Evidence-based Software Engineering for Practitioners*

Tore Dybå[1,3], Barbara A. Kitchenham[2,4], and Magne Jørgensen[1]
tore.dyba@sintef.no, barbara@cs.keele.ac.uk, magnej@simula.no

[1]*Simula Research Laboratory, P.O. Box 134, NO-1325 Lysaker, Norway*
[2]*NICTA, Locked Bag 9013 Alexandria, NSW 1435, Australia*
[3]*SINTEF ICT, NO-7465 Trondheim, Norway*
[4]*Dept. of Computer Science, Keele University, Staffordshire ST5 5BG, UK*

Abstract

Objective: Our objective is to explain how practitioners in industry can use evidence-based software engineering (EBSE) to assist decisions concerning the adoption of new techniques. *Rationale*: Practitioners may make incorrect technology adoption decisions if they do not consider scientific evidence about the efficacy of techniques. *Method*: We adapt procedures used for evidence-based medicine to software engineering and discuss how these procedures map to software process improvement. *Results*: We found EBSE fitted well with current ideas concerning process improvement and that it could be an important means for closing the gap between research and practice. However EBSE presents difficulties for practitioners because current software engineering research is limited and not reported in a manner that assists accumulation and evaluation of evidence. *Conclusions*: EBSE has potential benefits for software engineering practitioners, but will be hindered without changes to software engineering research.

Keywords: Evidence, empirical software engineering, evaluation.

Introduction

Software managers and practitioners often need to make decisions about what technologies to employ on their projects. They may be aware of problems with their current development practices (for example, bottlenecks in production, or a large number of defect reports from customers) and want to resolve the problem. Alternatively they may have read about a new technology and want to take advantage of its promised benefits. However it is difficult for practitioners to make an informed decision about whether or not to adopt a new technology because there is little objective evidence to confirm its suitability, limits, qualities, costs, and inherent risks. This can lead to poor decisions about technology adoption, for example Zelkowitz *et al*. (2003) say:

> *"Software practitioners and managers seeking to improve the quality of their software development processes often adopt new technologies without sufficient evidence that they will be effective, while other technologies are ignored despite the evidence that they most probably will be useful."*

For example, OO enthusiasts were initially keen to promote the value of hierarchical models. Only later did experimental evidence identify the problem that deep hierarchies are more error prone than shallow hierarchies.

In contrast, medical practice has changed dramatically during the last decade as a result of adopting an evidence-based paradigm. In the late 80s and early 90s, studies showed on the one hand that failure to undertake systematic reviews of medical research could cost lives, and

* Submitted to *IEEE Software*, 12 May 2004; revised 1 September 2004.

on the other hand that the clinical judgment of experts compared unfavorably with the results of systematic reviews. Since then, many medical researchers have adopted the evidence-based paradigm and medical practitioners are now trained in this approach (Sackett *et al*., 2000). Although not without its critics (e.g. Feinstein and Horowitz, 1997), evidence-based medicine (EBM) is generally regarded as successful and has prompted many other disciplines to adopt a similar approach, including, for example, psychiatry, nursing, social policy, and education.

Software companies are often under pressure to adopt immature technologies because of market and management pressures. In this paper, we suggest that practitioners consider evidence-based software engineering (EBSE) as a mechanism to support and improve their technology adoption decisions.

Aim and Methodology of Evidence-Based Software Engineering

The aim of EBSE is 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 (Kitchenham *et al*., 2004). This means we do not expect a technology to be universally good or universally bad, only more appropriate in some circumstance and to some organizations than to others. Furthermore, practitioners will need to assemble empirical research about a technology of interest and evaluate the research from the viewpoint of their own specific circumstances.

This aim is decidedly ambitious, particularly since the gap between research and practice can be wide. EBSE seeks to close this gap by encouraging a stronger emphasis on methodological rigor while at the same time focusing on relevance for practice. This is important since rigor is a necessary basis in any research that purports to be relevant. Moreover, since most of the research done so far is failing to influence industrial practice (Potts, 1993), there is also a pressing need to avoid SE research remaining an Ivory Tower activity with an emphasis on academic rigor over relevance to practice.

So, while rigor is a necessary condition for relevant SE research, it is not sufficient. Medical evidence is based on rigorous studies of therapies given to real patients requiring medical treatment; laboratory experiments are not considered to provide compelling evidence. This implies that software engineering should not rely solely on laboratory experiments and should attempt to gather evidence from industrial projects, using observations studies, case studies, surveys, and field experiments. These empirical techniques do not have the scientific rigor of formal randomized experiments but they do avoid the limited relevance of small-scale, artificial software engineering experiments.

Furthermore there are substantial problems accumulating evidence in a systematic fashion and not only because accumulating evidence from different types of study is difficult. A specific challenge in practicing EBSE is that we often find that different empirical studies of the same phenomenon report different and sometimes contradictory results (Pickard *et al*., 1998). Unless these differences can be understood, it is difficult to integrate individual pieces of evidence. Among other things, this points to the importance of reporting contextual information in empirical studies to assist the explanation of conflicting research results (Kitchenham *et al*., 2002).

EBSE is a process involving five steps:

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 performance and seeking ways to improve it.

However, EBSE is not a standalone activity. Much of what is needed to practice EBSE already exists in the concept of technology transfer (see Pfleeger and Menezes, 2000) and software process improvement (SPI) (e.g. Basili and Caldiera, 1995). SPI involves several different steps (each different researcher and consultant having his/her own view of exactly how many steps), e.g. (1) identifying a problem; (2) proposing a technology or procedure to address that problem; (3) evaluating the proposed technology in a pilot project; (4) if the technology is appropriate, adopting and implementing the technology; (5) monitoring the organization after implementation of the new technology; and finally (6) returning to step (1).

Thus, EBSE provides mechanisms to support various parts of SPI. In particular, EBSE is focused on finding and appraising an appropriate technology for its suitability in a particular situation. This is an area where SPI is usually rather weak. It is often assumed that finding a candidate technology is relatively easy and evaluating the technology is the hard part. However, we believe that selecting an appropriate technology is much more difficult than was previously assumed and is a critical element of a good process improvement activity. The only step in SPI not supported by EBSE is technology infusion, which is supported by change management procedures and diffusion models.

Step 1: Asking an answerable question

The first step in EBSE is to convert a relevant problem or information need into an answerable question. EBSE does not propose a specific method to identify and prioritize problems. If you are using SPI you should be monitoring your projects and so be in a position to identify process and product problems. Otherwise problem identification relies on the expertise of individual members of staff. Another form of help can be found in the Goal-Question-Metric (GQM) paradigm (Basili and Caldiera, 1995), in which questions are derived from specific goals.

Once the problem is identified, we need to specify an answerable question. Typical questions ask for *specific knowledge* about how to appraise and apply methods, tools, and techniques in practice. Sackett *et al*. (2000) suggests that well formulated questions usually have three components:

- The main intervention or action we are interested in.
- The context or specific situations of interest.
- The main outcomes or effects of interest.

For medical problems, partitioning the question into “intervention”, “context”, and “effect” not only makes it easier to go from general problem descriptions to specific questions, but also makes it easier to think about what kind of information is needed to answer the question.

In the software engineering context, factors to consider when deciding which question to answer first include:

- Which question is most important to our customers?
- Which question is most relevant to our situation?
- Which question is most interesting in the context of our business strategy?
- Which question is most likely to recur in our practice?

- Is the question possible to answer within the time we have available?

Sidebar 1

Example: Is pair programming useful?

According to what is suggested in evidence-based medicine, this question regarding the use of pair programming should be specified in more detail, for example into something like "Does the use of pair programming lead to improved code quality when practiced by professional software developers?" Now, we specified what intervention we are interested in (pair programming), what population we are interested in (professional software developers), and what effect we are looking for (code quality).

Ideally, we should be even more specific regarding the intervention. In this example we presume a comparison with something, without specifying it. Any estimation of an effect size involves either a comparison or an association. Here, we could have made it clear that we wanted to compare pair programming with "individual programming". Alternatively, we could compare it with "partner programming" (i.e. the programmers work on different tasks on different computers, but they share the same physical office or workspace so that they are able to share ideas, thoughts, problems and so forth). With respect to context, we could also have been more specific. We have, for example, not specified the nature of the software development organization, the skills and experience of the developers, nor the software engineering environment being used.

However, searching for the three keywords "pair", "programming", and "professional" in the abstracts of the nearly two million articles indexed in *IEEE Xplore* and ACM *Digital Library*, resulted in only two articles retrieved[1] – and neither were studies of pair programming using professionals as subjects. Therefore, for software engineering problems we may need to be less stringent in question formation because:

- We have a much smaller body of empirical research than medicine and we cannot afford to restrict our searches too much or we will not find any relevant studies.
- To support rational decision making we are usually interested in all possible outcomes; for example, we may not want to adopt a technique that results in very fast time to market, if the side effect is very poor operational reliability. This implies that, in particular, we should not restrict our questions too severely with respect to outcomes.

The main challenge in this step is, in other words, to convert a practical problem into a question that is specific enough to be answered, but not so precise that we do not get any answers (see Sidebar 1). When we have formulated a question, we can move to the next step, searching for the best evidence to answer it.

Step 2: Finding the best evidence

After we have specified our question, we need to identify the best available evidence with which we can answer it. Finding an answer to our question includes selecting an appropriate information resource and executing a search strategy. It is important, however, to separate: (1) the question we want to answer; (2) the question implemented in the search keywords; and (3) the questions answered in the studies found.

There are several information sources we can use. We can, for example, get viewpoints from our customers or the users of the software, we can ask a colleague or an expert, we can use

[1] Search performed 1 September, 2004.

what we learned as a student or in professional courses, we can use our own experience, or we can search for research-based evidence, which is our main focus.

By research-based evidence, we mean reports, articles, and other documents that describe a study conducted and reported according to certain guidelines (e.g. Kitchenham *et al*., 2002). The main source of research-based evidence is articles published in scientific journals. Additional evidence can also be found in books, bibliographical data bases, and on the Internet. However, when we start searching for evidence, a common problem is that relevant evidence is not as easy to find as we might hope for. Several thousand software related publications are published each year and even if you work within a rather specialized subject area, it is almost impossible for a practitioner to keep up to date by reading all the journals. For most practitioners, reading important magazines such as the *Communications of the ACM*, *IEEE Computer*, *IEEE Software*, and *IT Professional* would probably be enough to get a general overview of the latest developments within software engineering.

Keeping up to date is much easier when we can use sources that combine results from independent empirical studies of a particular phenomenon (Pickard *et al*., 1998). Systematic reviews have clearly defined inclusion criteria and standardized indicators of individual and combined effect sizes. Such reviews summarize the available evidence regarding specific phenomena, showing where the research correspond or contradict, and uncovering gaps in our knowledge. However, except for *ACM Computing Surveys*, there is no other software engineering journal dedicated to systematic reviews. This makes it important to search for such reviews in other journals as well. The situation is quite different in medicine where the Cochrane collaboration[2] publishes and updates systematic reviews of all important areas of health care online.

In addition to keeping up-to date by reading important magazines and using such evidence resources that systematize and summarize evidence, we will also have to search for evidence in electronic databases available on the Internet. By doing a literature search here, we get a more specific overview of the published research within our area of interest than what is generally the case for the magazines and the systematic reviews (at least for the time being).

There are many organizations that index published articles in several databases, that is, they include bibliographic information such as author, title, keywords etc. Such indexing makes it easier to search for information regarding a problem area or find an answer to a specific question. Examples of such organizations are IEEE, with its database *IEEE Xplore*, and ACM, with its *Digital Library* (see Table 1).

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

Table 1. Examples of useful information sources.

IEEE *Xplore,* http://ieeexplore.ieee.org

IEEE *Xplore* is an online delivery system that provides full text access to IEEE's transactions, journals, magazines and conference proceedings published since 1988 and all current IEEE standards via the Internet. IEEE *Xplore* serves IEEE members as well as users who are not members of the IEEE but have subscriptions to online publication packages.

ACM Digital Library, http://www.acm.org/dl

ACM Digital Library contains citations and full text from ACM journals and newsletter articles and conference proceedings. Full citations consist of title, author, publication data, and, when available, abstracts, citations (where the paper has been referenced by other papers), references (by the paper to other papers), index terms from ACM's Computing Classification System (CSS) and reviews from ACM's Computing Reviews.

ISI Web of Science, http://isiknowledge.com

ISI Web of Science consists of high-quality databases that contain information gathered from thousands of scholarly journals in several areas of research, e.g. Science Citation Index Expanded, Social Sciences Citation Index and Arts & Humanities Citation Index. With *Web of Science* you can search by bibliographic data and by cited reference, create sophisticated search strategies that you can save for future use, mark records for saving, printing and exporting and link to the full text of articles or to related content from other notable information sources.

EBSCOhost ESJ, http://ejournals.ebsco.com

EBSCOhost Electronic Journals Service (EJS) consolidates over 8,000 e-journals containing millions of articles from all major publishers, covering practically all disciplines. With EJS, you can easily find journals, view tables of contents and abstracts, and link directly to full text in the newest issue of your subscribed journals.

CiteSeer, http://citeseer.nj.nec.com

CiteSeer is a scientific literature digital library that aims to improve the dissemination and feedback of scientific literature, and to provide improvements in functionality, usability, availability, cost, comprehensiveness, efficiency, and timeliness by the use of Autonomous Citation Indexing. CiteSeer indexes Postscript and PDF research articles on the Web and provides a range of useful features.

Step 3: Critically appraising the evidence

Unfortunately, published research isn't always of good quality; the problem under study might be unrelated to practice or the research method could have weaknesses so that the results cannot be trusted. To assess whether research is of good quality and can be applied to practice, we must be able to critically appraise the evidence.

It can often be difficult to evaluate the scientific quality of an article without being a scientist oneself, although for the practitioner it is often more important to evaluate the article's relevance to practice rather than scientific rigor. Most journals make use of external referees to evaluate manuscripts before eventual publication, which makes such manuscripts more trustworthy. This means that research presented in non-refereed journals and conferences or on the Internet, requires additional insight from the reader in order to evaluate the results and their relevance to practice. However, even in a reputable scientific journal it may be difficult even for researches to agree about the rigor of an experiment (see for example the discussion arising from a recent study of formal methods, Berry and Tichy, 2003, and Sobel and Clarkson, 2003)

In evidence-based medicine the most convincing form of evidence is a systematic review of a series of double-blind randomized field trials. Within SE we do not yet have a large number of well-conducted replications of rigorous experiments, so our empirical studies are much less reliable scientifically. We believe software engineering researchers could provide more help to practitioners if they undertook and published more systematic reviews of important software engineering topics. However, until this happens, practitioners need to be prepared to summarize evidence themselves. When we have evidence from different types of study, we need to have some way to assess the quality of each study. We present a checklist of factors you need to consider when evaluating an empirical study in Table 2. In addition, the relative trustworthiness of different types of empirical study is discussed in the Australian National Health and Medical Research Council guidelines (2000).

Table 2. Checklist for appraising published studies.

1. **Is there any vested interest?**
   - Who sponsored the study?
   - Do the researchers have any vested interest in the results?
2. **Is the evidence valid?**
   - Was the design of the study appropriate to answer the question?
   - How were the tasks, subjects, and setting selected?
   - What data was collected and what were the methods for collecting the data?
   - Which methods of data analysis were used and were they appropriate?
3. **Is the evidence important?**
   - What were the results of the study?
   - Are the results credible and, if so, how accurate are they?
   - What conclusions were drawn and are they justified by the results?
   - Are the results of practical significance as well as of statistical significance?
4. **Can the evidence be used in practice?**
   - Are the findings of the study transferable to other industrial settings?
   - Were all important outcome measures evaluated in the study?
   - Does the study provide guidelines for practice based on the results?
   - Are the guidelines well described and easy to use?
   - Will the benefits of using the guidelines outweigh the costs?
5. **Is the evidence in this study consistent with the evidence in other available studies?**
   - Are there good reasons for any apparent inconsistencies?
   - Have the reasons for any disagreements been investigated?

Step 4: Applying the evidence

When the evidence has been critically appraised, we make use of it in our decision making process by integrating it with our own practical experience, our customers' requirements, and our knowledge of the specific circumstances of the concrete situation and apply it in practice. However, this is not a straightforward procedure.

Active use of new knowledge is characterized by applying or adapting specific evidence to a specific situation in practice. This is in contrast to traditional, passive modes of transmitting information through teachers, books, manuals, colleagues, or business partners. While such transmission can help in arranging the conditions required for learning to take place, it cannot substitute for learning through direct experience. Therefore, in order to practice EBSE, the individual software developer must commit him or herself to actively engage in a learning

process, combining the externally transmitted evidence with prior knowledge and experience. What characterizes a software developer using EBSE is that he or she makes individual judgments in a given situation rather than simply conforming to approved standards and procedures.

In practice, the ease of applying evidence depends on the type of technology (methods/tool/technique/practice) we have evaluated. Some technologies apply at the level of the individual developers, for example evidence related to how best to comment programs can be adopted by an individual developer. However, evidence related to the adoption of a CASE tool or a mathematically-based formal specific method cannot be used without support from project and senior managers. Furthermore even techniques that can be adopted and evaluated by the individual developer have little impact unless they lead to a project-wide or organizational-wide process change.

Thus, it is at this point that EBSE needs to be integrated with process improvement. Process improvement relies on a systematic introduction and evaluation of proposed process change and is often supported by change management processes. EBSE should provide the scientific basis for undertaking specific process changes while SPI should manage the process of introducing a new technology.

Step 5: Evaluating performance

In EBM, the final step in the process is for individual medical practitioners to reflect on their personal use of the EBM paradigm (Sackett *et al*., 2000). In SPI, the final step in the process is usually to confirm that the process change has worked as expected. We believe both concerns are of relevance for EBSE.

We need to consider how well we perform each step in the EBSE process and how we might improve our use of EBSE. In particular, we should ask ourselves how well we are integrating evidence with practical experience, customer requirements, and our knowledge of the specific circumstances.

Following SPI practice, we also need to assess whether process change has been effective. However, environmental turbulence and rapid changes in technology often lead to a need for adaptation and learning during projects with a large degree of creativity and improvisation, which suggests that we cannot wait until the end of a project to draw out the lessons learned (Dybå *et al*., 2004).

Short meetings aimed at evaluating performance in the midst of action, so called After Action Reviews (Collison and Parcell, 2001), are a simple way for individuals and teams to learn immediately, from both successes and failures. All that is needed is a suitable task with an identifiable purpose and some metrics on which performance can be measured. A typical meeting lasts for 10-20 minutes and answers four simple questions:

- What was supposed to happen?
- What actually happened?
- Why were there differences?
- What did we learn?

However, it is important not to over-react. One isolated bad result should not cause a new method to be abandoned, unless there are strong grounds to believe that the bad result is

intrinsic to the method itself rather than a chance effect resulting from the particular task and the particular engineering staff. Equally a single good result should not mean that further monitoring is unnecessary. One of the authors undertook a study of COCOMO in the early 80's. The first two projects on which data was collected were an almost perfect fit to the Intermediate COCOMO model, thereafter, however, none of the other projects exhibited effort values anywhere near the COCOMO predictions.

When the project, or a major part of it, is completed, SPI principles suggest we must confirm that the expected improvement has taken place. A simple way to do this is to arrange a post-mortem analysis (PMA) (Collier *et al.*, 1996). A PMA is similar to an after action review, but is conducted in more depth. Instead of 10-20 minutes, a PMA typically takes between a couple hours and up to a full day. It aims to capture lessons and insights (both good and bad) for future projects by evaluating the following questions:

- What went so well that we want to repeat it?
- What was useful but could have gone better?
- What were the mistakes that we want to avoid for the future?
- What were the reasons for the successes or mistakes and what can we do about it?

The main result from a PMA is better evidence regarding the specific process or technology that has been used – evidence that might be reused as guidelines for the future, experience notes, new or improved checklists, improved development models, and a general understanding of what works and what not works in projects in our own organization. In the context of EBSE, PMAs, and when available organization-wide measurement programs, provide the information needed to restart the EBSE cycle, allowing us to identify and prioritize product and process problems by collating experiences from different projects.

Discussion

We have suggested that it is important for software practitioners to base their choice of development methods on available scientific evidence. We do not suggest this is easy. Evidence-base medicine arose because medical practitioners were overwhelmed by the large number of scientific studies; in software engineering our problems are rather different. There are relatively few studies as our pair-programming example showed. Furthermore, when evidence is available, it is still difficult for a software practitioner to judge the quality of the evidence and assess what the evidence means in terms of his/her specific circumstances. This implies that given the current state of empirical software engineering, practitioners will need to adopt more proactive search strategies such as approaching experts, other experienced practitioners and researchers directly.

Since establishing a fruitful cooperation between research and practice is one of the basic ideas behind EBSE, there should be closer links between research and practice so that research is relevant to practitioners' needs and practitioners are willing to participate in research.

Readers may have noticed that we have offered no evidence of the benefits of EBSE. Although we have no examples of EBSE being used by practitioners, we present examples of our own use of EBSE in Sidebar 2. Based on this, and other ongoing industrial and educational initiatives that the authors are engaged in, we believe that evidence-based practice is possible and potentially useful for software practitioners

However, evidence-based practice also places requirements on researchers. Our recommendation is for researchers to adopt as much of the evidence-based approach as is possible. Specifically this includes being more responsive to the needs of practitioners when identifying topics for empirical research. Also, it means improving the standard both of individual empirical studies and of systematic reviews of such studies. Researchers need to perform and report replication studies so that we can accumulate reliable evidence about software engineering topics and report their results in a manner that is accessible to practitioners.

Conclusion

Evidence-based practice works in medicine (see Sackett *et al*., 2000). Furthermore, the authors' own experience from undertaking empirical studies, systematic reviews, and teaching students in EBSE gives us some confidence that it will work within software engineering as well. Therefore, to develop a more integrated approach to adopting research findings, we encourage both practitioners and researchers to develop coordinated mechanisms to support the continuing evolution of software engineering knowledge. This way, software organizations will be able to adopt good practice more quickly and with fewer risks, improve the quality of products, and reduce the risk of project failures.

References

AUSTRALIAN NATIONAL HEALTH AND MEDICAL RESEARCH COUNCIL, *How to use the evidence: assessment and application of scientific evidence*, Handbook Series on Preparing Clinical Practice Guidelines, NHMRC: Canberra, Feb. 2000.

BASILI, V.R. AND CALDIERA, G. Improve Software Quality by Reusing Knowledge and Experience, *Sloan Management Review*, 37(1):55-64, Fall 1995.

BERRY, D.M. AND TICHY, W.F. Comments on "Formal Methods Application: An Empirical Tale of Software Development", *IEEE Transactions on Software Engineering*, 29(6):567-571, June 2003.

COLLIER, B., DEMARCO, T., AND FEAREY, P. A Defined Process for Project Post Mortem Review, *IEEE Software*, 13(4):65-72, July/Aug. 1996.

COLLISON, C. AND PARCELL, G. *Learning to Fly: Practical Lessons from one of the World's Leading Knowledge Companies*, Capstone: New York, 2001.

DYBÅ, T., DINGSØYR, T., AND MOE, N.B. *Process Improvement in Practice – a Handbook for IT Companies*, The Kluwer International Series in Software Engineering, Boston: Kluwer Academic Publishers, 2004.

FEINSTEIN, A.R., and HOROWITZ, R.I. Problems with the "evidence" of "evidence-based medicine". Ann. J. Med., vol(103) pp529-535, 1997

KITCHENHAM, B.A., DYBÅ, T., AND JØRGENSEN, M. Evidence-based Software Engineering, Proceedings of the *26th International Conference on Software Engineering (ICSE 2004)*, Edinburgh, Scotland, 23-28 May, 2004.

KITCHENHAM, B.A., PFLEEGER, S.L., PICKARD, L.M., JONES, P.W., HOAGLIN, D.C., EL EMAM, K., AND ROSENBERG, J. Preliminary Guidelines for Empirical Research in Software Engineering, *IEEE Transactions on Software Engineering*, 28(8):721-734, Aug. 2002.

PICKARD, L.M., KITCHENHAM, B.A., AND JONES, P.W. Combining Empirical Results in Software Engineering, *Information and Software Technology*, 40(14):811-821, 1998.

PFLEEGER, S.L. AND MENEZES, W. Marketing Technology to Software Practitioners, *IEEE Software*, 17(1): 27-33, 2000.

POTTS, C. Software-Engineering Research Revisited, *IEEE Software*, 10(5):19-28, 1993.

SACKETT, D.L., STRAUS, S.E., RICHARDSON, W.S., ROSENBERG, W., AND HAYNES, R.B. *Evidence-Based Medicine: How to Practice and Teach EBM*, Second Edition, Churchill Livingstone: Edinburgh, 2000.

SOBEL, A.E.K. AND CLARKSON, M.R. Response to "Comments on 'Formal Methods Application: An Empirical Tale of Software Development'". *IEEE Transactions on Software Engineering*, 29(6):572-575, 2003.

ZELKOWITZ, M.V., WALLACE, D.R., AND BINKLEY, D.W. Experimental Validation of New Software Technology, in N. Juristo and Ana M. Moreno (Eds.) *Lecture Notes on Empirical Software Engineering*, World Scientific Publishing: Singapore, 2003, pp. 229-263.

Sidebar 2

EBSE Questions and Answers

1. *Is EBSE possible for ordinary practitioners?* Jørgensen has encouraging results from teaching final year college students the principles of EBSE, which shows that it is possible with relatively little effort to become better skilled at asking an answerable question, finding the best evidence both by searching the literature and by asking experts (i.e. practitioners and researchers), and critically appraising the available evidence.
2. *Can we develop appropriate infrastructure?* Kitchenham [1] has constructed guidelines for systematic review. These are being evaluated by several research groups.
3. *Does accumulation of evidence offer new insights?*
    1. Jørgensen and Moløkken [2] performed a systematic review of the size of software cost overruns. The systematic review showed that the results reported by the most influential study of the early 90s, the Standish Group's 1994 CHAOS report (i.e. cost overruns for challenged projects of 189%), were out of step with three other contemporary studies which showed cost overruns of 33%. A critical review of the Chaos Report and its use found that the many people who referenced the report were unaware that the overruns referred to "challenged" projects; that within the report "cost overrun" was not defined and was sometimes described as percentage of estimate (i.e. actual/estimate) and sometimes as percentage overrun (i.e. [estimate-actual]/estimate); and that the Standish group may have explicitly solicited failure stories. It seems therefore that there is evidence that project performance was never as bad as many people imagined and that subsequent "improvements" may be much smaller than many people have hoped.
    2. Jørgensen [3] performed a systematic review of studies of expert effort estimation. Although effort spent on developing cost estimation models is usually justified by suggesting human-based estimates are very poor, he found no evidence that models were superior to expert estimates. He identified a variety of conditions where expert estimates were likely to be superior and other conditions where models were likely to reduce situational or human bias.

References

[1] Barbara A. Kitchenham Procedures for performing Systematic Reviews. Joint Technical Report University of Keele, Dept. Computer Science TR.SE0401, National ICT Australia Empirical Software Engineering 0400011T.1, 2004.08.30.

[2] Magne Jørgensen and Kjetil Moløkken. How large are Software Cost Overruns? Critical Comments on the Standish Group's CHAOS Reports, http://www.simula.no/publication_one.php?publication_id=711, 2004.

[3] Magne Jørgensen. A Review of Studies on Expert Estimation of Software Development Effort. Journal Systems and Software, Vol 70, Issues 1-2, 2004, pp 37-60.