures are defined beforehand in the *research protocol* and reported with the findings.
- *Unbiased* in that as far as possible, all relevant studies are included and fairly aggregated by the most appropriate forms.
- *Repeatable* by other researchers (although we should note that the changes that occur in digital libraries from time to time might mean that searches are not completely repeatable).

Of course, one consequence of taking such a thoroughly systematic approach is that conducting an SLR may be time-consuming. However, evidence-based researchers generally consider this to be a worth-while price to pay in order to ensure much greater rigour for their research procedures.

B. Some Limitations of EBSE

While conceptually at least, EBSE has a sound scientific basis and the potential to deliver rigorous, unbiased appraisal of software engineering processes and products, we need to recognise that there are several factors that limit this potential. These include:

- The role of the participant in empirical studies (and especially experiments). Since this role usually involves performing skill-based tasks, it is impossible to apply techniques such as double-blinding (of participants and experimenters), considered essential for medical trials.
- The extensive use of laboratory experiments and quasi-experiments in SE, rather than field trials. Many also use students as participants—so taken together, determining how far the resulting outcomes can be generalised can be a difficult issue, as can their aggregation.
- For medicine, the outcomes (evidence) from an SLR is information that is clearly intended for use by clinicians, who will use this to aid diagnosis and treatment. For education, it is usually the policy-makers at national or possibly regional levels who are the end users. The end users for EBSE may be a much more varied group, since in a software engineering context, decisions about practice may be made at many different levels.
- More pragmatically, 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, making it necessary to translate the chosen search

[1]This is available from www.ebse.org.uk.

strings for use with each engine, inevitably adding to the researcher's work.

There are also related issues about the quality and quantity of many of the 'primary' empirical studies used as inputs to secondary studies such as SLRs. Our own early experiences with some of these issues are described in [8].

C. Recent Progress with EBSE

Researchers such as Basili have performed pioneering work in generating recognition of the need for empirical software engineering [9], [10]. There are now established specialised conferences (such as *Empirical Software Engineering & Metrics* (ESEM) and *Evaluation & Assessment in Software Engineering* (EASE)) as well as a dedicated journal (*Empirical Software Engineering*). However, methodological issues such as research practice and reporting standards remain active topics for research [11], [12], [13], [14].

In terms of EBSE itself, there has been considerable activity (mainly in Europe). This is demonstrated by three 'tertiary studies' of published SLRs [15], [16], [17], and in the rest of this paper, we will refer to these as TS1, TS2 and TS3 respectively. Two other relevant developments are:

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

Around 150 SLRs have been published in the period between 2004 and mid-2011, representing a very rapid expansion of interest in EBSE. There is now a need to draw the outcomes from these together in forms that are appropriate to the needs of different communities: researchers; developers; educators; students; and policy-makers.

D. Background Knowledge Needed by Students

If students are to be presented with evidence that has been derived from empirical studies and synthesised through the use of an SLR, then in order for them to be able to make some assessment of its value and significance, they require at least a basic understanding of the relevant processes. Empirical studies are complex, hence a key question is how much of that complexity needs to be understood by a student, and in what detail. Drawing upon our own experience, and upon model curricula such as the IEEE/ACM SE2004 curriculum guidelines[3], we list our 'expert' assessment of the material that is needed in Table I, and suggest the minimum time that needs to be allocated to each topic—noting that little of this is currently addressed within SE2004. The emphasis in this is very much upon *concepts*, as we see this material as being foundational, with more detail being taught in advanced courses if appropriate.

Table I
SUGGESTED ELEMENTS FOR STUDENT SKILL-SET

Skill Element	Description	Hours
Measurement Theory	Basic measurement scales and their use in SE. Internal and external measures. Combining different measurements. Precision and Error.	1
Statistics	Mainly conceptual issues about distributions and measures such as mean, median and mode (and their use in SE). Many SLRs present box plots and teaching about these provides a useful illustration of many statistical ideas.	2
The Nature of Evidence	Forms of evidence, weighting of evidence, probability and expectation	1
Experimental Practice	Outline knowledge about different empirical forms, confounding factors and threats to validity. Effects of these upon outcomes of experimental studies	2

III. METHOD

In order to answer our research question we have performed a structured analysis of the majority of the SLRs that have been published up to mid-2011, to identify how far these provide material that could be used to support teaching, and what aspects of the curriculum this addresses. In the rest of this section we describe how this was organised.

Not all published SLRs are directly relevant to education. Some are concerned with research trends, and others are in the form of *mapping studies* (also termed 'scoping reviews'), concerned largely with identifying the profile of primary studies addressing a given topic, with the aim of being able to recognise the 'gaps' where more studies are needed and the 'clusters' where a fuller SLR could be performed. The aim of our analysis process was to eliminate these from consideration, and to concentrate on SLRs that aggregate and report results for software engineering topics.

The following subsections are taken from the research protocol that we produced in order to analyse the data available to us from the published SLRs.

1) Search Strategy: For the period up to end 2009, we used the SLRs identified in the three tertiary studies (TS1–TS3). Since all three used rigorous and exhaustive searches, this should represent almost all studies published over that period. For the period from start of 2010 to mid-2011, we used a list generated by reading the index pages of five major software engineering journals (Empirical Software Engineering, IEEE Transactions on Software Engineering, Software Practice & Experience, Information & Software Technology, Journal of Systems & Software). While less exhaustive, the availability of the special journal section

[2]www.ebse.org.uk

[3]sites.computer.org/ccse

mentioned above should ensure that this covers a substantial proportion of the studies published in this period. We also included one paper missed by the tertiary studies.

2) Inclusion/Exclusion Criteria: We excluded:

- SLRs that addressed research trends
- Mapping studies with no analysis of collected data
- SLRs on topics that were not deemed to be relevant to teaching (based on the content of four major textbooks)

Our inclusion criterion was that the SLR covered a topic addressed in the IEEE/ACM curriculum guidelines (SE2004).

3) Data Extraction: We sought information about:

- Any recommendations for practice that are relevant to how we teach about it, produced by the original authors.
- Any similar recommendations that we felt were justified by the outcomes (not all authors of SLRs provide explicit recommendations).
- Where the topic of the SLR was positioned against the categories used for the SEEK (Software Engineering Education Knowledge) from SE2004.

4) Organisation: The tasks of inclusion/exclusion and data extraction were undertaken by all four authors, working as pairs, using different pairings to reduce possible bias.

IV. Results

We first describe the process of conducting the analysis, together with any divergences from our plan, and then present the outcomes.

A. Process of Inclusion/Exclusion and Data Extraction

Our search process identified 145 candidate SLRs, covering the period from 2004 to mid-2011. In two cases, two of the papers were identified as using the same data set, reducing the effective number to 143.

For the inclusion/exclusion process, the papers were assigned randomly to pairs of reviewers drawn from the four authors. Each paper was reviewed to determine whether or not it met our inclusion criteria (i.e. contained usable material or guidelines and addressed a relevant topic). Where two reviewers had different opinions they discussed these in order to come to a shared position (33 were discussed).

For data extraction from the 48 papers that remained, we used a 'extractor-checker' model. Each paper was assigned to one person (usually someone who had reviewed it for inclusion/exclusion) who extracted the core information about the study, and then this was checked by a second person. Again, any differences of interpretation were resolved between them. We prototyped our data extraction template on one of the papers, and then used it for all papers. Our template required us to extract:

1) Paper number
2) Name of first author
3) Brief description of the review topic
4) Suggested assignment to a SEEK *Knowledge Area* (KA) and *Knowledge Unit* (KU)
5) Summary of what the authors were seeking to obtain from the outcomes
6) Summary of what might be useful for teaching
7) Identification of any explicit guidelines provided

The process of reading the papers in more detail led to five (5) additional exclusions. Table II summarises the outcomes from these two stages.

Table II
Papers included from each source

Source	No of papers included ...		
	Initial	After inclusion/exclusion	After data extraction
TS1	20	8	8
TS2	32	10	9
TS3	66	19	17
Journals	24	10	8
Other	1	1	1
Totals	143	48	43

B. The Outcomes

Tables III to X provide more detail about the contents of the papers themselves as well as their relationship to the major SEEK headings (knowledge areas and knowledge units). Space constraints preclude our being able to provide this in much detail, and we have used the paper identifier from the first three tertiary studies to limit the total number of direct references. As a result, only the papers from the journals in the period 2010-11 are referenced directly.

V. Discussion

We first consider the main threats to validity arising from our approach, and then examine how far we can currently identify useful and authoritative support for teaching.

A. Threats to Validity

Since much of our study is interpretive in nature, this issue needs to be considered with care.

- *Internal validity:* is mainly concerned with how well we performed the study, and to what extent any bias might arise from this process. We suggest that there are two key factors here. The first was the way that we *selected* studies for inclusion. We are all experienced teachers of SE, and we followed a well-defined process that ensured that each paper was assessed by at least two of us. On that basis we would argue that we are unlikely to have omitted any material of significance. The second factor is that of categorising the papers against the SEEK, and determining what guidelines they provide for a given SEEK topic. While categorisation was relatively straightforward for most papers, the extraction of guidelines was certainly not so, and even though we followed a well-defined process, there could still be some element of error in our interpretation of the results for any given SLR. (It was often difficult to decide upon the most suitable KU from the SEEK.)

Table III
CATEGORISATION OF AVAILABLE EVIDENCE FOR PROFESSIONAL PRACTICE (PRF)

SEEK KA.KU	Title	Paper	Evidence/Guidelines provided
PRF.psy	Group dynamics/psychology	TS2-25	**Topic:** Challenges faced by teams in collaborative creation of graphical models for systems. **Results:** Major challenges/trade-offs are: involving stakeholders can improve correctness, buy-in and completeness, but lead to conflict; including a skilled modeller can improve quality but reduce stakeholder acceptance; providing an initial model can speed up the process but cause detachment; producing sub-models in parallel can lead to problems with integration.
PRF.com	Communications skills		
PRF.pr	Professionalism	[18]	**Topic:** Reasons why IT professionals change their jobs. **Results:** Provides recommendations on how to retain staff and also which groups of staff are more likely to remain in a job for a long time.

- *External validity:* this can be interpreted as being concerned with how widely the results are likely to be useful for SE teaching in general. Inevitably our interpretation has largely been made in the context of the UK's education structures. So, although the SEEK is intended to be relatively independent of culture or context, we cannot easily demonstrate that our conclusions are equally valid in all educational frameworks.

B. The Evidence and Guidelines

Tables III to X demonstrate that material to support teaching is available for all of the major topics (Knowledge Areas), although it is not particularly evenly spread and its main role is likely to be to augment and interpret more 'classical' textbook material. Many of the outcomes are inconclusive—probably reflecting the breadth and variety of the subject matter as much as any empirical limitations. Indeed, the absence of clear-cut results is itself an important pedagogical element—countering the situation where the use of relatively simplified models can give the impression to the student of greater certainty than is actually the case.

From a teaching perspective, the relatively large proportion of studies listed under *Software Quality* and *Software Management* can be considered valuable, since these are topics where it is particularly valuable to be able to draw upon wider experiences.

Perhaps the most disappointing aspect is that few of the SLRs really provide clear guidelines or interpretation of what they found. Authors of SLRs are apt to be critical of the reporting found in the primary studies they seek to aggregate, but we might suggest that a little of the medicine often suggested for others (better reporting standards) might also be taken by systematic reviewers!

C. The Gaps

For some Knowledge Units the lack of SLRs is hardly surprising (particularly those labelled as 'fundamentals' or 'foundations'). Allowing for this, the most significant gap is in the table for *Software Design*. Even allowing that design is a challenging topic for empirical studies (although not necessarily for case studies), its central importance for any engineering discipline should make this an area of concern for both teachers and for the empirical community.

Perhaps inevitably, the available SLRs and primary studies also tend to focus on aspects of a Knowledge Area that researchers consider to be topical—such as agile methods, global software development, software product lines etc. However, here the needs of teachers and researchers tend to diverge—since teachers need evidence that will consolidate our knowledge about 'core' topics (object-orientation, testing, design techniques,...). Perhaps not surprisingly, these are less likely to have been scrutinised by empirical studies.

D. The SEEK as Our Framework

At a time when the SE2004 curriculum guidelines, for which the SEEK plays a central part, have been under review, it seems useful to identify where we found it difficult to categorise studies during data extraction (regardless of whether or not these were finally included). Some examples of topics which do not seem to have a clear 'home' in the current version of the SEEK include:

- Reuse other than of code, especially in design (DES)
- Open Source Software (OSS), both in terms of effect upon design (DES) and management (MGT)
- Software Product Lines (DES)
- Global software development (MGT)
- Personnel issues such as motivation (MGT)

E. The Role of Mapping Studies

While we excluded mapping studies from this analysis, which is primarily concerned with teaching about core software engineering concepts, we should observe that from our own experiences, these do potentially have some roles to play in teaching, and particularly for more advanced research-led forms of teaching about software engineering. Examples of some of these roles include:

- Providing the teacher with an overview of current research on a given topic.
- Forming the basis for comparative (and other) studies of topics by students, who can use the mapping study to identify papers that address specific issues.

Table IV
CATEGORISATION OF AVAILABLE EVIDENCE FOR SOFTWARE MODELING & ANALYSIS (MAA)

SEEK KA.KU	Title	Paper	Evidence/Guidelines provided
MAA.md	Modeling foundations		
MAA.tm	Types of models	TS2-17	**Topic:** Should cross-company or within-company data be used to build and apply management models. **Results:** Local (within-company) models are considered preferable, but results were tentative.
		TS2-22	**Topic:** Industry experience with using model-driven engineering (MDE). **Results:** Although widely used, "MDE is far from mature" with regard to key elements such as support tools. There is some evidence for productivity gains, but mainly from small-scale studies.
		[19]	**Topic:** Whether the TAM (Technology Acceptance Model) is a good predictor of *actual* use of a technology. **Results:** Behavioural intention to use (BI) is the best predictor, and all TAM variables are worse predictors of objective usage than subjective usage.
MAA.af	Analysis fundamentals	[20]	**Topic:** How tools can be used to manage the inter-related artefacts that need to be managed for domain analysis. **Results:** Discusses domain analysis issues and identifies the scope of existing tools, which tend to address specific processes rather than complete needs.
MAA.rfd	Requirements fundamentals		
MAA.er	Eliciting requirements	TS3-SE49	**Topic:** Challenges that face RE because of increasing use of global software development (GSD) and where the risks differ from co-located RE and development. **Results:** Identifies how GSD changes the elicitation process and provide examples of the risk categories.
		[21]	**Topic:** Investigates elicitation methods and their effectiveness. **Results:** Provides 5 guidelines: interviews are as good as or more effective than introspective techniques and sorting techniques; interviews produce more complete information than introspective techniques, sorting or laddering; unstructured interviews are less efficient than sorting techniques and laddering but as efficient as introspective techniques; introspective techniques are the worst of all techniques tested; and laddering is preferable to sorting.
MAA.rsd	Requirements specification & documentation	TS3-SE55	**Topic:** The use of software engineering models as a starting point for creating textual requirements specifications. **Results:** Use of literature modelling enables better links to be seen between SE models and requirements specifications. Their use helps to ensure that all factors have been considered at the requirements stage.
MAA.rv	Requirements validation		

- Forming the start point for an extended systematic literature review, whereby a student can extend the SLR using the same search strings, in order to identify how a topic has developed.

One illustration is the study of the UML described in [27]. As a mapping study, it offers a range of opportunities for more advanced teaching and study on a topic that will be reasonably familiar to students. Another good example is the study of agile methods reported in [28].

VI. CONCLUSIONS

Our analysis of the available SLRs indicates a strong emphasis upon studying research trends and patterns (mapping studies), which is perhaps to be expected at a time when the use of the evidence-based paradigm for SE has been developing. However, as we have observed, many can usefully contribute to more advanced research-led teaching.

When it comes to the models and frameworks that underpin much of core software engineering teaching, and especially software design, coverage is at best rather patchy. In part this reflects a lack of primary studies addressing these issues (perhaps because no-one has thought it necessary), but there is also a dearth of secondary studies in a number of core areas. That said however, there is much material that can be used, especially in areas such as QUA and MGT where 'front-line' data is likely to be particularly useful to the teacher. If some of this is inconclusive, this is perhaps more realistic than the 'certainties' that can easily be implied when we present students with 'textbook' models that we know are abstractions and simplifications.

Regarding what is available—from the results provided in our tables we can clearly see that while the coverage of the major SEEK headings is uneven, there is a variety of valuable material emerging that also relates to the way that software engineering itself is evolving. By cataloguing this, our paper provides a useful first contribution to the development of evidence-informed teaching in software engineering. This 'map' may also be of use to practitioners too of course, and might usefully be employed by researchers to identify where new studies might contribute by addressing some of the more obvious omissions in its coverage.

Finally, we would also encourage those reporting secondary studies to provide reasoned interpretations of their outcomes for use by teachers and practitioners.

Table V
CATEGORISATION OF AVAILABLE EVIDENCE FOR SOFTWARE DESIGN (DES)

SEEK KA.KU	Title	Paper	Evidence/Guidelines provided
DES.con	Design concepts		
DES.str	Design strategies	[22]	**Topic:** Identifying strengths and weaknesses of aspect-oriented programming, compared to other approaches. **Results:** Identified a range of generally positive effects, particularly for performance, memory consumption, modularity etc. Some studies report negative effects and many included studies were rated as 'poor' for the credibility of any evidence provided.
DES.ar DES.hci DES.dd DES.ste	Architectural design Human-computer interface design Detailed design Design support tools & evaluation		

Table VI
CATEGORISATION OF AVAILABLE EVIDENCE FOR SOFTWARE V & V(VAV)

SEEK KA.KU	Title	Paper	Evidence/Guidelines provided
VAV.fnd	V&V terminology & foundations		
VAV.rev	Reviews	TS1-S15	**Topic:** Use of capture-recapture to estimate the number of faults not found by an inspection. **Results:** The review found that this can extend the rigour of the inspection process by providing a 'control', identified the best estimator model to use, and described how it can be employed.
		[23]	**Topic:** Identify progress with software inspection research. **Results:** Profiles what is known about inspections, based on 153 articles from both technical and management aspects.
VAV.tst	Testing	TS1-S10	**Topic:** Assist with selection of unit testing techniques. **Results:** Creates a framework and recommendations for test-case generation, test set evaluation and test case selection.
		TS3-SE08	**Topic:** Forms of automated acceptance testing, with particular focus upon its use in agile development. **Results:** Discusses eight effects and the benefits of writing these in terms of making developers reflect on design and system behaviour.
		TS3-SE18	**Topic:** Use of meta-heuristic search techniques to generate software tests that address non-functional properties. **Results:** Mostly used for execution-time testing, and violation of timing constraints in particular. Other properties addressed include quality of service, safety, security and usability.
		TS3-SE35	**Topic:** Techniques for performing regression test selection. **Results:** Shows evolution and groupings/classification of techniques. No one technique is superior to others although the information provided may make it possible to identify the most applicable techniques for a given situation. Most techniques are not evaluated sufficiently well to enable a user to make decisions about use.
VAV.hct	Human-computer UI testing & evaluation		
VAV.par	Problem analysis & reporting	TS1-S17	**Topic:** Investigates techniques that can be used for defect detection at different stages of software development. **Results:** For requirements, inspection is clearly better than testing; for design, inspections are more effective and efficient than testing; for code there is no clear answer as to which is better.

ACKNOWLEDGMENT

The authors would like to thank Barbara Kitchenham, both for her major contribution to the concept and realisation of EBSE, and also for many valuable discussions.

REFERENCES

[1] R. Glass, V. Ramesh, and I. Vessey, "An Analysis of Research in Computing Disciplines," *Communications of the ACM*, vol. 47, pp. 89–94, Jun. 2004.

[2] 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*, vol. 1, no. 5, pp. 161–171, 2007.

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

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

Table VII
CATEGORISATION OF AVAILABLE EVIDENCE FOR SOFTWARE EVOLUTION (EVO)

SEEK KA.KU	Title	Paper	Evidence/Guidelines provided
EVO.pro	Evolution processes		
EVO.ac	Evolution activities	TS3-SE39	**Topic:** Determining whether duplication of code affects changeability of systems. **Results** It was not possible to demonstrate or reject the existence of any direct link between duplication of code and changeability. Identifies the extent of the evidence for different relationships as well as the lack of evidence about the effectiveness of mitigation strategies.
		[24]	**Topic:** To identify the architectural characteristics that link changes in software to the resulting effects in a system. **Results:** Creates a software architecture change characterisation scheme (SACCS) mapping high-level changes to lower-level characteristics, together with an assessment of likely impact. SACCS provides a tool for assessing the potential effects of proposed changes to a system.

Table VIII
CATEGORISATION OF AVAILABLE EVIDENCE FOR SOFTWARE PROCESS (PRO)

SEEK KA.KU	Title	Paper	Evidence/Guidelines provided
PRO.con	Process concepts		
PRO.imp	Process implementation	TS2-19	**Topic:** Tailoring of the Rational Unified Process (RUP) to meet the needs of individual development organisations. **Results:** The 5 studies available indicate that the RUP is "too complex to be used without any tailoring" but that doing so requires detailed knowledge of the RUP. They conclude that the RUP is too complex and that more agile approaches are needed.
		TS2-20	**Topic:** Quality, productivity and economic benefits of software reuse. **Results:** Reuse has a positive and significant effect upon software quality and productivity.
		TS3-SE37	**Topic:** Effectiveness of pair programming. **Results:** There is a high level of variance between studies, but two key conclusions are to employ PP either when task complexity is low and time is important, or when task complexity is high and correctness is important.
		TS3-SE40	**Topic:** Use of Scrum in global software development (GSD) projects. **Results:** Scrum practices may be constrained by GSD contextual factors affecting communication, coordination and collaboration.
		TS3-SE43	**Topic:** Challenges facing distributed software development teams and strategies for addressing them. **Results:** Identifies and classifies processes with a strong focus on organisational issues and presents a set of success factors. **Note:** This paper is orthogonal to the SEEK model and spans several KAs.

[5] A. Coomarasamy and K. S. Khan, "What is the evidence that postgraduate teaching in evidence-based medicine changes anything? A systematic review," *British Medical Journal*, vol. 329, pp. 1017–1021, Oct. 2004.

[6] T. Dybå, B. Kitchenham, and M. Jørgensen, "Evidence-based software engineering for practitioners," *IEEE Software*, vol. 22, no. 1, pp. 58–65, 2005.

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

[8] 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*, vol. 80, no. 4, pp. 571–583, 2007.

[9] V. Basili and R. Reiter, "A controlled experiment quantitatively comparing software development approaches," *IEEE Transactions on Software Engineering*, vol. 7, no. 3, pp. 299–320, 1981.

[10] S. Vegas, N. Juristo, and V. Basili, "Maturing software engineering knowledge through classifications: A case study on unit testing techniques," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 551–565, 2009.

[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*, vol. 28, pp. 721–734, 2002.

[12] A. Jedlitschka, M. Ciolkowski, and D. Pfahl, "Reporting experiments in software engineering," in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. Sjøberg, Eds. London: Springer-Verlag, 2008, ch. 8.

[13] 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*, vol. 13, no. 4, pp. 435–468, 2008.

Table IX
CATEGORISATION OF AVAILABLE EVIDENCE FOR SOFTWARE QUALITY (QUA)

SEEK KA.KU	Title	Paper	Evidence/Guidelines provided
QUA.cc QUA.std	Software quality concepts & culture Software quality standards		
QUA.pro	Software quality processes	TS1-S3	**Topic:** To identify the value for an organisation in investing in a CMM program for software process improvement. **Results:** Provides median and range values for improvement across seven common performance metrics and the authors argue that CMM programs can therefore lead to improved software development and maintenance.
		TS2-47	**Topic:** Use of software process improvement (SPI) in small organisations. **Results:** Identifies some success factors and reasons why small organisations have difficulty coping with the requirements of CMM and other standards.
		TS2-49	**Topic:** Organisational motivations for adopting CMM-based software process improvement (SPI). **Results:** Organisations are more strongly motivated by product issues such as software quality and development time/cost than by 'process' issues.
QUA.pca	Process assurance	TS3-SE11	**Topic:** Analysis of the causes of defects in code to aid product-based process improvement. **Results:** Provides support for the 'traditional' defect prevention process; recommends which metrics should be collected; and advises using a taxonomy of defects. Describes a proposed process improvement process based on the results.
QUA.pda	Product assurance	TS3-SE24	**Topic:** Effectiveness of coupling metrics as predictor of maintainability for Aspect-Oriented Programming (AOP). **Results:** Existing static coupling metrics are not adequate as predictors and specific metrics need to be created. Dynamic metrics may be more useful for AOP.
		TS3-SE59	**Topic:** Identify techniques and models for predicting the maintainability of software. **Results:** Provides classification of techniques and list of successful metrics for predicting maintainability as well as a review of definitions of maintainability.

[14] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[15] 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*, vol. 51, no. 1, pp. 7–15, 2009.

[16] B. Kitchenham, R. Pretorius, D. Budgen, P. Brereton, M. Turner, M. Niazi, and S. Linkman, "Systematic literature reviews in software engineering — a tertiary study," *Information & Software Technology*, vol. 52, pp. 792–805, 2010.

[17] F. Q. da Silva, A. L. Santos, S. Soares, A. C. C. França, C. V. Monteiro, and F. F. Maciel, "Six years of systematic literature reviews in software engineering: An updated tertiary study," *Information and Software Technology*, vol. 53, no. 9, pp. 899–913, 2011.

[18] A. H. Ghapanchi and A. Aurum, "Antecedents to IT personnel's intentions to leave: A systematic literature review," *Journal of Systems & Software*, vol. 84, pp. 238–249, 2011.

[19] M. Turner, B. Kitchenham, P. Brereton, S. Charters, and D. Budgen, "Does the Technology Acceptance Model predict Actual Use? A Systematic Literature Review," *Information & Software Technology*, vol. 52, no. 5, pp. 463–479, may 2010.

[20] L. B. Lisboa, V. C. Garcia, D. Lucrédio, E. S. de Almeida, S. R. de Lemos Meira, and R. P. de Mattos Fortes, "A systematic review of domain analysis tools," *Information and Software Technology*, vol. 52, no. 1, pp. 1 – 13, 2010.

[21] O. Dieste and N. Juristo, "Systematic review and aggregation of empirical studies on elicitation techniques," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 283–304, 2011.

[22] M. S. Ali, M. A. Babar, L. Chen, and K.-J. Stol, "A systematic review of comparative evidence of aspect-oriented programming," *Information and Software Technology*, vol. 52, no. 9, pp. 871 – 887, 2010.

[23] S. Kollanus and J. Koskinen, "Survey of software inspection research," *The Open Software Engineering Journal*, vol. 3, pp. 15–34, 2009.

[24] B. J. Williams and J. C. Carver, "Characterizing software architecture changes: A systematic review," *Information & Software Technology*, vol. 52, no. 1, pp. 31–51, 2010.

[25] D. Smite, C. Wohlin, T. Gorschek, and R. Feldt, "Empirical evidence in global software engineering: a systematic review," *Empirical Software Engineering*, vol. 15, pp. 91–118, 2010.

[26] K. Peterson, "Measuring and predicting software productivity: A systematic map and review," *Information & Software Technology*, vol. 53, pp. 317–343, 2011.

[27] D. Budgen, A. Burn, P. Brereton, B. Kitchenham, and R. Pretorius, "Empirical evidence about the UML: A systematic literature review," *Software — Practice and Experience*, vol. 41, no. 4, pp. 363–392, 2011.

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

Table X
CATEGORISATION OF AVAILABLE EVIDENCE FOR SOFTWARE MANAGEMENT (MGT)

SEEK KA.KU	Title	Paper	Evidence/Guidelines provided
MGT.con	Management concepts		
MGT.pp	Project planning	TS1-S7	**Topic:** Effectiveness of cost estimation for software development, using experts. **Results:** Provides outcomes from comparisons between expert estimation and formal estimation models, and provides 12 'best practice' guidelines for expert effort estimation.
		TS1-S8	**Topic:** Comparison of model-based and expert judgement based predictions of software development effort. **Results:** Model-based prediction failed to systematically out-perform expert-based predictions, and some explanations of why this is so are provided.
		TS1-S12	**Topic:** to look at the internal consistency, and between-studies consistency, of studies that compare regression and analogy-based cost estimation. **Results:** Little clear evidence as results are inconsistent—from 20 studies comparing relative accuracy of the two approaches, 45% offer some support for analogy, 35% for regression, and 20% are inconclusive.
		TS1-S14	**Topic:** To identify any factors that contribute to the success of software estimation. **Results:** Large projects are more prone to underestimation; managers mainly rely upon expert estimation; estimations for in-house projects tend to be more accurate; choice of method is often based upon previously successful use.
		TS2-32	**Topic:** Impact of clients upon effort estimation accuracy. **Results:** Examines factors leading to over-runs, and discusses importance of requirements elicitation, project management and cost estimation.
		TS2-33	**Topic:** Factors that lead to uncertainty in estimating development costs. **Results:** Provides a set of guidelines with subjective classification of the strength of supporting evidence.
		TS3-SE52	**Topic:** To compare between the waterfall and iterative/incremental development approaches for effect upon cost, duration and quality. **Results:** Factors such as cost tended to be reported using different measures. Main findings are presented as a table, and overall there is no clear evidence in favour of either approach.
		TS3-SE56	**Topic:** Risks and resolution techniques in global/distributed software projects (GSDPs). **Results:** Synthesis of potential risks to GDSPs and resolution techniques to be aware of.
		[25]	**Topic:** Progress and best practice regarding global software development (GSD). **Results:** Profiles activities world-wide and the activities studies, almost all from inter-organisational development. Identifies a set of best practices for GSD, benefits they provide, and constraints upon their use.
MGT.per	Project personnel & organisation	TS3-SE01	**Topic:** What motivates and de-motivates developers, and what models exist for this. **Results:** Of the 21 motivators identified, those most frequently cited relate to the need to identify with the task. De-motivators include working conditions and lack of resources. Factors may also depend upon a developer's current career stage.
		TS3-SE75	**Topic:** Identifies key motivation factors to create a new model of motivation. **Results:** Provides data about motivation, which is complex, in the form of a number of formal models.
MGT.ctl	Project control	TS3-SE72	**Topic:** Identify what makes software productivity vary across different contexts. **Results:** The review examined four groups of factors: product, personnel, project and process. Concludes that productivity "still depends on the capabilities of the people and tools involved" and provides comments on selected factors and what influences them. Observes that reuse is not the "key to productivity improvements" as is commonly believed.
		[26]	**Topic:** Examines accuracy/usefulness of different approaches to predicting and measuring software development productivity. **Results:** Classifies approaches and presents advantages and limitations of both measurement and prediction approaches.
MGT.cm	Software configuration management		