*"Perspectives on Data Science for Software Engineering":*

Tore Dybå, Gunnar Bergersen and Dag I.K. Sjøberg

**Evidence-Based Software Engineering**

“I believe in evidence. I believe in observation, measurement, and reasoning, confirmed by independent observers. I'll believe anything, no matter how wild and ridiculous, if there is evidence for it. The wilder and more ridiculous something is, however, the firmer and more solid the evidence will have to be.”

(Isaac Asimov)

A decade ago, Kitchenham, Dybå and Jørgensen coined the term and pro­vided the foundations for *evidence-based software engineering* (EBSE). A trilogy of papers was written for researchers (Kitchenham et al., 2004), practitioners (Dybå et al., 2005), and educators (Jørgensen et al. 2005). They suggested that practitioners consider EBSEas a mechanism to support and improve their technology adoption decisions, and that researchers should use systematic literature reviews (SLRs) as a methodology for performing unbiased aggregation of empirical results. This spurred a large international activity and a renewed focus on research methods and theory, and on the future of empirical methods in SE research (Sjøberg et al., 2007).

**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.

Based on the stages in evidence-based medicine, EBSE involves five steps (Dybå et al., 2005):

1. *Ask an answerable question*. The main challenge in this step is 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. Partitioning the question into the main intervention or action you’re interested in, the context or specific situation of interest, and the main outcomes or effect of interest, 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.
2. *Find the best evidence*. Finding an answer to your question includes selecting an appropriate information resource and executing a search strategy. There are several information sources you can use. You can, for example, get viewpoints from your customers or the software’s users, colleagues 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.
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 practice, you must be able to critically appraise the evidence.
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 and, among other things, depends on the type of technology you’re evaluating.
5. *Evaluate performance*. The final step is 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. You must also assess whether the change has been effective.

It’s important to note that recommendations on evidence-based medicine tend to be context independent and implicitly universal, while software engineering prescriptions are contingent and sensitive to variation in the organizational context.

**Contextualizing evidence**

What works for whom, where, when, and why is the ultimate question of EBSE (Dybå, 2013). Still, the empirical research seems mostly concerned with identifying universal relationships that are independent of how work settings and other contexts interact with the processes important to software practice. Questions of “What is best?” seem to prevail. For example, “Which is better: pair or solo programming? test-first or test-last?”

However, just as the question of whether a helicopter is better than a bicycle is meaningless, so are these questions because the answers depend on the settings and goals of the projects studied. Practice settings are rarely, if ever, the same. For example, the environments of software organizations differ, as do their sizes, customer types, countries or geography, and history. All these factors influence engineering practices in unique ways. Additionally, the human factors underlying the organizational culture differ from one organization to the next and also influence the way software is developed.

We know these issues and the ways they interrelate are important for the successful uptake of research into practice. However, the nature of these relationships is poorly understood. Consequently, we can’t a priori assume that the results of a particular study apply outside the specific context in which it was run. Taking something out of context leads to misunderstanding; for instance, “I am attached to you” has very different meanings for a person in love and a handcuffed prisoner.

**Strength of evidence**

Several systems exist for making judgments about the strength of evidence of a body of knowledge. Most of these systems suggest that the strength of evidence can be based on a hierarchy with evidence from systematic reviews and randomized experiments at the top of the hierarchy and evidence from observational studies and expert opinion at the bottom of the hierarchy. The inherent weakness with such evidence hierarchies is that randomized experiments are not always feasible and that, in some instances, observational studies may provide better evidence.

To cope with these weaknesses of evidence hierarchies, Dybå and Dingsøyr (2008) used the GRADE working group definitions (Atkins et al., 2004, see Table 1) to grade the overall strength of the evidence of their systematic review of empirical studies of agile software development as high, moderate, low, or very low.

Table 1. Definitions used for grading the strength of evidence (Atkins et al., 2004).

|  |  |
| --- | --- |
| High | Further research is very unlikely to change our confidence in the estimate of effect. |
| Moderate | Further research is likely to have an important impact on our confidence in the estimate of effect and may change the estimate. |
| Low | Further research is very likely to have an important impact on our confidence in the estimate of effect and is likely to change the estimate. |
| Very low | Any estimate of effect is very uncertain. |

According to GRADE, the strength of evidence can be determined on the basis of the combination of four key elements: study design (e.g., there were only two randomized trials in the review), study quality (i.e., how well methods were described, how issues of bias, validity, and reliability were addressed, and how methods of data collection and analysis were explained), consistency (i.e., the similarity of estimates of effect across studies), and directness (i.e., the extent to which the people, interventions, and outcome measures are similar to those of interest).

Combining these four components, Dybå and Dingsøyr found that the strength of the evidence regarding the benefits and limitations of agile methods, and for decisions related to their adoption, was *very low*. Hence, they concluded that any estimate of effect that is based on evidence of agile software development from research is very uncertain. However, this review was published in 2008, and many more empirical studies of agile software development have been published since then.

**Evidence and theory**

Consider, for example, the “reversing effect theory” (Cf. Sjøberg et al. this book). Why would this theory be worthy of being believed by contemporary software engineers? A plausible answer is that the evidence possessed by contemporary software engineers strongly suggests that it is true. Paradigmatically, those theories that are worthy of being believed enjoy such status in virtue of the availability of evidence sufficient to justify belief in their truth.

Evidence that supports or tells in favor of a given theory *supports* that theory. On the other hand, evidence that tells against a theory *weakens* that theory. Of course, a given piece of evidence might confirm or disconfirm a theory to a greater or lesser degree. ‘Verification’ signifies the maximal degree of confirmation: evidence may *support* a theory in the sense it conclusively establishes that the theory in question is true (even though one can never be absolutely certain). At the opposite end of the spectrum, *falsification* signifies the maximal level of disconfirmation: evidence falsifies a theory just in case it conclusively establishes that the theory in question is false (Kelly, 2008).

In considering questions about how a given body of evidence bears on a theory, it is crucial to distinguish between the *balance* of the evidence and its *weight*. Intuitively, the balance of the evidence concerns how decisively the evidence tells for or against the theory. On the other hand, the weight of the evidence is a matter of how substantial the evidence is. However, for any body of evidence, there will always be alternative theories that constitute equally good explanations of that evidence (Kelly, 2008).

**References**

D. Atkins et al., “Grading quality of evidence and strength of recommendations,” *BMJ*, no. 1490, vol. 328, 2004.

T. Dybå, “Contextualizing Empirical Evidence,” *IEEE Software*, 30(1): 81-83, 2013.

T. Dybå, B.A. Kitchenham, and M. Jørgensen, “Evidence-based Software Engineering for Practitioners,” *IEEE Software*, 22(1): 58-65, 2005.

T. Dybå and T. Dingsøyr, “Empirical Studies of Agile Software Development: A Systematic Review,” *Information and Software Technology*, 50(9-10): 833-859, 2008.

M. Jørgensen, T. Dybå, and B.A. Kitchenham, “Teaching Evidence-Based Software Engineering to University Students,” *Proc. 11thInternational Software Metrics Symposium (Metrics 2005)*, Como, Italy, 19-22 Sept. 2005.

T. Kelly, “Evidence: Fundamental Concepts and the Phenomenal Conception,” *Philosophy Compass*, 3/5: 933-955, 2008.

B.A. Kitchenham, T. Dybå, and M. Jørgensen, “Evidence-based Software Engineering,” *Proc.* *26th International Conference on Software Engineering (ICSE 2004)*, Edinburgh, Scotland, 23-28 May, pp. 273-281, 2004.

D.I.K. Sjøberg, T. Dybå, and M. Jørgensen, “The Future of Empirical Methods in Software Engineering Research,” *29th International Conference on Software Engineering (ICSE'07)*, Minneapolis, Minnesota, USA, 20-26 May, *Future of Software Engineering*, L. Briand and A. Wolf (Eds.), IEEE Computer Society Press, pp. 358-378, 2007.