

Evidence-Based Software Engineering for Practitioners

Tore Dybå, *Simula Research Laboratory and SINTEF Information and Communication Technology*

Barbara A. Kitchenham, *National Information and Communications Technology Australia and Keele University*

Magne Jorgensen, *Simula Research Laboratory*

Software managers and practitioners often must make decisions about what technologies to employ on their projects. They might be aware of problems with their current development practices (for example, production bottlenecks or numerous defect reports from customers) and want to resolve them. Or, they might have read about a new technology and want to take advantage of its promised benefits. However, practitioners can have difficulty making informed decisions about

whether to adopt a new technology because there's little objective evidence to confirm its suitability, limits, qualities, costs, and inherent risks. This can lead to poor decisions about technology adoption, as Marvin Zelkowitz, Dolores Wallace, and David Binkley describe:

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.¹

For instance, enthusiasts of object-oriented programming were initially keen to promote the value of hierarchical models. Only later did experimental evidence reveal that deep hierarchies are more error prone than shallow ones.

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 that failure to undertake systematic reviews of medical research could cost lives and that experts' clinical judgment compared unfavorably with the results of systematic reviews. Since then, many medical researchers have

Software engineers might make incorrect decisions about adopting new techniques if they don't consider scientific evidence about the techniques' efficacy. They should consider using procedures similar to ones developed for evidence-based medicine.

adopted the evidence-based paradigm, and medical practitioners are now trained in this approach.² Although *evidence-based medicine* (EBM) has its critics,³ it is generally regarded as successful and has prompted many other disciplines (for example, psychiatry, nursing, social policy, and education) to adopt a similar approach.

Software companies are often under pressure to adopt immature technologies because of market and management pressures. We suggest that practitioners consider *evidence-based software engineering* as a mechanism to support and improve their technology adoption decisions.

The aim and methodology of EBSE

EBSE aims to improve decision making related to software development and maintenance by integrating current best evidence from research with practical experience and human values.⁴ This means we don't expect a technology to be universally good or universally bad, only more appropriate in some circumstances and for some organizations. Furthermore, practitioners will need to accumulate empirical research about a technology of interest and evaluate the research from the viewpoint of their specific circumstances.

This aim is decidedly ambitious, particularly because the gap between research and practice can be wide. EBSE seeks to close this gap by encouraging a stronger emphasis on methodological rigor while focusing on relevance for practice. This is important because rigor is necessary in any research that purports to be relevant. Moreover, because most SE research hasn't influenced industrial practice,⁵ there's also a pressing need to prevent SE research from remaining an ivory tower activity that emphasizes academic rigor over relevance to practice.

So, although rigor is a necessary condition for relevant SE research, it isn't sufficient. Medical evidence is based on rigorous studies of therapies given to real patients requiring medical treatment; laboratory experiments aren't considered to provide compelling evidence. This implies that SE shouldn't rely solely on laboratory experiments and should attempt to gather evidence from industrial projects, using observation studies, case studies, surveys, and field experiments. These empirical techniques don't have the scientific rigor of formal randomized experiments, but

they do avoid the limited relevance of small-scale, artificial SE experiments.

Furthermore, there are substantial problems with accumulating evidence systematically, and not only because accumulating evidence from different types of studies is difficult. A specific challenge in practicing EBSE is that different empirical studies of the same phenomenon often report different and sometimes contradictory results.⁶ Unless we can understand these differences, integrating individual pieces of evidence is difficult. This points to the importance of reporting contextual information in empirical studies to help explain conflicting research results.⁷

EBSE involves five steps:

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 customer's values and circumstances to make decisions about practice.
5. Evaluate performance and seek ways to improve it.

However, EBSE isn't a standalone activity. Much of what's needed to practice EBSE already exists in the concept of technology transfer⁸ and software process improvement.⁹ SPI involves several steps (each researcher and consultant has his or her own view of how many)—for example,

1. Identify a problem.
2. Propose a technology or procedure to address that problem.
3. Evaluate the proposed technology in a pilot project.
4. If the technology is appropriate, adopt and implement it.
5. Monitor the organization after implementing the new technology.
6. Return to step 1.

Thus, EBSE provides mechanisms to support various parts of SPI. In particular, EBSE focuses on finding and appraising an appropriate technology for its suitability in a particular situation. This is an area where SPI is usually

EBSE aims to improve decision making related to software development and maintenance by integrating current best evidence from research with practical experience and human values.

Asking the Right Question

Assume we want to ask, “Is pair programming useful?” According to what evidence-based medicine suggests, we should specify this question in more detail—for example, “Does pair programming lead to improved code quality when practiced by professional software developers?” Here, we’ve specified what intervention we’re interested in (pair programming), what population we’re interested in (professional software developers), and what effect we’re 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. We could have clearly stated that we wanted to compare pair programming with “individual programming.” Alternatively, we could compare it with “partner programming” (that is, the programmers work on different tasks on different computers but share the same physical office or workspace so that they can share ideas, thoughts, problems, and so forth). Regarding context, we could also have been more specific. We have, for example, not specified the software development organization’s nature, the developers’ skills and experience, or the software engineering environment being used.

However, searching for the keywords “pair,” “programming,” and “professional” in the abstracts of the nearly two million articles indexed in IEEE Xplore and the ACM Digital Library (see the “Useful Information Sources” sidebar) resulted in only four articles retrieved (search performed 22 Nov. 2004). Neither article examined pair programming using professionals as subjects. So, for software engineering problems, we might need to be less stringent in question formation, for these reasons:

- We have a much smaller body of empirical research than medicine has. We can’t afford to restrict our searches too much, or we won’t find any relevant studies.
- To support rational decision making, we’re usually interested in all possible outcomes. For example, we might not want to adopt a technique that results in very fast time to market if a side effect is poor operational reliability. This implies that, in particular, we shouldn’t restrict our questions too severely with respect to outcomes.

rather weak. People often assume 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 good process improvement. The only step in SPI that EBSE doesn’t support is technology infusion, which is supported by change management procedures and diffusion models.

Step 1: Ask an answerable question

EBSE doesn’t propose a specific method to

identify and prioritize problems. If you’re 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 staff members. Another form of help is the Goal-Question-Metric method,⁹ in which you derive questions from specific goals.

Once you’ve identified the problem, you need to specify an answerable question. Typical questions ask for specific knowledge about how to appraise and apply methods, tools, and techniques in practice. David Sackett and his colleagues suggest that well-formulated questions usually have three components:²

- The main intervention or action you’re 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 makes it easier not only to go from general problem descriptions to specific questions but also to think about what kind of information you need to answer the question.

In the SE context, factors to consider when deciding which question to answer first include these:

- Which question is most important to your customers?
- Which question is most relevant to your situation?
- Which question is the most interesting in the context of your business strategy?
- Which question is most likely to recur in your practice?
- Is the question possible to answer in the available time?

The main challenge in this step is, in other words, to convert a practical problem into a question that’s specific enough to be answered but not so specific that you don’t get any answers (see the “Asking the Right Question” sidebar).

Step 2: Find the best evidence

Finding an answer to your question includes selecting an appropriate information resource and executing a search strategy. However, you need to separate the question

you want to answer, the question implemented in the search keywords, and the questions answered in the studies found.

There are several information sources you can use. You can, for example, get viewpoints from your customers or the software's users, ask a colleague or an expert, use what you've learned as a student or in professional courses, use your own experience, or 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.⁷ The main source of such evidence is scientific journals. Additional sources include books, bibliographical databases, and the Internet. However, when you start searching for evidence, relevant evidence often isn't as easy to find as you might wish. Several thousand software-related publications are published each year; even if you work in a rather specialized area, keeping up-to-date by reading all the journals is almost impossible. For most practitioners, reading important magazines such as the *Communications of the ACM*, *Computer*, *IEEE Software*, and *IT Professional* would probably be enough to get a general overview of the latest SE developments.

Keeping up to date is much easier when you can use sources that combine results from independent empirical studies of a particular phenomenon.⁶ 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 studies correspond or contradict and uncovering gaps in your knowledge. However, *ACM Computing Surveys* is the only SE journal dedicated to systematic reviews. So, you need to search for such reviews in other journals as well. The situation is quite different in medicine, where the Cochrane Collaboration (www.cochrane.org) publishes and updates systematic reviews of all important areas of healthcare online.

In addition, you'll have to search for evidence in electronic databases on the Internet. By doing a literature search here, you get a more specific overview of the published research in your area of interest than is generally the case for magazines and systematic reviews (at least for the time being).

Useful Information Sources

- *IEEE Xplore* (<http://ieeexplore.ieee.org>) provides access to IEEE publications published since 1988 (and selected articles back to 1950) and to current IEEE standards. Access to abstracts and tables of contents is free. Access to full text requires IEEE membership, a subscription, or payment for individual articles.
- The *IEEE Computer Society Digital Library* (www.computer.org/publications/dlib) provides access to 22 IEEE Computer Society magazines and journals and more than 1,200 conference proceedings. Access to full text requires Computer Society membership, a subscription, or payment for individual articles.
- The *ACM Digital Library* (www.acm.org/dl) provides access to ACM publications and related citations. Full access requires ACM membership and possibly a subscription; nonmembers can browse the DL and perform basic searches.
- The *ISI Web of Science* (www.isinet.com/products/citation/wos) consists of databases containing information from approximately 8,700 journals in different research areas—for example, the Science Citation Index Expanded, Social Sciences Citation Index, and Arts & Humanities Citation Index. Users can perform searches, mark records, and link to full text.
- *EBSCOhost Electronic Journals Service* (<http://ejournals.ebsco.com>) provides access to over 8,000 e-journals. Users can view tables of contents and abstracts and can access the full text of the e-journals to which they subscribe.
- *CiteSeer* (<http://citeseer.nj.nec.com>), sponsored by the US National Science Foundation and Microsoft Research, indexes PostScript and PDF files of scientific research articles on the Web. Access is free.
- *Google Scholar* (<http://scholar.google.com>) indexes scholarly literature from all research areas, including abstracts, books, peer-reviewed papers, preprints, technical reports, and theses. Users can find scholarly literature from different publishers, professional societies, preprint repositories, and universities, as well as articles posted on the Web.

Many organizations index published articles in several databases; that is, they include bibliographic information such as author, title, and keywords. Such indexing simplifies searching for information regarding a problem area or finding an answer to a specific question. The "Useful Information Sources" sidebar gives examples of such databases.

Step 3: Critically appraise the evidence

Unfortunately, published research isn't always of good quality; the problem under study might be unrelated to practice, or the research method might have weaknesses such that you can't trust the results. To assess whether research is of good quality and is applicable to

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 study's design 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 study's results?
 - 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 and statistical significance?
4. Can the evidence be used in practice?
 - Are the study's findings transferable to other industrial settings?
 - Did the study evaluate all the important outcome measures?
 - 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?

Figure 1. A checklist for appraising published studies.

practice, you must be able to critically appraise the evidence.

For a nonscientist, evaluating an article's scientific quality can often be difficult (although for practitioners, evaluating the article's relevance to practice is often more important). Most journals use external referees to evaluate manuscripts before publication, which makes such manuscripts more trustworthy. This means that for research in non-refereed journals and conferences or on the Internet, the reader will need additional insight to evaluate the results and their relevance to practice. However, even in a reputable scientific journal, researchers might have difficulty agreeing about an experiment's rigor (see, for example, the discussion arising from a recent study of formal methods^{10,11}).

In EBM, the most convincing form of evidence is a systematic review of a series of double-blind randomized field trials. SE doesn't yet have many well-conducted replications of

rigorous experiments, so our empirical studies are much less reliable scientifically. SE researchers could provide more help to practitioners if they undertook and published more systematic reviews of important SE topics. However, until this happens, practitioners must be prepared to summarize evidence themselves. When you have evidence from different types of studies, you need some way to assess each study's quality. Figure 1 presents a checklist of factors to consider when evaluating an empirical study. In addition, Australian National Health and Medical Research Council guidelines discuss the relative trustworthiness of different types of empirical studies.¹²

Step 4: Apply the evidence

To employ the evidence in your decision making, you integrate it with your practical experience, your customers' requirements, and your knowledge of the concrete situation's specific circumstances, and then you apply it in practice. However, this procedure isn't straightforward.

Active use of new knowledge consists of applying or adapting specific evidence to a specific situation in practice. This contrasts with traditional, passive modes of transmitting information through teachers, books, manuals, colleagues, or business partners. Although such transmission can help in arranging the conditions required for learning to occur, it can't substitute for learning through direct experience. So, to practice EBSE, a software developer must commit to actively engaging 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 (method, tool, technique, or practice) you're evaluating. Some technologies apply at the level of the individual developer—for example, a developer can adopt evidence related to how best to comment programs. However, evidence related to the adoption of a computer-aided software engineering tool or a specific mathematically based formal method requires support from project and senior managers. Furthermore, even techniques that the individual

developer can adopt and evaluate have little impact unless they lead to a project-wide or organizational-wide process change.

So, it's at this point that you need to integrate EBSE with SPI. SPI relies on a systematic introduction and evaluation of proposed process change and, as we mentioned before, is often supported by change management processes. EBSE should provide the scientific basis for undertaking specific process changes, while SPI should manage the new technology's introduction.

Step 5: Evaluate performance

In EBM, the final step is for individual medical practitioners to reflect on their use of the EBM paradigm.² In SPI, the final step is usually to confirm that the process change has worked as expected. We believe both concerns are relevant for EBSE.

You need to consider how well you perform each step of EBSE and how you might improve your use of it. In particular, you should ask yourself how well you're integrating evidence with practical experience, customer requirements, and your knowledge of the specific circumstances.

Following SPI practice, you also must assess whether process change has been effective. However, environmental turbulence and rapid changes in technology often lead to the need to adapt and learn during projects. This involves a high degree of creativity and improvisation, which suggests that you can't wait until a project's end to draw out the lessons learned.¹³

After-action reviews,¹⁴ short meetings aimed at evaluating performance in the midst of action, are a simple way for individuals and teams to learn immediately from both successes and failures. All that's needed is a suitable task with an identifiable purpose and some metrics with which to measure performance. A typical AAR lasts for 10 to 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's important not to overreact. One isolated bad result shouldn't cause abandonment of a new method, unless strong

grounds exist 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 shouldn't mean that further monitoring is unnecessary. Barbara Kitchenham undertook a study of COCOMO in the early '80s. The first two projects on which she collected data were an almost perfect fit to the intermediate COCOMO model. Thereafter, however, no other project exhibited effort values anywhere near the COCOMO predictions.

When the project, or a major part of it, is completed, SPI principles suggest you must confirm that the expected improvement has taken place. A simple way to do this is to arrange a *postmortem analysis*.¹⁵ A PMA is similar to an AAR but is conducted in more depth. Instead of 10 to 20 minutes, a PMA typically lasts from a couple of hours to a full day. It aims to capture lessons and insights (both good and bad) for future projects by evaluating these 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 them?

A PMA results mainly in better evidence regarding the specific process or technology—evidence that you might reuse as guidelines for the future, in the form of experience notes, new or improved checklists, improved development models, and a general understanding of what works and what doesn't in projects in your organization. PMAs and, when available, organization-wide measurement programs provide the information needed to restart the EBSE cycle, letting you identify and prioritize product and process problems by collating experiences from different projects.

Discussion

Although it's important for software practitioners to base their choice of development methods on available scientific evidence, this isn't necessarily easy. EBM arose because medical practitioners were overwhelmed by the large number of scientific studies; in SE our problems are rather different. There are relatively few studies, as our pair-programming

A postmortem analysis results mainly in better evidence regarding the specific process or technology—evidence that you might reuse as guidelines for the future.

Evidenced-Based Software Engineering Q&A

Is EBSE possible for ordinary practitioners?

Magne Jørgensen has obtained encouraging results from teaching EBSE principles to final-year university students. The results show that with relatively little effort, people can become better skilled at asking an answerable question, finding the best evidence by searching the literature and by asking experts (that is, practitioners and researchers), and critically appraising the available evidence.

Tore Dybå was external examiner of the reports that these students produced using EBSE principles to investigate a software engineering technology. In his opinion, the quality of these reports was at least as good as more conventional reports on software engineering topics.

Can we develop appropriate infrastructure?

Barbara Kitchenham has constructed guidelines for systematic reviews,¹ which several research groups are evaluating.

Does accumulation of evidence offer new insights?

Magne Jørgensen and Kjetil Moløkken performed a systematic review of the size of software cost overruns.² Their review showed that the results reported by the Standish Group's 1994 CHAOS report, the most influential study of the early '90s, were out of step with the results of three other contemporary studies. (The CHAOS report showed cost overruns of 189 percent; the other studies showed cost overruns of approximately 33 percent.) Differences in cost overrun measurements between the CHAOS study and the other studies were unable to explain this difference. A critical examination of the report revealed several problems, including these:

- The Standish Group reported in their 1998 CHAOS report an average cost overrun of 69 percent; that is, an improve-

ment from 189 percent to 69 percent overrun in about four years. This is hardly likely to have happened.

- The CHAOS report didn't define how cost overrun was measured and described it inconsistently.
- The CHAOS report didn't report how the included projects were selected. Formulations in the CHAOS report suggest that the Standish Group collected mainly failure stores in 1994. This explains the high cost overrun number but also means that the study results can't be used as indicators of the software industry in general.

So, evidence exists that project performance was never as bad as many people imagined and that subsequent "improvements" might be much smaller than many people have hoped.

In addition, Jørgensen performed a systematic review of studies of expert effort estimation.³ Although effort spent on developing cost estimation models is usually justified by the argument that human-based estimates are poor, Jørgensen 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. B.A. Kitchenham, *Procedures for Performing Systematic Reviews*, tech. report SE0401, Dept. of Computer Science, Univ. of Keele, and tech. report 0400011T.1, Empirical Software Eng., National Information and Communications Technology Australia, 30 Aug. 2004.
2. M. Jørgensen and K. Moløkken, "How Large Are Software Cost Overruns? Critical Comments on the Standish Group's CHAOS Reports," Simula Research Laboratories, 2004; www.simula.no/publication_one.php?publication_id=711.
3. M. Jørgensen, "A Review of Studies on Expert Estimation of Software Development Effort," *J. Systems and Software*, vol. 70, nos. 1-2, 2004, pp. 37-60.

example (see the "Asking the Right Question" sidebar) showed. Furthermore, when evidence is available, software practitioners still have difficulty judging the evidence's quality and assessing what the evidence means in terms of their specific circumstances. This implies that given the current state of empirical SE, practitioners will need to adopt more proactive search strategies such as approaching experts, other experienced practitioners, and researchers directly.

Because a basic idea behind EBSE is to establish a fruitful cooperation between research and practice, a closer link should exist between research and practice so that research is

relevant to practitioners' needs and practitioners are willing to participate in research.

You might have noticed that we've offered no evidence of EBSE's benefits. Although we have no examples of other practitioners using EBSE, the sidebar "Evidenced-Based Software Engineering Q&A" presents examples of our own use of EBSE. On the basis of this experience, and other ongoing industrial and educational initiatives in which we're engaged, we believe that evidence-based practice is possible and potentially useful for software practitioners.

However, evidence-based practice also places requirements on researchers. We recommend that researchers adopt as much of the

evidence-based approach as is possible. Specifically, this includes being more responsive to practitioners' needs 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 in order to accumulate reliable evidence about SE topics. Researchers also need to report their results in a manner that's accessible to practitioners.

Evidence-based practice works in medicine.² Furthermore, our experience from undertaking empirical studies, systematic reviews, and teaching students in EBSE gives us some confidence that it will also work in software engineering. So, 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 SE 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

1. M.V. Zelkowitz, D.R. Wallace, and D.W. Binkley, "Experimental Validation of New Software Technology," *Lecture Notes on Empirical Software Engineering*, World Scientific, 2003, pp. 229–263.
2. D.L. Sackett et al., *Evidence-Based Medicine: How to Practice and Teach EBM*, 2nd ed., Churchill Livingstone, 2000.
3. A.R. Feinstein and R.I. Horowitz, "Problems with the 'Evidence' of 'Evidence-Based Medicine,'" *Am. J. Medicine*, vol. 103, no. 6, 1997, pp. 529–535.
4. B.A. Kitchenham, T. Dybå, and M. Jørgensen, "Evidence-Based Software Engineering," *Proc. 26th Int'l Conf. Software Eng. (ICSE 2004)*, IEEE CS Press, 2004, pp. 273–281.
5. C. Potts, "Software-Engineering Research Revisited," *IEEE Software*, vol. 10, no. 5, 1993, pp. 19–28.
6. L.M. Pickard, B.A. Kitchenham, and P.W. Jones, "Combining Empirical Results in Software Engineering," *Information and Software Technology*, vol. 40, no. 14, 1998, pp. 811–821.
7. B.A. Kitchenham et al., "Preliminary Guidelines for Empirical Research in Software Engineering," *IEEE Trans. Software Eng.*, vol. 28, no. 8, 2002, pp. 721–734.
8. S.L. Pfleeger and W. Menezes, "Marketing Technology to Software Practitioners," *IEEE Software*, vol. 17, no. 1, 2000, pp. 27–33.

About the Authors



Tore Dybå is the chief scientist at SINTEF Information and Communication Technology and a visiting scientist at the Simula Research Laboratory. His research interests include empirical software engineering, software process improvement, and organizational learning. He received his Dr. Ing. in computer and information science from the Norwegian University of Science and Technology. He's a member of the International Software Engineering Research Network and the IEEE Computer Society. Contact him at SINTEF ICT, NO-7465 Trondheim, Norway; tore.dyba@sintef.no.

Barbara A. Kitchenham is a professor of quantitative software engineering at Keele University and a senior principal researcher at National Information and Communications Technology Australia. Her main research interest is software metrics and its application to project management, quality control, risk management, and evaluation of software technologies. She's particularly interested in the limitations of technology and the practical problems associated with applying measurement technologies and experimental methods to software engineering. She's a Chartered Mathematician and a fellow of the Institute of Mathematics and Its Applications and of the Royal Statistical Society. Contact her at National ICT Australia, Locked Bag 9013, Alexandria, NSW 1435, Australia; barbara.kitchenham@nicta.com.au.



Magne Jørgensen is a professor in software engineering at the University of Oslo and a member of the Simula Research Laboratory's software engineering research group. He has about 10 years' industry experience as a software developer, project leader, and manager. He received his Dr. Scient. in informatics from the University of Oslo. Contact him at the Simula Research Laboratory, PO Box 134, NO-1325 Lysaker, Norway; magne.jorgensen@simula.no.

9. V.R. Basili and G. Caldiera, "Improve Software Quality by Reusing Knowledge and Experience," *Sloan Management Rev.*, vol. 37, no. 1, 1995, pp. 55–64.
10. D.M. Berry and W.F. Tichy, "Comments on 'Formal Methods Application: An Empirical Tale of Software Development,'" *IEEE Trans. Software Eng.*, vol. 29, no. 6, 2003, pp. 567–571.
11. A.E.K. Sobel and M.R. Clarkson, "Response to 'Comments on "Formal Methods Application: An Empirical Tale of Software Development,"'" *IEEE Trans. Software Eng.*, vol. 29, no. 6, 2003, pp. 572–575.
12. *How to Use the Evidence: Assessment and Application of Scientific Evidence*, Australian Nat'l Health and Medical Research Council, Feb. 2000.
13. T. Dybå, T. Dingsoyr, and N.B. Moe, *Process Improvement in Practice: A Handbook for IT Companies*, Kluwer Academic, 2004.
14. C. Collison and G. Parcell, *Learning to Fly: Practical Lessons from One of the World's Leading Knowledge Companies*, Capstone, 2001.
15. B. Collier, T. DeMarco, and P. Fearey, "A Defined Process for Project Postmortem Review," *IEEE Software*, vol. 13, no. 4, 1996, pp. 65–72.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.