Voice of Evidence

Contextualizing Empirical Evidence

Tore Dybå

What Works for whom, when, and why is the ultimate question of evidence-based software engineering.1 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.

Here, I offer an overview of how context affects empirical research and how to better contextualize empirical evidence so that others can better understand what works for whom, where, when, and why.2

What Is Context?

The word contextus is Latin for weaving together or making a connection. Approaches to context and contextual dimensions vary widely, reflecting different philosophical stances and practical orientations. In linguistics, for example, context refers to how readers can infer a passage’s meaning by referring to its intratextual clues—something that transcends the text itself.3

In other words, trying to make sense of a single word in a sentence or of a sentence in a paragraph by looking at it in isolation from the rest of the text can be problematic, even if you’re technically knowledgeable about the linguistic meanings. For instance, “I am attached to you” has very different meanings for a person in love and a handcuffed prisoner.4

Taking something out of context leads to misunderstanding; there is no meaning without context. On the other hand, even if you’re not familiar with the specific meaning or meanings of a word or sentence, you can correctly infer the meaning by connecting the word or sentence with the rest of the text.

According to Gary Johns,5 research could benefit from careful consideration of journalism’s five Ws in study design and reporting:

- Who
- What
- When
- Where
- Why

Research could benefit from careful consideration of journalism’s five Ws in study design and reporting.
Because several lists of possible discrete context variables for software engineering are proposed elsewhere (for example, see Paul Clarke and Rory V. O’Connor), I won’t consider the details here. Therefore, the elements of the technical, social, and environmental context variables in Figure 1 aren’t meant to be exhaustive but only to present important examples.

However, we must be cautious in selecting discrete context variables. Evaluating even a small selection of such variables quickly runs into the difficulties of combinatorial complexity. It doesn’t take many variables before the variable-oriented logic becomes absurd. My focus is therefore on the broader, omnibus perspective of what, who, where, when, and why.

**Omnibus Context: The Broad Perspective**

“What” constitutes a study’s substantive content—the factors (variables, constructs, concepts) or treatments that logically relate to explanations of the phenomena of interest. Although it might seem obvious—and maybe not strictly part of the context—what is actually studied isn’t always clear. A typical problem is the arbitrary description of measures of the constructs being studied and the lack of justification for variable encoding. For example, if software quality is the phenomenon of interest but the study measures only defects, it would require quite a bit of justification to link the broad concept of quality with the narrow construct of defects and the specific way in which they’re coded.

“Who” refers to the occupational and demographic context. It concerns both the direct research participants and those who surround them. The study must clearly identify the population about which it intends to make claims and must include and describe representative subjects of that population. The usual assumption is that the target population is professional software developers. However, just stating the occupational context or even a participant’s personality might not account for skill differences that could affect a study’s outcome. In many domains, higher skill levels mean fewer errors and faster execution times for a given task. Describing the “who” of study participants, especially their skill levels, has an important impact on the transfer of technology to other users.

“Where” the software development occurs can also affect a study’s results. An important distinction is whether it occurs in an artificial laboratory or a more realistic industry setting. For example, configuring an experimental environment requires implementing an infrastructure of supporting technology (processes, methods, tools, and so on) that resembles an industrial development environment. Because the logistics are simpler, a classroom is often used instead of a real workplace. An experiment conducted at a worksite with professional development tools also implies less experimental control than one conducted in a classroom setting with pens and paper. Worksite research also requires consideration of location effects such as economic conditions and organizational and national culture.

“When” refers to the time at which the research was conducted or research events occurred. It includes when the data was collected and reflects the role of temporal factors in the research. Time affects the sociotechnical relationships that surround all aspects of software development, and it’s especially important for research that deals with software product life cycles. Time is also related to whether the study is cross-sectional or longitudinal. The importance of the temporal dimension in empirical evidence is underscored by repeated calls for longer duration of experimental tasks and more longitudinal research in software engineering.

Time is often an important variable in software engineering experiments as well. Asking subjects to solve tasks with satisfactory quality in as short a time as possible effectively mirrors the relatively high time pressure of most software engineering jobs. However, if the time pressure is too high, the task solution quality might be reduced to the point of becoming meaningless for any subsequent analyses. It’s a challenge to put realistic time pressure on experimental subjects. How to best deal with this challenge depends to some extent on the experiment’s size, duration, and location. A promising first step involves...
efforts to combine time and quality as a task performance measure for programming skill in both industry and research settings.7

“Why” refers to the rationale for the research or data collection. Why data is collected can have a compelling impact on organizational behavior and associated research. For example, an experimental setting would ideally either reflect the subjects’ organizational setting or let them see some professional benefit from the experimental tasks. This would motivate them to put more effort and thought into the study. Motivation can be a problem when subjects are asked to work on toy problems, are given unrealistic processes, or see some other disconnection between the study and their professional experience.10

The Way Forward

We need to shift focus away from a checklist-based approach to contextualizing empirical evidence in favor of a more dynamic view of software practice. Instead of seeing a set of discrete variables that statically surround parts of practice, we can view the relationships between empirical evidence and context as a process that emerges and changes through time and space. It’s crucial to acknowledge that any definition of context can occur only in relation to a specific practice situation. Consequently, I dispute any attempt to provide a general framework or checklists of specific factors intended to describe the context of local, situated practice.

Given that any study offers an infinite number of contextual factors and combinations to consider, the decision as to the parameters along which to contextualize should be no different from the decision regarding which variables to control. Both decisions should be grounded in the goals and theories relevant to the phenomenon under study.

Contextualization requires immersion and a focus on relevant phenomena, which means that software engineering researchers should invest considerable time within the practice they wish to understand. Immersion will also help us move our discipline in a more useful direction that will counter the common criticism that empirical evidence is often irrelevant for software organizations and their members. Accounting for the larger sociotechnical frameworks that embed empirical evidence makes it more comprehensible. It’s all about context, interpretation, and evaluation. However, what counts as context will depend on the substantive problem under scrutiny. Generalized lists of discrete variables won’t capture it.

To move beyond simple assertions that the context is important, we must articulate more clearly how contextual influences operate. At a minimum, we should expect an empirical study to report what was studied, who was studied, where they were studied, when they were studied, and why they were studied.

References


TORE DYBÅ is chief scientist and research manager at SINTEF and a professor at the University of Oslo, Norway. He coedits the Voice of Evidence for IEEE Software. Contact him at tore.dyba@sintef.no.